N° d'ordre: 1411

50376
1987
257

THESE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le grade de

DOCTEUR TROISIEME CYCLE

spécialité :

AUTOMATIQUE ET INFORMATIQUE INDUSTRIELLE

par

Roger LAWSON
maître es Sciences

sur

LA REALISATION ET LA MISE AU POINT D'UN LOGICIEL DIDACTIQUE DE SIMULATION HYBRIDE MULTITACHE

Soutenue le 29 Octobre 1987 devant la Commission d'Examen

Membres du jury :

President	M . P.VIDAL	Professeur
Rapporteur	M. J.C.GENTINA	Professeur
Examinateurs	Mme. G.DAUPHIN	Maître de conference
	M. D.CORBEEL	Maître de conference

AVANT-PROPOS

Le travail présenté dans ce mémoire a été effectué au Laboratoire d'Informatique Industrielle de l'IDN sous la direction scientifique de Monsieur le Professeur GENTINA, Directeur de l'IDN.

Je remercie très vivement Monsieur le Professeur VIDAL, Directeur du Laboratoire du Centre d'Automatique de Lille I d'avoir accepté de présider le jury auquel je soumets ce mémoire.

Je suis très reconnaissant à Monsieur GENTINA, Monsieur CORBEEL qui ont été les promoteurs de cette recherche et qui en ont suivi sans cesse le déroulement, faisant tout pour que rien ne puisse arrêter mon travail, tant sur le plan scientifique que sur le plan humain.

Je suis très flatté de la présence dans ce jury de Madame le Professeur DAUPHIN qui m'a fait profiter de ses expériences dans le domaine de la simulation et je tiens tout particulièrement à l'en remercier.

Enfin, je remercie les personnes qui ont assuré la réalisation matérielle de ce mémoire, tout particulièrement Monsieur MAYET, Mesdames BOURDREL et BUKOWSKI, pour la dactylographie et Madame FERRAR, pour la reprographie.

S O M M A I R E

CHAPITRE I

PRESENTATION DU LOGICIEL DE BASE TUTSIM

CHAPITRE II

NOUVELLE CONFIGURATION DE TUTSIM
AUTOUR DU VAX 11/750

CHAPITRE III

EXTENSION DU LOGICIEL DE BASE POUR L'EXECUTION PARALLELLE : CONCEPT DE PRODUCTEUR-CONSOMMATEUR DANS LE TRAITEMENT EN MODE PARALLELLE

CHAPITRE IV

EXEMPLES D'APPLICATION

INTRODUCTION GENERALE

INTRODUCTION

La plus grande partie des processus industriels est issue des systèmes numériques généralement non linéaires et multivariables ; les difficultés liées à la commande de ces systèmes sont de deux ordres :

- en premier lieu, les variables nécessaires à la description du système peuvent être de nature différente ; aussi, a-t-on recours aux mathématiques permettant de définir un modèle de référence semblable au processus, ce que les automaticiens appellent la modélisation,

- en second lieu, ces systèmes se caractérisent, dans la plupart des cas, par une multiplicité des objectifs de commande, le plus souvent antagonistes, donc difficiles à formaliser. C'est ainsi que les objectifs d'une unité de production qui portent principalement sur la qualité et la quantité des produits ouvrés sont en concurrence avec le temps nécessaire à la fabrication et donc les coûts.

L'évaluation du modèle et donc sa résolution mathématique conduisent naturellement à la mise en oeuvre des outils permettant le calcul simultané des grandeurs physiques (calculateurs analogiques) et des grandeurs numériques (calculateurs numériques) ; le premier constitué d'une série d'opérateurs effectuant chacun une opération mathématique bien définie résoud un problème par connexions d'opérateurs (par cablage) ; le second jouant le rôle de contrôle et de décision établit une correspondance entre les grandeurs analogiques et numériques à l'aide des informations sous forme de programme.

Une différence fondamentale liée à la nature des grandeurs et de leur

évolution temporelle entre ces deux calculateurs est que, dans le premier cas, les informations sont définies et traitées sous forme continue, dans le second, elles sont présentées sous forme séquentielle et traitées de manière discrète.

La phase expérimentale du processus, réalisée grâce à la simulation de son modèle mathématique, nécessite le couplage de ces deux types de calculateur offrant ainsi au domaine de la recherche appliquée l'élaboration d'un outil puissant de simulation en temps réel des systèmes complexes : les calculateurs hybrides.

Ce couplage est dans une certaine mesure facilité par le développement des mini-calculateurs digitaux industriels assurant la gestion temps réel des échanges avec leur environnement ; ceci nécessite des programmes moniteurs temps réel pour assurer, d'une part, la gestion des tâches correspondant au matériel et au logiciel spécialisé de simulation et, d'autre part, le pilotage des entrées/sorties des systèmes, plus précisément les périphériques informatiques.

Néanmoins, un tel couplage dont le principe de fonctionnement d'ensemble équivaut à un pilotage du calculateur analogique par le calculateur digital tend de nos jours à disparaître en raison, d'une part, des possibilités de plus en plus croissantes des traitements simultanés favorisés par l'accroissement de la puissance et de la vitesse d'exécution des calculs qu'offrent les calculateurs numériques et, d'autre part, des limitation du coût et de la non flexibilité des opérateurs analogiques.

Aussi, le calculateur hybride se réduit à la mise en oeuvre d'un calculateur unique traitant en mode parallèle les informations de type continu par discrétisation temporelle et spatiale.

L'utilisation de tels calculateurs soulève deux problèmes connexes :

(i) le choix du matériel adapté,

(ii) la nature des algorithmes de calcul modélisant les opérateurs analogiques et le logiciel de simulation temps réel dont le volume est directe-

ment lié à la capacité mémoire du calculateur. Le caractère purement informatique de la simulation "hybride" permet une large souplesse dans le choix de tels ou tels langages évolués pour le développement et la mise au point du logiciel de simulation ; ce qui a conduit tout naturellement à de nombreuses recherches dans ce domaine dont voici les plus récentes :

- le langage de simulation hybride : "SHD" /MARQUETTE,1/, développé sur calculateur T1000 (19bits) couplé à un calculateur hybride. Ce langage assure une transcription quasi-directe des graphes fonctionnels, tels que les réseaux de Pétri ou le Grafcet,

- le langage de programmation des systèmes continus : "CSPL" ou "CSMP", /TOMOYUKI,MATSUZAKA,2/ qui est un logiciel de simulation de blocs orientés à partir des deux méthodes de représentation des systèmes de contrôle : la première basée sur la représentation des équations sous forme des fonctions de transfert, la seconde repose sur la représentation des équations dans l'espace d'état,

- le simulateur transportable : "PSI", /R. HERMAN, M. KOHNE,3/, est aussi un logiciel de simulation de diagramme de blocs orientés. Ecrit en langage "C", il offre, par rapport au langage précédent, un avantage du point de vue gain de temps de calcul et d'espace mémoire,

- le logiciel de simulation "TUTSIM", /MEERMANN,4/, qui est à la base du travail que nous présentons dans ce mémoire et qui sera développé dans le chapitre suivant,

- le programme "DASP", /F. GANSCH,5/, dont le concept de simulation est proche de TUTSIM ; il présente cependant une particularité due à la manipulation de paramètres auxilliaires reliés aux paramètres réels du modèle à travers des expressions arithmétiques préprogrammées,

- le logiciel de simulation en mode parallèle des systèmes dynamiques à degrés de complexité variables : "RAMS", /V. MALBASA,6/, est un simulateur temps réel des processus modélisés par un système d'équations différentielles non linéaires et adapté aux calculateurs multiprocesseurs. Bien que RAMS soit un calculateur de blocs au même titre que les autres, il dénote quelques

particularités très remarquables qui, comparées aux logiciels précédemment cités, méritent d'être détaillées et dont les points essentiels sont :

. l'aptitude d'extensions modulaires du "SOFT et HARD" permettant d'adapter les différentes tâches de simulation aux contraintes temps du processus étudié,

. l'aptitude à analyser le modèle mathématique du processus et de passer de manière automatique du travail monoprocesseur à un travail pultiprocesseur ; ce qui, de toute évidence, sous-entend l'intégration d'un module qui permet au logiciel de donner avec satisfaction le rendement d'un travail multiprocesseur.

Le concept de simulation en modèle parallèle s'appuie sur la représentation du modèle mathématique du système étudié sous forme d'équation d'état à partir de laquelle on déduit un algorithme parallèle de simulation et le travail d'exécution sous RAMS de cet algorithme. Afin de préciser davantage ce concept, il convient de donner un exemple simple d'un système dont le modèle mathématique est décrit par l'équation d'état (1) :

$$(1)\quad \begin{cases} \dot{x} = f(x,u,t) \\ y = g(x,u,t) \end{cases}$$

dans laquelle $x \in \mathbb{R}^u$ est le vecteur du système, $u \in \mathbb{R}^p$ est le vecteur de commande, $y \in \mathbb{R}^q$ le vecteur de sortie, f et g représentent des applications de $\mathbb{R}^q \times T \times \mathbb{R}^p \rightarrow \mathbb{R}^u$...

De cette équation, on déduit un premier algorithme parallèle définissant l'ensemble de fonctions primitives telles que :

(2) $A = \{B, <, T\}$ où :

- B est un bloc élémentaire parmi les Bi (i=1,2,...,n) blocs,
- < signe de relation d'ordre entre un bloc suivant et son prédécesseur,
- T le temps au bout duquel toutes les opérations doivent être exécutées.

L'exécution de cet algorithme est engendrée par un deuxième algorithme :

(3) $J = (m,z,<,T)$ où

- m désigne le nombre de microprocesseurs,
- z l'ensemble des tâches,
- < signe de relation d'ordre entre les tâches,
- T la contrainte temps.

Notons que les équations (1), (2) et (3) définissent la phase de traitement qui débouche sur la phase finale qui est la simulation automatique du modèle grâce aux trois sous-programmes suivants :

- S.P d'analyse algorithmique et son adaptation au mode parallèle de travail de RAMS,

- S.P d'exécution des blocs,

- S.P de contrôle du processus.

Toute la procédure de travail sous "RAMS" est illustrée par la figure 1.

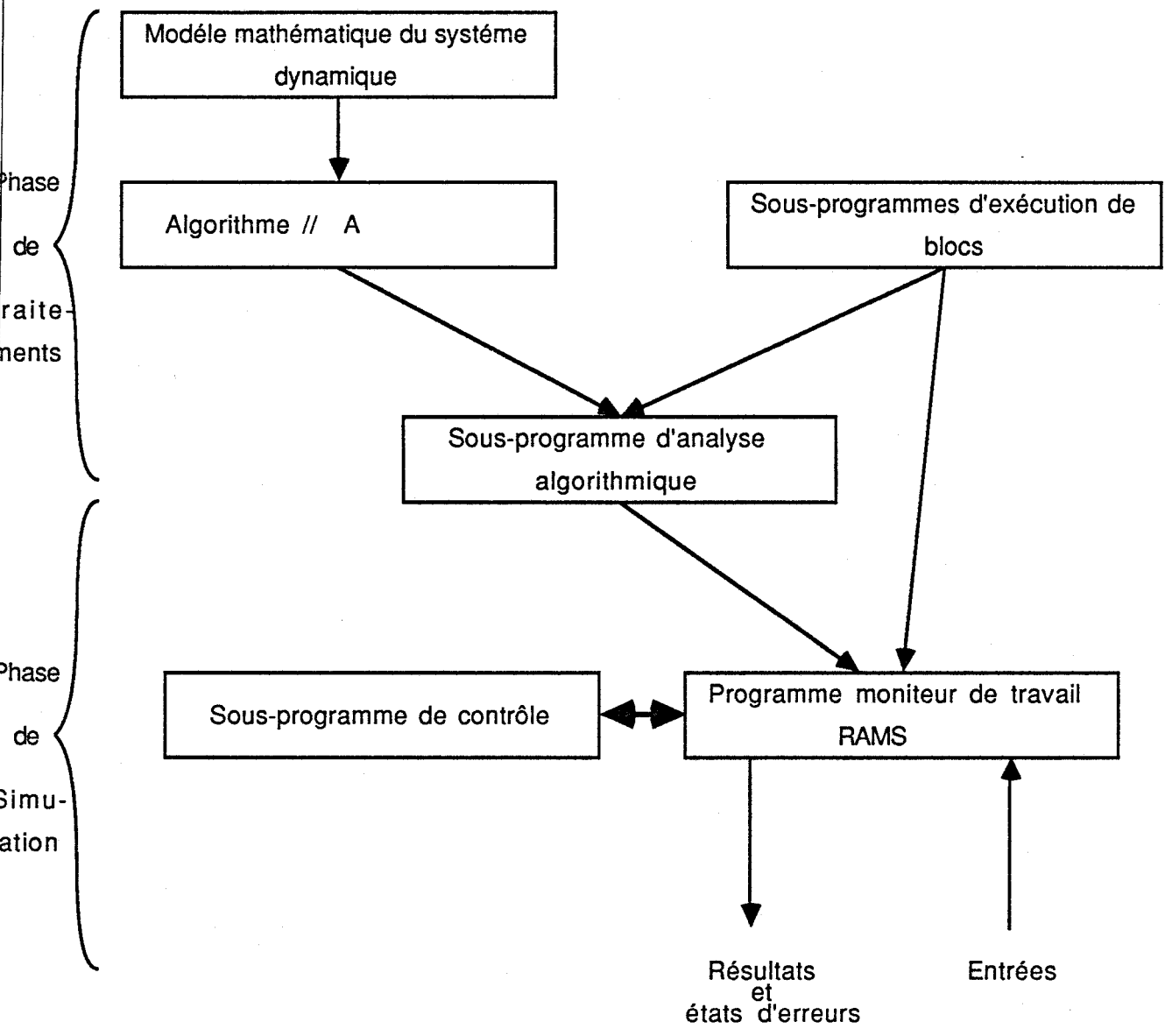


Figure N° 1

Tous ces logiciels présentent un avantage du point de vue didactique car ce sont des simulateurs temps réel et interactifs ; cependant, hormis les caractéristiques particulières du logiciel multiprocesseurs RAMS et celles relatives à l'environnement matériel du langage de simulation SHD, la grande lacune commune à tous les autres est, d'une part, le manque de la notion de multitâches et, d'autre part, l'inexistence d'un parallélisme.

En effet, nombreuses sont les applications industrielles qui, de nos jours, nécessitent la saisie et une gestion simultanée de plusieurs tâches élémentaires composantes d'un seul et même processus descriptif d'un système de production.

C'est cette lacune qui a suscité dans le cadre de notre travail l'étude et le développement d'un simulateur hybride plus performant à partir d'un logiciel existant.

Aussi dans cet esprit, nous nous proposons d'exposer ce travail selon une présentation en quatre parties.

Dans le premier chapitre, nous procédons à la présentation du logiciel de base existant TUTSIM et son environnement matériel.

Afin de réaliser un logiciel de simulation hybride multitâches, le second chapitre précise les modifications (basées sur la méthode de vectorisation) qu'il nous a fallu apporter au logiciel TUTSIM.

Le troisième chapitre est consacré aux extnesions logicielles permettant la gestion des diverses tâches, à savoir, le passage automatique des paramètres de réglage, la mise en condition initiale, l'utilisation des résultats d'une tâche par une autre selon le principe de producteur-consommateur.

La dernière partie de ce mémoire présente des exemples d'applications illustrant les extensions que nous avons apportées sur ce simulateur hybride multitâches.

CHAPITRE I

PRESENTATION DU LOGICIEL DE BASE TUTSIM

INTRODUCTION

Le simulateur TUTSIM développé sur un micro-ordinateur de type WANG-PC par la Société hollandaise MEERMAN AUTOMASERING est un outil de simulation des systèmes dynamiques dont les modèles mathématiques sont généralement décrits par un ou plusieurs systèmes d'équations différentielles linéaires ou non, de type continu ; il est écrit en plusieurs versions dont la version FORTRAN 4 modifiée en FORTRAN 77 sert de version de base pour notre travail. La structure de son programme moniteur permet à l'utilisateur deux modes de dialogue :

- l'un externe, relatif à la configuration du matériel de développement,

- le second, interne, mettant en relief l'interactivité et la structure temps réel du logiciel.

Avant d'aborder la description complète de ce simulateur, il convient de faire une brève présentation de l'environnement périphérique de base afin de préciser, dans le chapitre suivant, les modifications et adaptations nécessaires qui ont été apportées.

I. ENVIRONNEMENT MATERIEL

Les micro-calculateurs numériques connaissant à l'heure actuelle un développement très important dans le domaine de la commande automatique des processus industriels, il est évident, pour des raisons d'ordre économique et commercial, de mettre en oeuvre des outils qui permettent leurs utilisations pour diverses applications, ce qui explique le choix d'un matériel simple d'utilisation pour le développement du TUTSIM. Il se compose :

- d'un micro-ordinateur WANG-PC sous système d'exploitation CP/M,

- d'un terminal graphique de type TEKTRONIX 4010 pour les sorties graphiques des résultats de la simulation,

- d'une console de visualisation pour le dialogue entre l'utilisateur et son écran ; elle peut être aussi utilisée pour la comparaison des résultats numériques par rapport à l'évolution des courbes sur le terminal graphique,

- d'une imprimante à liaison série,

- d'une table traçante pour la copie graphique de l'écran.

L'architecture d'une telle configuration est illustrée par la figure 2.

REMARQUE : Il est aussi possible d'utiliser une seule console de visualisation interfacée d'une carte graphique par liaison série RS 232 et permettant une résolution de 300 x 256 pixels pour les sorties graphiques des résultats de la simulation. Le choix de ce type de matériel n'est pas exclusif, compte tenu de la prolifération des micro-calculateurs sous système CP/M ; aussi, comme l'exécutable de TUTSIM occupe un espace mémoire de 64 KO, toute configuration de matériel compatible disposant d'un compilateur/éditeur de liens Fortran Microsoft et dont la capacité mémoire RAM est supérieur ou égale à 64 KO, suffit pour l'utilisation de TUTSIM.

II. ENVIRONNEMENT INTERNE : LE LOGICIEL TUTSIM

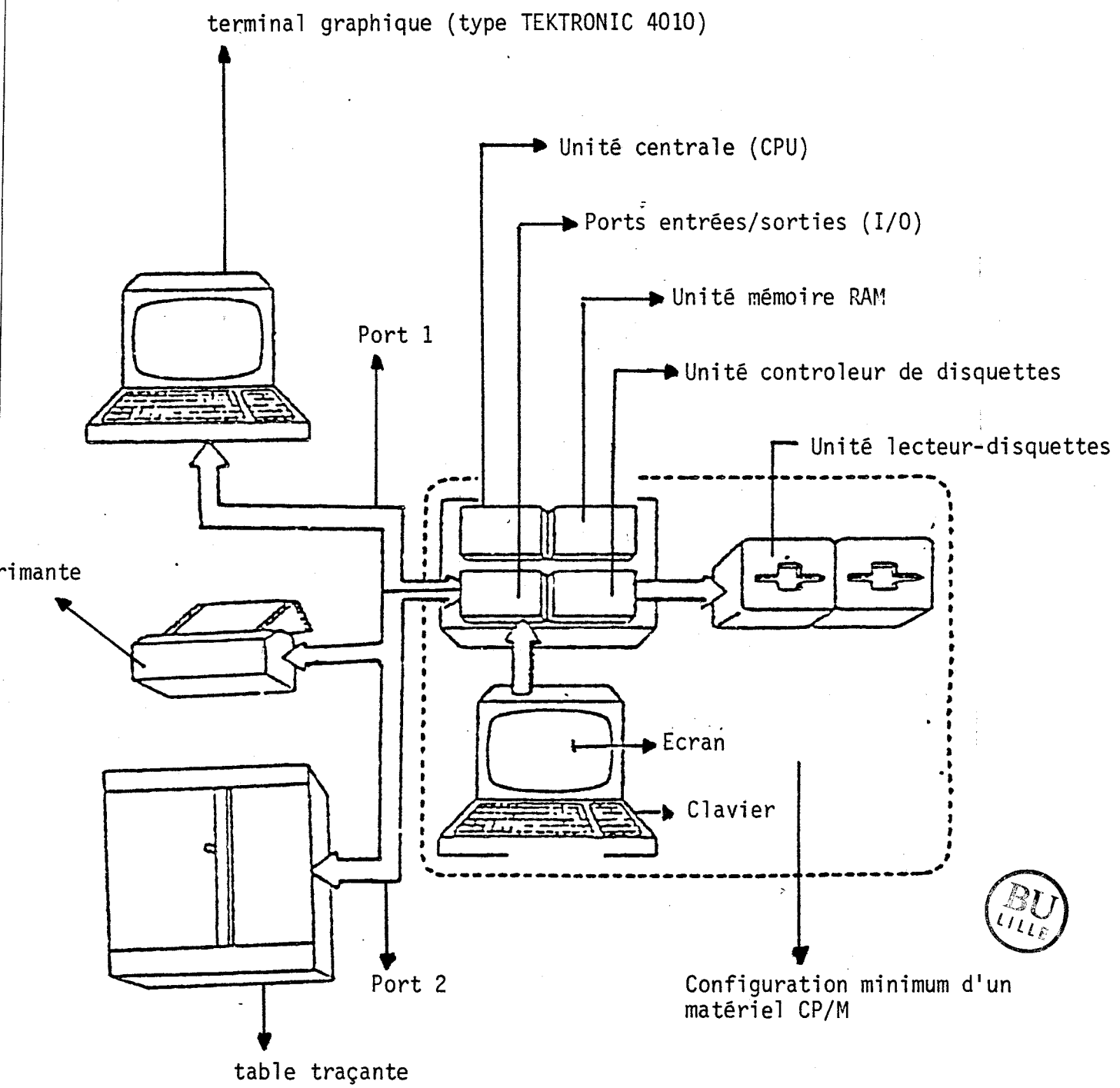


FIGURE 2

Le corps du logiciel est composé de quatre parties ou modules interdépendants (TUTSI 1, TUTSI 2, TUTSI 3, TUTSI 4) que nous détaillons brièvement comme suit :

- le module TUTSI 1 constitue le programme moniteur et comprend 36 sous-programmes qui permettent l'ordonnancement et la gestion d'une tâche sous TUTSIM, le dialogue avec le périphérique externe et enfin l'interactivité dans l'édition et l'exécution de cette tâche,

- le module TUTSI 2 comprenant 13 sous-programmes permet la création de listing du modèle de processus simulé,

- le module TUTSI 3 également composé de 13 sous programmes, est essentiellement un fichier de gestion des entrées-sorties, tant au niveau des périphériques externes que pour les sauvegardes de nouveaux fichiers créés par TUTSI 2,

- le module TUTSI 4, écrit en macro-assembleur, comprend 2 sous-programmes dont l'un sert à la gestion des interruptions issues du clavier au cours de la simulation, et le second, au retour du mode graphique du terminal en mode alpha-numérique grâce aux deux touches retour-chariot (RC) et le line-feed (LF).

Le concept de simulation s'appuyant sur l'analyse des processus linéaires à commande non linéaire dans lesquels l'information est transmise sous forme d'échantillons pris à des instants déterminés, il est bien évident que cette analyse, comme nous l'avons signalé dans l'introduction, peut se faire à partir de différentes relations mathématiques qui modélisent le processus simulé ; aussi, pour être adaptées à la méthode de simulation sous TUTSIM, chacune de ces relations mathématiques est traduite et mise sous forme de diagramme de blocs orientés ; à chaque bloc est associée une ou plusieurs expressions mathématiques qui lient les variables du bloc d'entrée à la variable du bloc de sortie. De ce fait, la représentation mathématique d'un modèle du processus étudié se traduit par un réseau de blocs numérotés de 1 à n (n entier > 0) ; le nombre de blocs représentatifs du réseau ne dépend que de la capacité mémoire du calculateur.

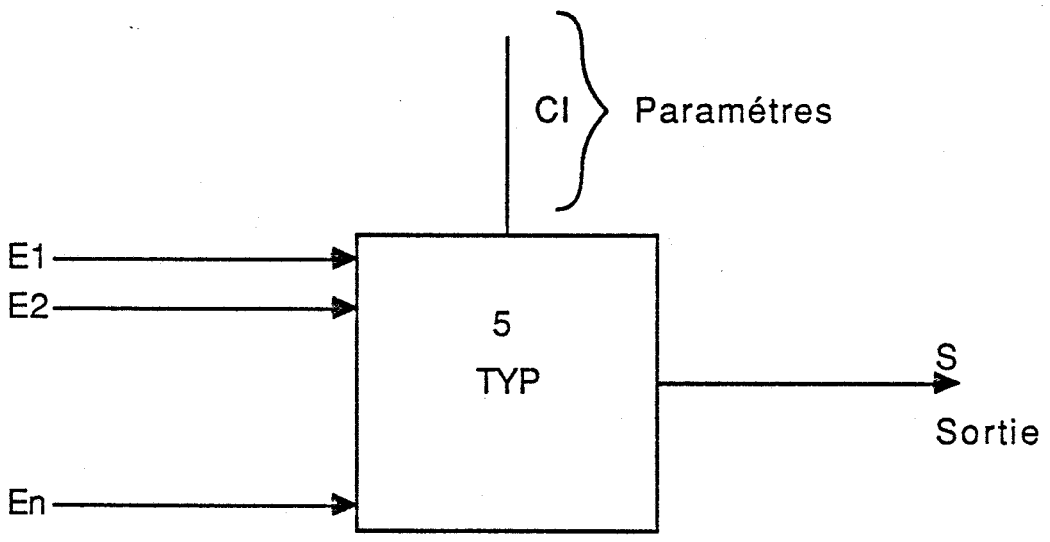
Un programme interpréteur de blocs et d'édition de la configuration du

réseau permet de décomposer en quatre étapes l'entrée du réseau sous TUTSIM ; il est à noter que l'interpréteur de blocs est équivalent à un "check-list" (liste de contrôle) de blocs prédéfinis établissant ainsi une correspondance entre la nature du bloc et son algorithme de calcul, d'où la terminologie "fonctions blocs".

Les quatre phases de la procédure d'entrée du réseau représentatif du modèle sont prises en compte dans un ordre précis dont nous donnons les caractéristiques essentielles ci-après :

PHASE 1 : STRUCTURE DU MODELE

Elle est la plus déterminante et représentative du processus étudié, car elle reproduit très fidèlement le modèle du système physique à travers l'édition numérique complète du réseau. Pour illustrer cette phase, prenons à titre d'exemple un modèle de processus dont le réseau est représenté par une seule fonction bloc : le bloc de nature indéfinie TYP.



TYP = abréviation en trois lettres majuscules désignant la nature de la fonction du bloc,

E1, E2, ..., En représentent les entrées du TYP et les sorties des blocs numérotés 1, 2, 3, ..., n (n entier 0 ou 0),

d'où la structure de liste suivante définie sur un exemple :

5, TYP, -2, 3 où

5 est le numéro du bloc TYP ayant comme entrée respective la somme des sorties négative issues du bloc numéro 2 et positive du bloc numéro 3.

PHASE 2 : PARAMETRES DU MODELE

Elle est équivalente à la phase de réglage et correction du processus, se traduisant de ce fait par la prise en compte des facteurs multiplicatifs, des valeurs initiales et des valeurs de contraintes limitatives relatives à certaines variables particulières. L'entrée des paramètres de réglage se présente comme suit :

5, 0.5, etc... où

0.5 est le premier paramètre du bloc numéro 5.

PHASE 3 : BLOCS DE SORTIE DU MODELE

Les variables qui traduisent le comportement du système et par conséquent le résultat de la simulation, sont décrites par un maximum de cinq blocs dont les images X0, Y1, Y2, Y3, Y4 représentent les blocs de sortie du modèle ; il est à observer que le bloc numéroté 0 et d'image X est un bloc particulier puisque simulant la fonction temps, il est imposé par le logiciel et permet ainsi de définir de façon implicite la durée de la simulation. Les quatre autres blocs restants dont les numéros de blocs sont libres de choix et d'images Y1, Y2, Y3, Y4 permettent à l'utilisateur d'obtenir les informations nécessaires sur les variables descriptives des états instantanés du processus (échelles de visuali-

sation). L'illustration de cette phase est donnée par l'exemple suivant :

Y1 : 5, 0, 20 où

5 représente le numéro du bloc dont la sortie représente la première information souhaitée et observable dans un cadre d'échelle allant de 0 à 20 suivant l'axe Y.

PHASE 4 : DUREE D'OBSERVATION DU PROCESSUS

Elle permet de définir de façon explicite les instants d'échantillonnage ou encore le pas de calcul ΔT et la durée totale T de simulation du processus.

REMARQUE : Pour illustrer de manière globale les quatre phases étudiées, nous traiterons plus loin un exemple complet portant sur l'étude d'un système de suspension d'automobile dont nous observerons l'évolution de la vitesse et position verticales de la roue pendant le mouvement de rebond lors de la montée sur un trottoir.

Toute la procédure de simulation sous TUTSIM peut se résumer au schéma de la figure 3 .

III. DOMAINE D'APPLICATION

Les objectifs visés dans la réalisation de TUTSIM concernent la mise au point d'un outil d'aide à l'enseignement. Dans ce sens, TUTSIM est un outil de valisation et de test de modèles ou de commandes simples d'automatismes. Il est donc nécessaire que ce logiciel puisse permettre de couvrir un vaste champ d'application ; aussi, il s'adapte assez facilement à l'étude de nombreuses applications dont nous énumérons les plus courantes sur le marché des industries et de la recherche :

- les circuits et systèmes électroniques,
- les processus thermodynamiques,
- la dynamique des fluides,
- les systèmes électromécaniques,

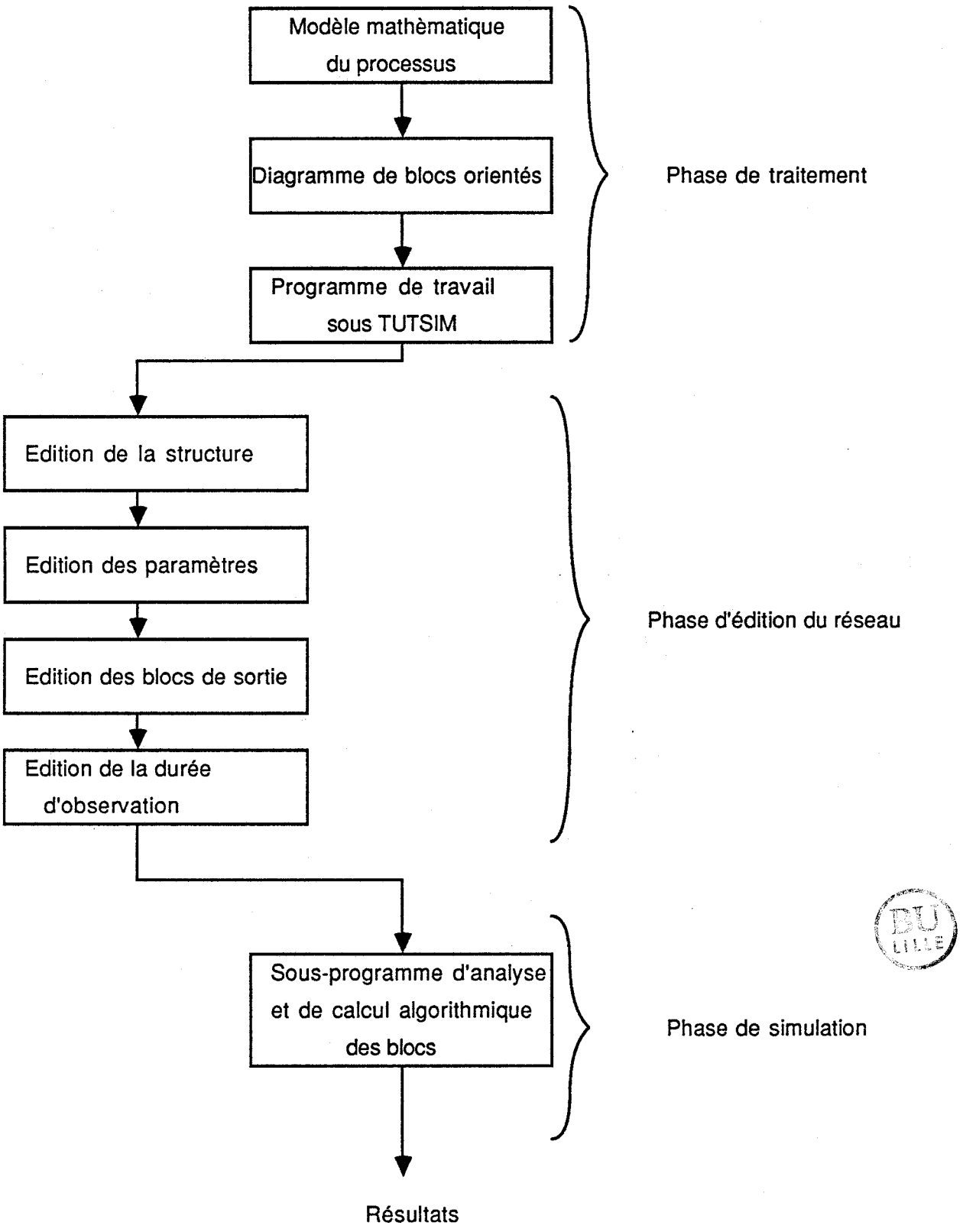


Figure N° 3

- les processus de réactions chimiques,
- les processus biologiques et physiologiques,
- et d'autres encore comme les processus économétriques ou l'économie statistique.

Bien que le nombre de blocs nécessaires à la simulation d'un processus ne dépende que de la taille mémoire du calculateur, il existe cependant certaines limites, voire contraintes, inhérentes à la fois aux conditions de précisions et d'optimalisation des résultats de la simulation et aux caractéristiques de réalisation de TUTSIM ; la version actuelle du simulateur ne permet qu'un réseau représentatif limité à 999 blocs.

IV. EXEMPLE D'APPLICATION

Afin de préciser davantage la procédure de travail sous TUTSIM, nous proposons d'illustrer par un exemple simple le "cahier des charges" de sa mise en oeuvre.

IV.1 ANALYSE ET DONNEES DU PROBLEME

Nous proposons l'étude d'un système de suspension avant et arrière d'une automobile en nous intéressant seulement aux mouvements de rebond de la roue lors de la montée sur un trottoir. Pour simplifier le système physique, nous appliquons à la limite de la hauteur de la bordure du trottoir une force constante ascendante sur la roue et supposerons qu'à l'instant initial t0 du démarrage de la simulation, le véhicule n'a pas encore amorcé son mouvement ascendant.

La figure 4 donne une illustration du modèle physique de la donnée du problème.

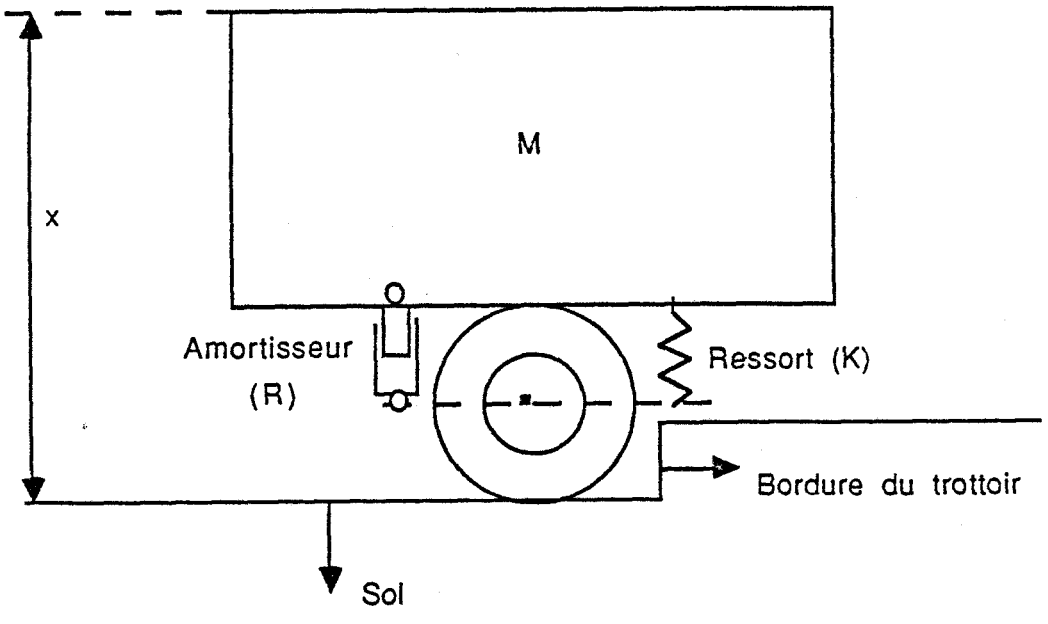


Figure N° 4

L'augmentation de l'amplitude de x est due à l'ensemble F_E des forces d'entrée exercées par le trottoir et le pneu sur le véhicule. D'où, en considérant le système véhicule-trottoir auquel nous appliquons les hypothèses ci-dessous, il est possible d'élaborer une équation aux petites oscillations de la roue en guise de modèle mathématique du système.

HYPOTHESES :

- le mouvement en x n'est pas amorcé au début de l'expérience,

- la masse de la roue est supposée négligeable,

- x représente la position verticale du corps du véhicule, repérée par rapport à sa position d'équilibre statique,

- M la masse du véhicule,

- C l'inverse de la constance de raideur du ressort,

- R la résistance de l'amortisseur.

Nous étudierons le mouvement vertical du véhicule selon l'axe x, le solide supposé immobile et en équilibre suivant le même axe.

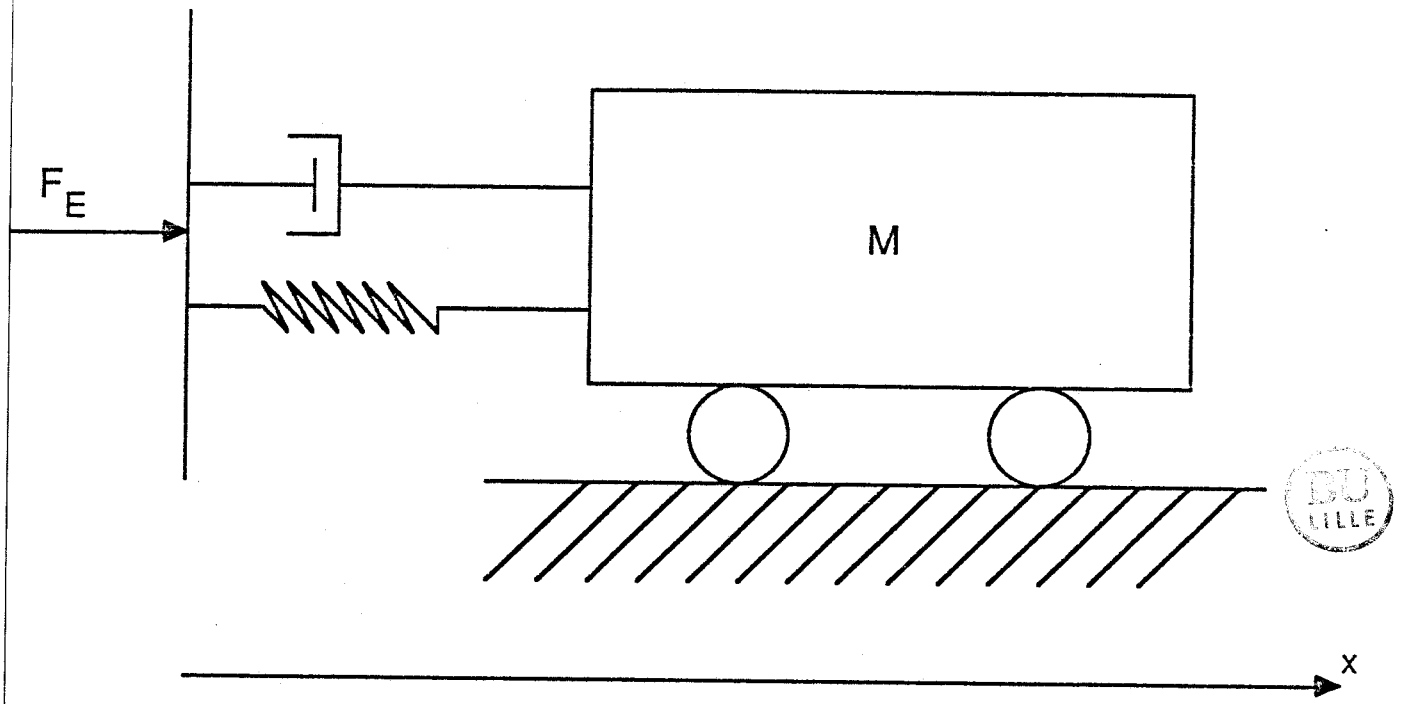


Figure N° 5

L'équation de la dynamique appliquée au solide M fournissant un modèle de son mouvement selon cet axe est décrit par l'équation suivante :

$$(4)\quad M\frac{d^2x}{dt^2} = F_E - R\frac{dx}{dt} - \frac{1}{C}x$$

Elle caractérise le modèle mathématique du processus ; aussi, à partir de la donnée des conditions initiales $\dot{x}(0)$, $x(0)$, relatives respectivement à l'accélération et à la vitesse, de la donnée des valeurs de M, C, R et de celle de l'intensité de la force F_E réduite à une impulsion temporelle, il est possible de résoudre cette équation caractéristique du modèle en élaborant simultanément les échantillons $x_i(t)$, $\dot{x}_i(t)$, images instantanées respectives de la position verticale du corps du véhicule et de sa vitesse verticale.

IV.2 SCHEMA ANALOGIQUE DU PROCESSUS ET SA REPRESENTATION PAR LA METHODE DE RESEAU DE BLOCS ORIENTES

Si nous appliquons la méthode générale de programmation des calculateurs analogiques à l'équation différentielle linéaire non homogène de la relation en supposant qu'il existe un générateur qui délivre un signal f(t) proportionnel à la force d'entrée F_E, il vient le schéma analogique de la figure 6.

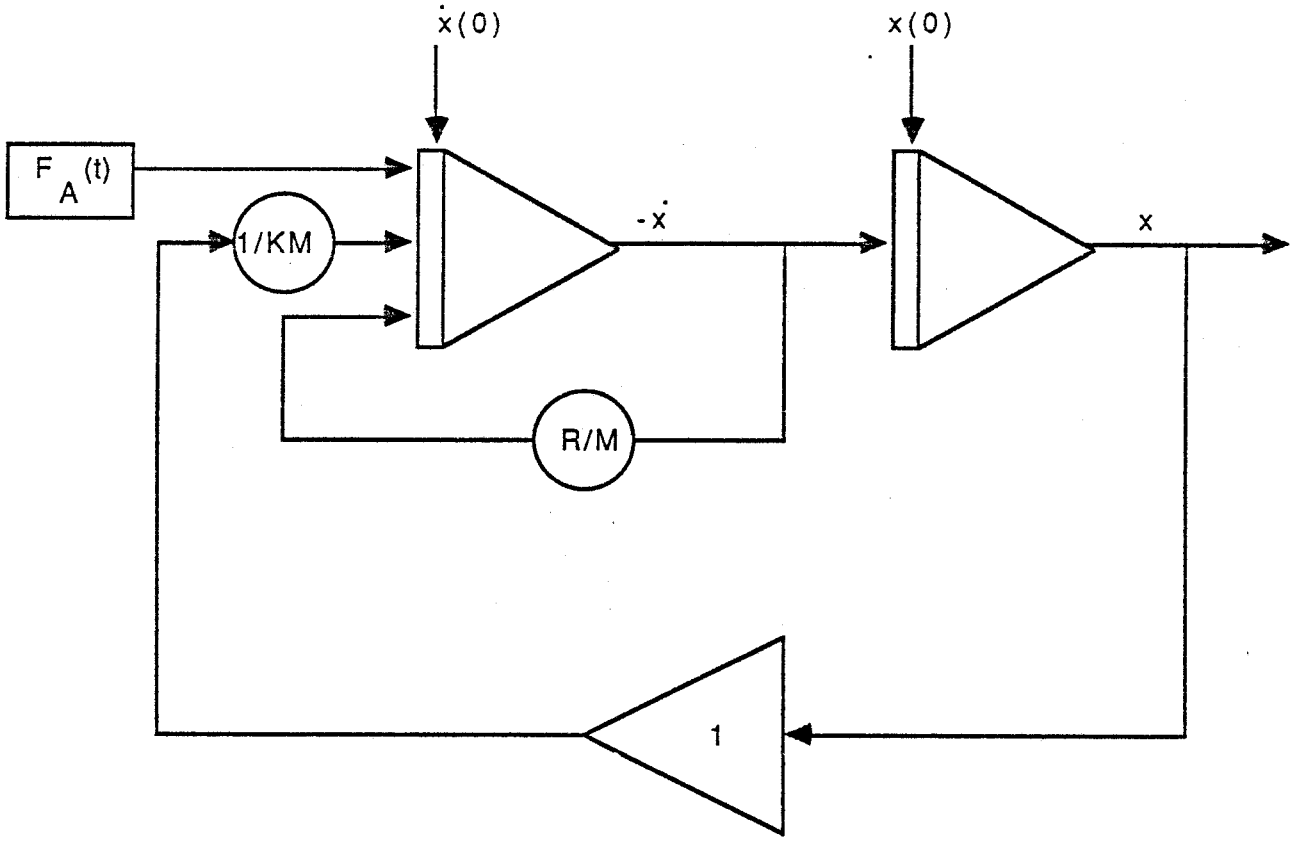


Figure N° 6

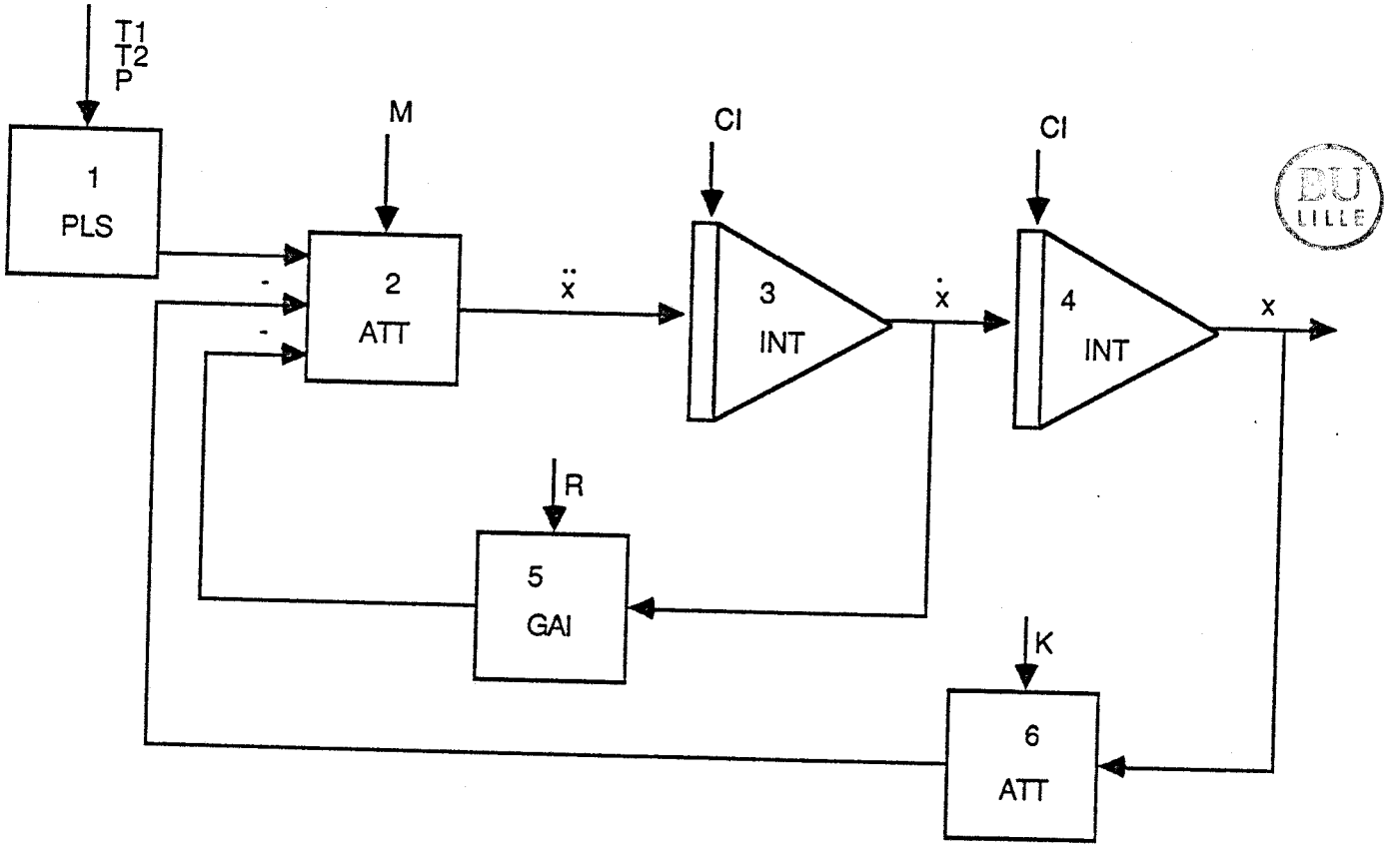
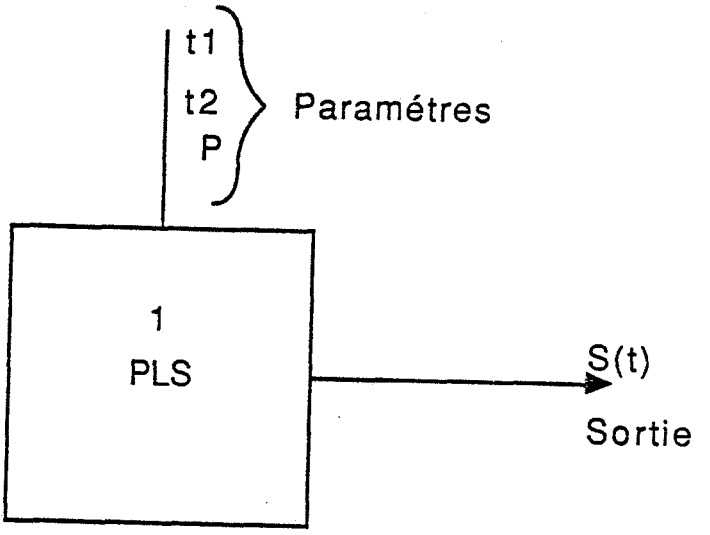


Figure N° 7

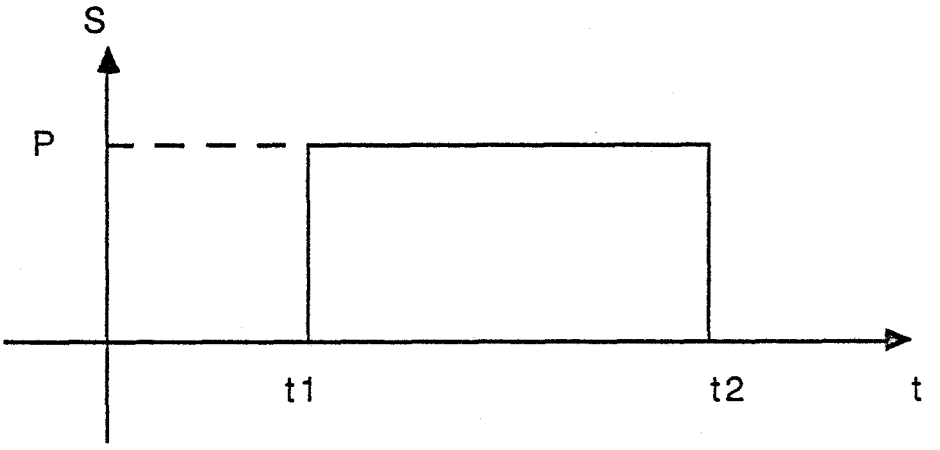
IV.3 DESCRIPTION DU RESEAU

La numérotation des blocs du réseau est arbitraire ; aussi le bloc numéroté 1 représentant la force d'entrée F_E est un bloc impulsion (PLS) dont la valeur de l'amplitude est P et la durée l'intervalle fermée $|t_1,t_2|$.



S(t) est une fonction telle que :

$$S(t) = \begin{cases} P \text{ si } T_1 \leqslant t < T_2 \\ 0 \text{ si } t > T_2 \end{cases}$$



A partir de cette fonction mathématique, on élabore un algorithme de calcul du bloc grâce à la définition de deux tableaux de vecteurs, l'un réel RBLK(P) et le second entier IBLK (n) où P et n sont deux pointeurs donnant respectivement la dimendion de ces tableaux. La structure du programme de calcul est donnée par l'organigramme suivant :

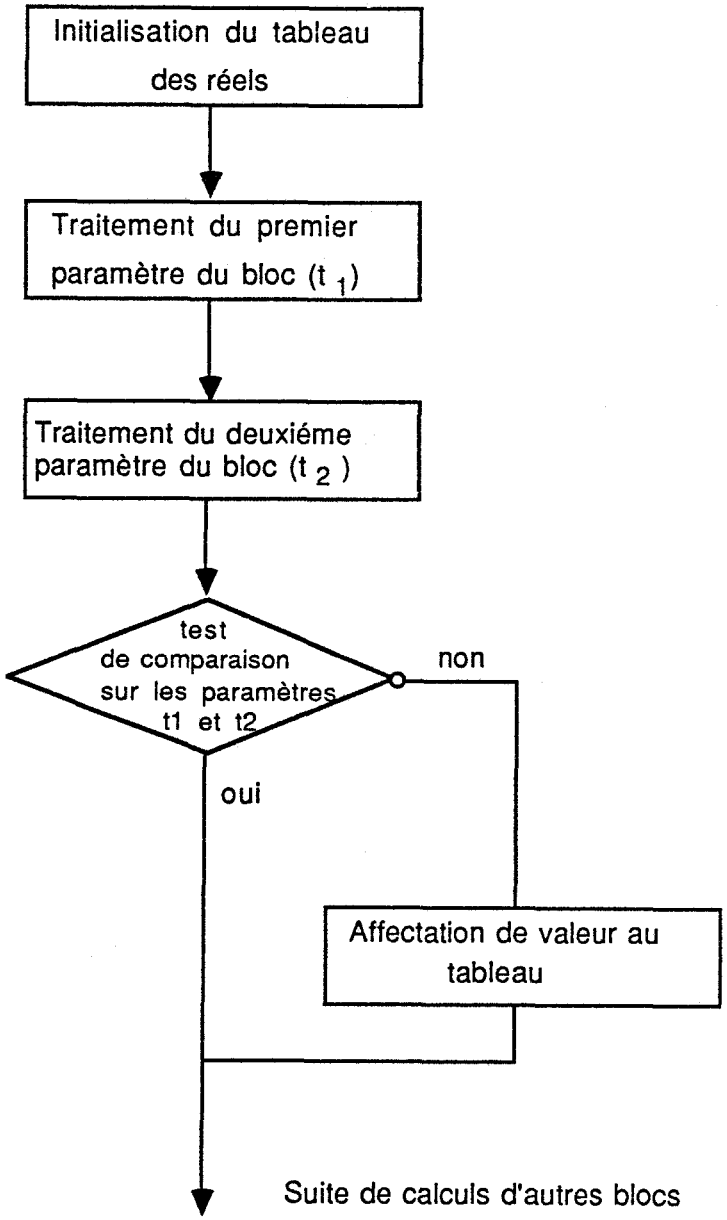
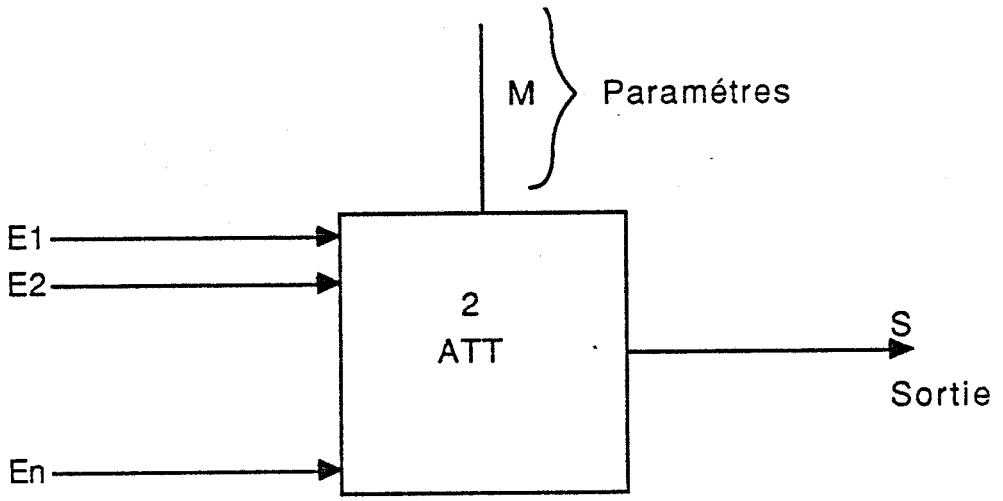


Figure N° 8

— les blocs numérotés 2 et 6 sont deux atténuateurs de facteurs 1/M et 1/K et dont la description générale se présente comme suit :



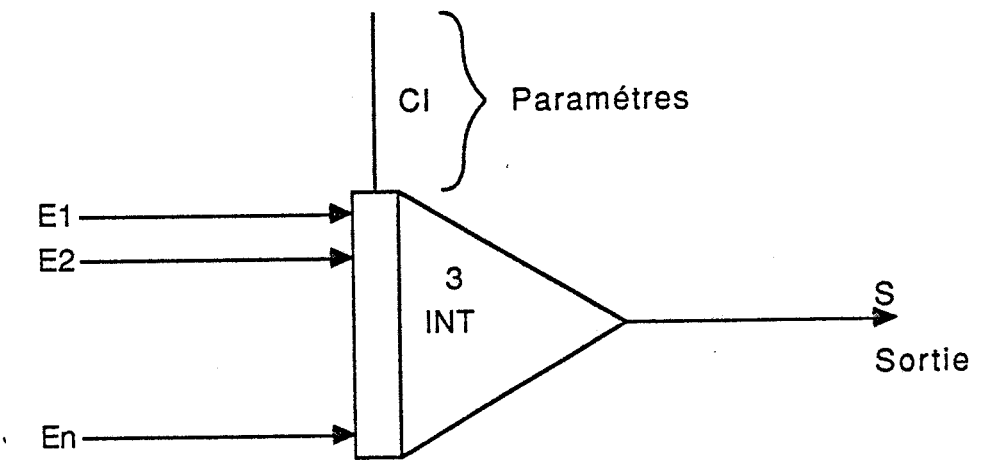
La définition mathématique du bloc est donnée par la fonction

$S = f(M,E_i)$ telle que

$$S = (1/M) \cdot \left(\sum_{i=1}^{n} E_i \right)$$

La structure du programme de calcul de cette fonction intègre deux algorithmes dont un algorithme de programmation de la fonction comme $\sum_{i=1}^{n} E_1$ relative à la prise en compte de tous les blocs d'entrée et le second relatif à la fonction d'affectation de paramètre définie précédemment.

— les blocs numérotés 3 et 4 sont deux intégrateurs de description mais dont les fonctions intégrales sont différentes selon la forme du signal à intégrer.



La sortie S représente l'intégrale par rapport au temps de la somme

des entrées E_n.

Les instants Δt de prise d'informations auxquels correspond la période d'échantillonnage sont directement liés à la forme du signal ; aussi, pour l'obtention d'une plus grande précision dans les approximations, le logiciel dispose de deux fonctions mathématiques pour l'intégration :

- l'une en accord avec l'intégrale d'Euler pour les signaux à fortes pentes en forme de dents de scie (S_1),

- la seconde en accord avec l'intégrale d'Adam Bashforth pour les signaux à faible taux d'ondulation ou à ondulation nulle (S_2).

$$S_1(0) = CI \quad \text{(Condition Initiale)}$$

$$S_1(t) = S_1(t - \Delta t) + \Delta t \cdot \sum_{i=1}^{n} E_i(t - \Delta t)$$

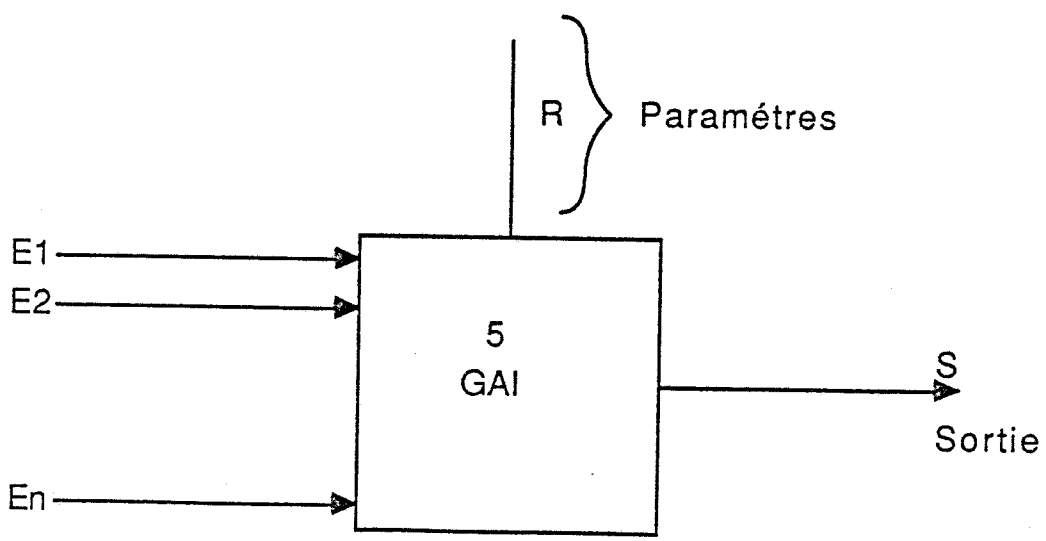
$$S_2(0) = CI$$

$$S_2(0) = S_2(0) + \Delta t \cdot \sum_{i=1}^{n} E_i(0)$$

$$S_2(t) = S_2(t - \Delta t) + 0{,}5\Delta t \left| 3 \sum_{i=1}^{n} E_i(t - \Delta t) - \sum_{i=1}^{n} E_i(t - 2\Delta t) \right|$$

La condition initiale (CI) est égale à 0 par défaut.

— le bloc numéroté 5 est un bloc de gain de facteur R et dont la description générale et par suite les algorithmes de calcul sont semblables aux blocs atténuateurs.



Sa fonction mathématique est donnée par la relation :

$$S = R \cdot \sum_{i=1}^{n} E_i$$

Il convient de faire quelques remarques importantes relatives à la description des blocs du réseau.

REMARQUES :

* Comme le bloc 1, il existe deux autres blocs dont la structure diffère des autres par le fait qu'ils n'ont pas d'entrée ; il s'agit du bloc représentatif de l'information temps (TIM) et du bloc descriptif de l'information relative au bruit pseudo-aléatoire (bloc NOI).

* Tous les autres types de blocs, selon la version actuelle du logiciel peuvent avoir de 10 à 40 blocs comme entrées et un seul [illegible] comme sortie.

* Certains blocs n'ont pas de paramètres comme, par exemple, les blocs TIM, NOI ou les blocs qui réalisent certaines opérations arithmétiques et logiques. Pour ces blocs et ceux dont les paramètres sont nécessaires, mais ignorés lors d'une application, la valeur par défaut des paramètres est dans ces cas égale à 1.

IV.4 INTERPRETATION DU RESEAU

Pour simuler le modèle mathématique du processus grâce au réseau de blocs de la figure 7 , le logiciel dispose de quatre sous-programmes interpréteurs qui permettent l'édition du réseau conformément aux quatre phases définies dans le paragraphe I et qui formalisent le modèle suivant le principe de simulation sous TUTSIM. C'est dans cet esprit que, pour l'exemple choisi, nous proposons les valeurs numériques suivantes qui permettront d'écrire les composantes de chaque phase.

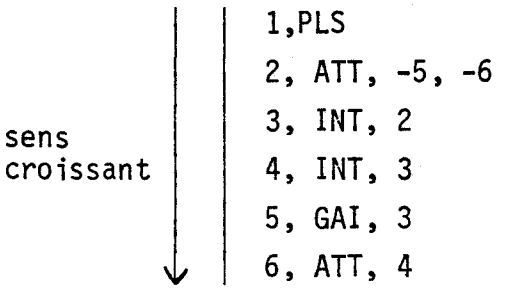
M = 10kg
K = 10m/N

R = 0,5 NS/m

$$F_E = \begin{cases} 0 \text{ si } t < 10s \\ 1 \text{ si } t \geq 10s \end{cases}$$

IV.4.1. PHASES D'EDITION DU RESEAU

IV.4.1.1. Edition de la structure



L'écriture de la structure du réseau présente deux niveaux hiérarchiques :

- le premier niveau de hiérarchie verticale se traduit par l'écriture des numéros des blocs descriptifs du réseau ; ces numéros pouvant être dans un ordre quelconque, bien que nous ayons choisi, pour notre exemple, un ordre croissant en descendant,

- le second, d'hiérarchie horizontale dont chaque ligne est décrite de la façon suivante :

le premier caractère de type chiffre représente le numéro du bloc suivi de son type, suivi lui-même d'un ou plusieurs nombres entiers positifs ou négatifs qui représentent les numéros des blocs d'entrée relatifs au bloc considéré.

Dans l'exemple ci-dessus, la seconde ligne de la structure traduit le fait que le bloc numéroté 2 est un atténuateur dont les entrées sont les sorties des blocs numérotés 1, 5 et 6.

IV.4.1.2 Edition des paramètres de réglage

```
1, 10, 200, 1
2, 10
5, 0,5
6, 10
```

Elle présente les mêmes niveaux hiérarchiques, mais avec une variante dans la hiérarchie horizontale ; le premier chiffre représentant le numéro de bloc d'une ligne n'est suivi que d'un ou plusieurs nombres réels représentant les paramètres de réglage du processus ; il est à noter que l'ordre d'écriture de ces paramètres est conforme à la définition du bloc telle qu'elle a été traitée dans le paragraphe IV.3.

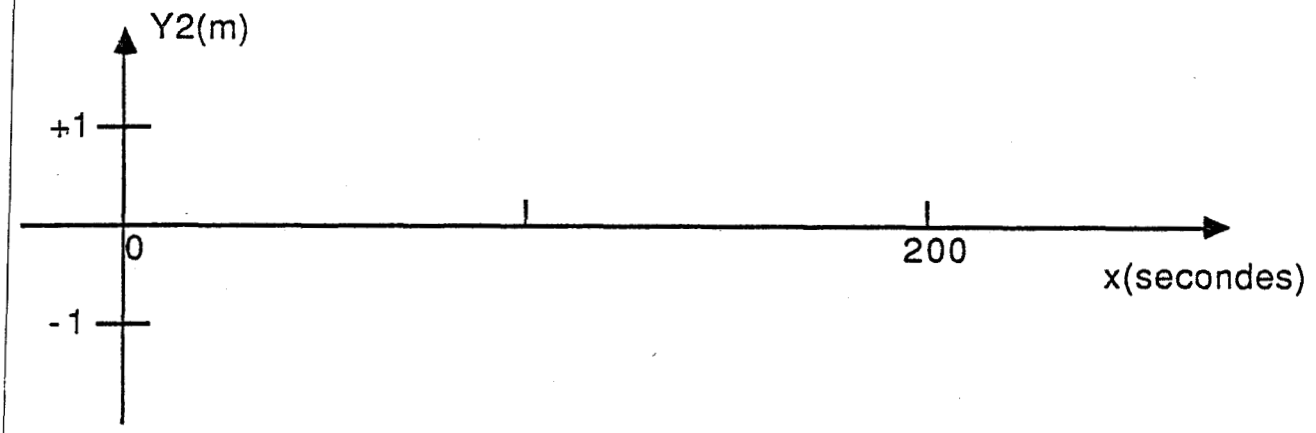
IV.4.1.3 Edition des blocs descriptifs des informations ou variables de sortie

```
 X : 0, 0, 200
Y1 : 1, -1, 9
Y2 : 3, -1, 1
Y3 : 4, 0, 20
```

On retrouve les deux niveaux évoqués précédemment avec une restriction relative à la hiérarchie verticale pour ce qui concerne l'ordre dans l'édition des variables génératrices des informations souhaitées sur le processus ; aussi cette phase commence toujours par l'édition de la variable X qui représente implicitement l'abscisse temporelle, suivie de celle des variables Y1, Y2, Y3 descriptives de l'axe des ordonnées et qui permettent d'obtenir simultanément trois informations instantanées relatives respectivement à la force impusionnelle d'entrée F_E, à la vitesse des oscillations du mouvement de l'ossature du véhicule et enfin à sa position.

Pour ce qui concerne la hiérarchie horizontale, outre la particularité de la première ligne évoquée plus haut, les autres lignes commencent toujours par la première information souhaitée (Y1), suivie du numéro de bloc dont la sortie définit la nature de l'information et enfin de deux nombres réels descriptifs de l'échelle de mesure de cette information.

En nous référant à l'exemple ci-dessus, la troisième ligne traduit le fait que la sortie du bloc numéroté 3 et dont l'image est donnée par l'axe Y2, représente la dérivée première de la variable position x, donnant ainsi une information sur la vitesse verticale du mouvement oscillatoire de la caisse du véhicule dans un intervalle de mesure allant de -1 à +1m/s.



IV.4.1.4. Edition de la durée d'observation des informations

0.2, 200

La durée de simulation est directement liée au temps référencé par le bloc prédéfini X de numéro 0. Un sous-programme d'intégration de ce temps suivant la valeur du pas de calcul (Δt) équivalent à la période d'échantillonnage permet d'obtenir une information sur la durée totale (T), d'obtenir du processus le rapport $T/\Delta t$ de ces deux variables. Il définit le nombre de pas de calcul (NP) effectué ; il est à souligner que cette grandeur est un facteur déterminant pour la précision des résultats numériques du processus.

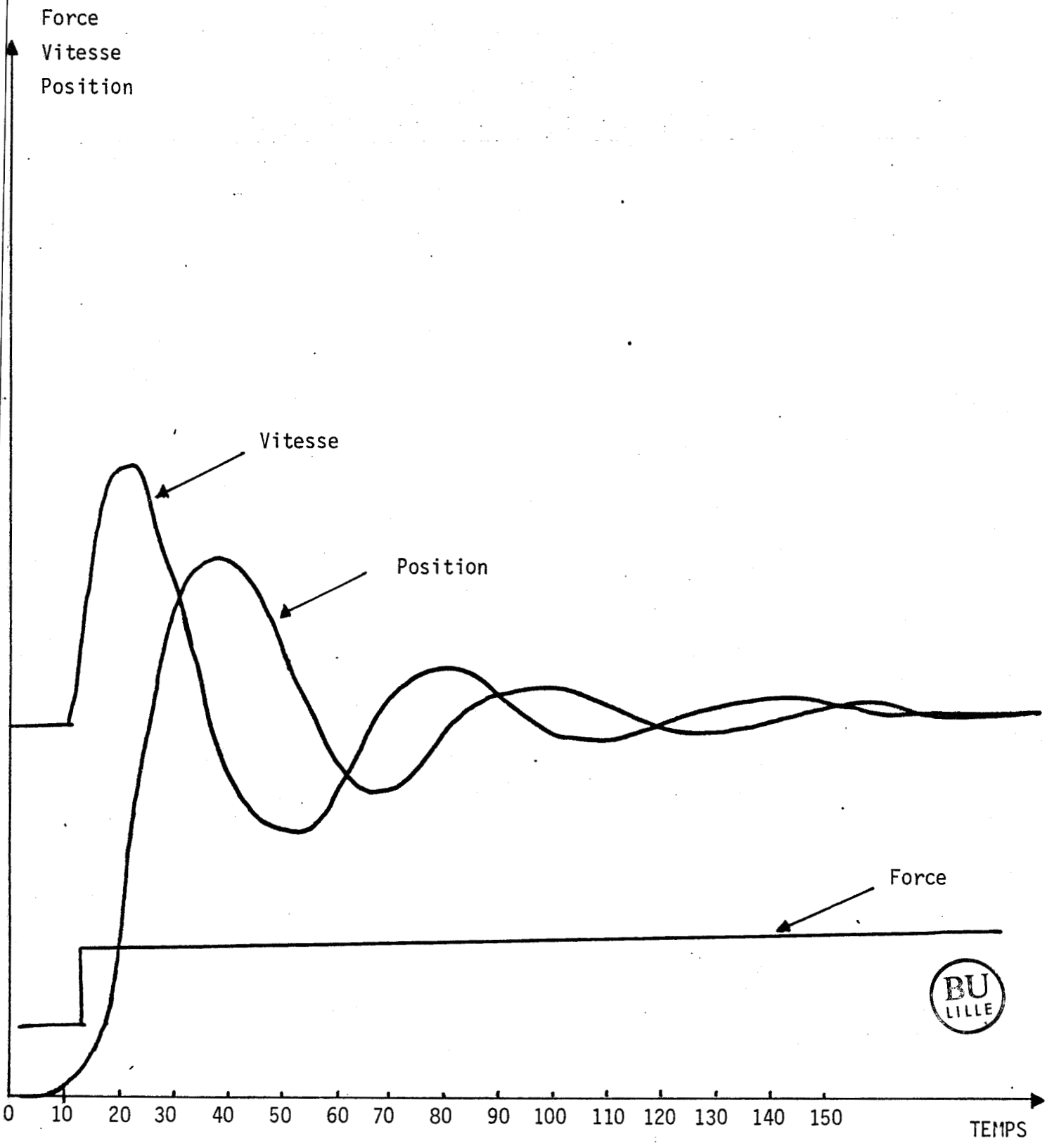


FIGURE 9

CONCLUSION

L'exemple traité dans ce chapitre nous a permis, dans un premier temps, de faire une présentation globale du matériel et du logiciel TUTSIM de base associé et de dégager, dans un deuxième temps, les limites et avantages inhérents à son exploitation dans le domaine des applications pour lesquelles il a été développé.

En tout premier lieu, utilisé comme simulateur temps réel de diagrammes de blocs orientés, il constitue un outil fiable et facile d'accès. En effet, pour un utilisateur non spécialisé en informatique, les séquences de la procédure de travail illustrée par le schéma de la figure est d'une relative simplicité, le dialogue interactif reste cependant d'un niveau assez primitif du fait que l'utilisateur n'est pas suffisamment guidé dans le choix des commandes pendant l'exécution d'une tâche.

Dans le domaine des portables, il offre de larges possibilités d'adaptation sur divers calculateurs de type PC, par exemple, ce qui justifie son impact didacticiel remarquable.

Au delà de ces quelques points positifs, le logiciel reste très limité en tant que simulateur monotâche. Le développement actuel d'outils de simulation se porte essentiellement sur des aspects multitâches. En effet, de nombreuses applications industrielles nécessitent généralement une coopération entre plusieurs tâches conduisant à un parallélisme dans les traitements.

Aussi, pour préciser davantage cet aspect dans la résolution des problèmes industriels, justifiant par ailleurs les limites du logiciel de base et les transformations nécessaires et indispensables que nous devons apporter pour

son amélioration dans les objectifs de simulateurs de la nouvelle génération, outre l'exemple de résolution hybride d'un problème régi par un modèle de type à "constantes réparties" décrit par une équation à deux variables indépendantes que nous développerons plus loin dans le chapitre IV, voici trois autres exemples de modèles de type industriel et qui illustrent assez bien la méthode de traitement parallèle des simulateurs multitâches :

- En premier lieu, il s'agit d'un exemple portant sur l'étude d'un système de sécherie multicylindre de machine à papier en régime permanent, /André LEMAITRE, 7/, dont le but permet d'obtenir pour chaque cylindre en tout point de la sécherie la température et l'humidité de la feuille ; la résolution de ce système détermine les deux variables caractéristiques de l'état d'humidité du papier, c'est-à-dire, le taux moyen d'évaporation ainsi que le flux de chaleur moyen pour chaque sécheur.

- Le deuxième exemple concerne l'étude d'un système de réacteur d'épitaxie en phase liquide /J.P. BABARY, 8/, dont le modèle mathématique de fonctionnement thermique du réacteur est décrit par un système d'équations aux dérivées partielles non linéaires, pouvant être simplifiées si on étudie le problème autour d'un profil désiré. Le but de cette étude est la conception et la mise en oeuvre d'un algorithme de commande numérique sous-optimale et la régulation du profil de température qui règne à l'intérieur du tube de quartz.

- Enfin, citons un dernier exemple de modèle portant sur l'étude d'une extrudeuse bi-vis en pression et températuer /J.F. LAFAY, 9/ dont l'une des applications peut être la fabrication des sacs plastiques et dont le modèle mathématique permet de concevoir un algorithme qui, en fonction des écarts de qualité, déterminera de nouveaux profils de température et de pression pour l'amélioration du produit en sortie de l'extrudeuse.

L'étude de tous ces systèmes cités est basée sur le même principe de commande : une commande multivariable à laquelle sont adaptés les outils de simulation offrant un mode de traitement parallèle, ce qui implique pour nous dans l'élaboration de la version multitâches de TUTSIM l'intégration de structures à deux niveaux de dialogues :

- un dialogue interne, implicitement orienté vers les échanges de données

entre diverses tâches élémentaires et

- un dialogue externe qui développe davantage l'aspect conversationnel du simulateur vis-à-vis de l'utilisateur.

C'est dans ce sens que nous proposons d'apporter une extension sensible au produit TUTSIM tout en conservant, par ailleurs, les caractéristiques d'une approche didactique aisée, spécificité essentielle de ce logiciel dans sa version initiale.

CHAPITRE II

NOUVELLE CONFIGURATION DE TUTSIM AUTOUR DU VAX 11/750

INTRODUCTION

La structure du moniteur de base actuellement disponible a été définie simplement de manière à réaliser, d'une part, les fonctions essentielles relatives à l'édition et au calcul de blocs et, d'autre part, à sélectionner le périphérique de visualisation des résultats.

Du point de vue utilisateur, la résolution d'un problème se limite seulement à trois points :

- le premier concerne la description du processus étudié par un modèle mathématique,

- le second point traduit ce modèle par un diagramme de blocs orientés,

- enfin, le troisième fournit les résultats de la simulation de ce diagramme grâce à des algorithmes de fonctions à blocs.

Toute modification ou réglage du processus, voire introduction d'un nouveau processus ou de nouvelles informations contraint l'utilisateur à faire une nouvelle réédition de son modèle.

Aussi, nous proposons dans ce chapitre de donner les différents moyens qui peuvent, d'une part, améliorer le caractère interactif du logiciel par introduction de nouveauxpoints d'interventions supplémentaires pour mieux guider l'utilisateur et, d'autre part, de permettre une simulation parallèle de dix tâches pour la version VAX afin de donner au logiciel une structure orientée

vers la simulation multitâches.

Pour atteindre ces objectifs, nous proposons de présenter un nouveau moniteur hybride. Ce moniteur peut être considéré comme un système d'exploitation dont les buts essentiels sont d'assurer à la fois la gestion du calculateur analogique/numérique (TUTSIM) et l'exécution des programmes hybrides.

Ceci nous conduit tout naturellement à soulever deux problèmes :

. le premier se rattache au nouvel environnement dans le domaine du matériel et du logiciel ; en effet, la mémorisation de plusieurs tâches nécessite un calculateur de capacité espace-mémoire vive plus importante et adaptée au volume des traitements pour une simulation en mode parallèle. Concernant le logiciel, le langage choisi doit pouvoir satisfaire à la fois l'orientation didactique du simulateur et en même temps que son aptitude à intégrer aisément les fonctions particulières adaptées à la résolution des problèmes de caractères scientifiques ;

. le second problème en partie dépendant du premier (structure de gestion de mémoire en mémoire partagée) souligne toutes les modifications des programmes de base et la définition de nouveaux indicateurs ou pointeurs qui ont permis de doter le simulateur des structures multitâches.

C'est dans cette optique que nous adoptons pour ce chapitre une présentation en deux parties :

- dans la première, nous proposons une brève description du nouvel environnement,

- les modifications et les procédés de gestion des tâches seront traités dans la deuxième partie.

I. CHOIX DU LOGICIEL ET MATERIEL DE DEVELOPPEMENT

Le simulateur de base dans sa version FORTRAN IV se situe déjà dans le domaine des portables ; aussi est-il, à notre point de vue, plus aisé de passer de cette version sous système d'exploitation MS-DDS à la version FORTRAN 77 sous le sous-système micro RSX.

En outre, ce langage est universel, mieux orienté vers le calcul scientifique et cependant, moins structuré que le langage Pascal. Il permet en conséquence de bénéficier de l'apport des bibliothèques scientifiques Fortran. Nous avons par ailleurs adopté une méthode de programmation structurée afin d'assurer la lisibilité et la mointenance correctes de ce logiciel.

Le souci de disposer d'un calculateur de plus grande capacité espace mémoire que celle de l'environnement matériel d'origine a guidé notre choix qui, dans un premier temps, s'est porté sur le micro PDP 11 sous système d'exploitation RSX en guise de système de développement.

Aussitôt les premières transformations, le micro PDP s'est révélé inadapté, l'exécutable ayant atteint la limite des 64 KO d'expace mémoire vive disponible. Nous avons donc opté pour le VAX 11/750 sous système d'exploitation VMS ; ce matériel dispose d'une capacité espace mémoire vive de 5 MO étendue à un espace mémoire disque de 456 MO et d'un système de gestion d'espace virtuelle.

II. MODIFICATIONS PROPOSEES DU LOGICIEL TUTSIM

II.1 INTRODUCTION

L'état d'un système (généralement représenté par un ensemble d'équations différentielles) est à tout instant (t) quelconque représenté par un ensemble de variables dites variables d'état du système. La représentation sous forme matricielle des équations différentielles descriptives du modèle mathématique du système permet de définir le nombre "n" de ces variables auxquelles correspondent le même nombre "N" d'intégrateurs nécessaires pour la description du processus par un réseau de blocs orientés. /12,15/

On observe sur la figure 7 de l'exemple traité dans le chapitre précédent qu'à l'instant initial (to), ces variables d'état se confondent aux conditions initiales du systèmes. On en déduit de toute évidence que les variables d'état représentent "l'évolution d'un système observable", c'est-à-dire qu'il est ici toujours possible de reconstituer à tout instant le vecteur état à partir d'une simple observation des sorties.

Aussi, pour un processus donné, il s'agira de coupler ces informations "mémoires" à des indicateurs ou pointeurs afin que chaque élément soit rangé dans des tableaux dont les définitions tant au niveau des dimensions qu'à celui des tailles doivent permettre le choix de telle ou telle contre stratégie dans les techniques d'adressage. Avant de décrire les points essentiels de la transformation de TUTSIM, nous proposons d'effectuer un bref rappel sur la gestion de structure des tables. /11,18/

II.2 GESTION DE TABLES

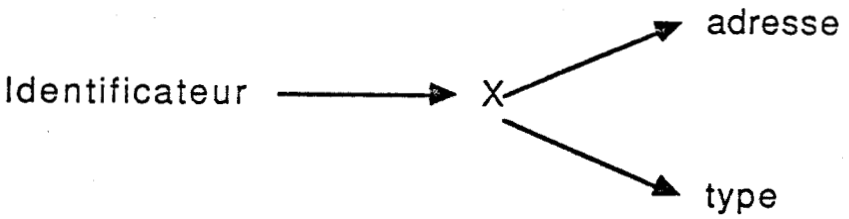
Nous définissons une table par le biais de ses fonctions d'accès, soit :

- I un ensemble dit "ensemble des indicateurs",
- E l'ensemble des emplacements de la mémoire.

Une fonction dite "fonction d'accès de table" permet, étant donné l'indicateur de I, de leur associer un emplacement E, c'est-à-dire, un enregistrement

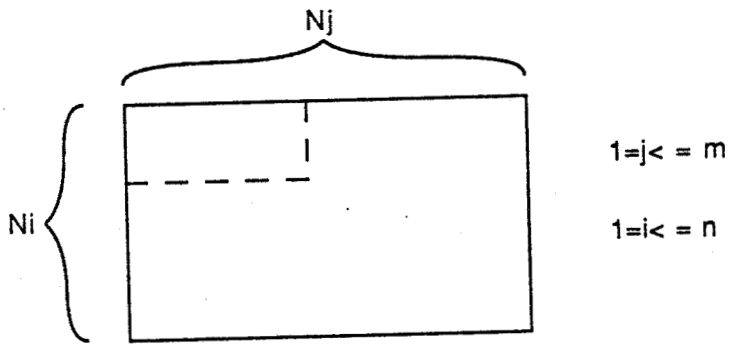
ou une information.

EXEMPLE : Considérons l'ensemble des blocs descriptifs d'un réseau quelconque, nous pourrons définir une table de blocs par la relation d'application suivante :



où x représente l'emplacement auquel on associe les informations adresse et type.

L'identificateur peut être un couple d'entiers (Ni, Nj) auquel cas il permet d'accéder à une matrice :



Quelle que soit la définition de l'identificateur (un entier ou couple d'entiers), toute de représentation de table doit comporter :

- un choix d'une méthode d'accès,

- un choix d'une méthode de modification, laquelle consiste à :

  * modifier la valeur d'un élément dans la table,

  * supprimer un élément de l'ensemble,

* ajouter un élément (c'est-à-dire, une information nouvelle correspondant à un nouvel indicatif).

La méthode d'accès, dépendante de la structure de table, permet de définir une fonction bijective entre l'indicatif et l'emplacement mémoire. C'est le principe même de la méthode d'adressage calculé pour laquelle nous avons basé toutes les transformations nécessaires pour la sairie et mémorisation de plusieurs tâches afin d'en assurer la gestion simultanée, discrète et hiérarchisée pour un traitement parallèle.

II.2.1 PRINCIPE DE L'ADRESSAGE CALCULE /19/

Selon la structure de la table, étant donné un indicatif $x \in I$, il s'agira de calculer l'adresse a(x) de l'élément ; aussi étudierons-nous deux cas :

a) cas d'un tableau à une dimensicn :
Dans cette structure, l'indicatif est un nombre entier compris entre p_1 et q_1 tel que $x \in |p_1, q_1|$.

Si nous désignons par l'adresse du premier élément du tableau et $\alpha + q_1 - p_1$ l'adresse du dernier, il vient la fonction d'adressage de type linéaire donnée par la relation :

$$a(x) = \alpha + x - p_1$$

Si chaque élément du tableau est un K-uple, alors cette fonction devient :

$a(x) = \alpha + K(x-p_1)$ où K définit la taille de l'élément.

Exemple : en Fortran, K=2 pour un élément complexe du tableau.

b) cas d'un tableau à n dimensions
C'est un tableau de n vecteurs de dimensions quelconques.

Dans la structure à n dimensions, l'indicateur x est de type n-uple, soit :

$$x = (i_1, i_2, \ldots, i_n)$$

avec $i_1 \in |p_1, q_1|$

$i_2 \in |p_2, q_2|$

$\vdots$

$i_n \in |p_n, q_n|$

Si nous supposons toujours α l'adresse du premier élément et désignons par K la taille, c'est-à-dire le nombre de mots par entrée, alors il ivent la fonction d'adressage a(x) telle que :

$$a(x) = a(i_1, i_2, \ldots, i_n) = \alpha + Kr_n(i_1, i_2, \ldots, i_n)$$

En remarquant que le tableau de dimension K est un sous-tableau de dimension k + 1, on peut aisément calculer la fonction r_n grâce à la relation de récurrence suivante :

$$r_1(i_1) = i_1 - p_1$$
$$r_2(i_1,i_2) = r_1(i_n)d_2 + i_2 - p_2 \text{ où } d_2 = q_2 - p_2 + 1$$
$$r_3(i_1,i_2,i_3) = r_2(i_1,i_2)d_3 + i_3 - p_3 \text{ où } d_3 = q_3 - p_3 + 1$$
$$\vdots$$
$$r_n(i_1,i_2,\ldots,i_n) = r_{n-1}(i_n,\ldots, i_{n-1})d_{n-1} + i_n - p_n$$

Nous pouvons conclure sur l'extrême rapidité de la méthode puisqu'il suffit de n multiplications pour calculer la fonction d'adressage $a(i_1,i_2\ldots i_n)$

II.3 APPLICATION DE LA METHODE DE GESTION DES TABLES : GENERATION DE PLUSIEURS TACHES DANS LE CONTEXTE DE SIMULATEUR MULTITACHES

La méthode d'adressage calculée hiérarchisée développée ci-dessus révèle qu'un tableau de n dimensions n'est en fait qu'un tableau à une dimension dont chaque élément constitue en lui-même un tableau à (n-1) dimensions. L'application de cette méthode présente les deux avantagesusuivants :

1° la représentation d'un tableau grâce à une série de vecteurs à une dimension, chacun de ces vecteurs contenant soit des valeurs, soit des pointeurs vers d'autres vecteurs.

2° la gestion accélérée de l'espace mémoire occupée si les tableaux sont dynamiques (de dimensions variables).

Aussi, pour faire la saisie de plusieurs tâches, nous donnons une nouvelle définition aux principaux tableaux dans lesquels sont mémorisées toutes les variables d'enregistrement des modèles, soient :

- IBLK (K,n), le tableau des variables entières,
- RBLK (K,P), le tableau des variables réelles,

où les variables K, n et P représentent respectivement le nombre des modèles de processus, la taille des variables entières et celle des variables réelles.

L'acceptation de nouvelles fonctions par le logiciel implique un élargissement de son espace mémoire tel que P soit dans le rapport n/2.

L'application de la théorie développée serait incomplète si nous nous limitions uniquement à la transformation de ces tableaux enregisteurs de variables.

Il faut pouvoir, en dehors des dimensions des tableaux, accéder aux différents types de variables effectifs qui définissent chaque modèle de processus ; aussi, nous allons associer à chacun de ces tableaux deux vecteurs en guise de pointeurs, soit :

- NVOLI (K) : vecteur pointeur donnant la taille effective des données d'un modèle dans le tableau des entiers)

- NVOLR (K) : indiquant la taille de ces mêmes données dans le tableau des réels.

Les deux tableaux étant présents dans presque toutes les parties de pro-

programmes (fonctions et sous-programmes) du simulateur, il est évident que leur nouvelle structure impose une modification de presque toutes les lignes de programmes ; ces transformations nécessitent par ailleurs la définition et une extension de nouvelles boucles d'itération à certains endroits de programmes, dans l'objectif de la saisie simultanée de plusieurs tâches en vue d'une exécution parallèle.

Pour ne pas alourdir notre exposé par le détail de tous les points de transformation, nous ne donnerons dans ce paragraphe qu'une partie simplifiée de l'organigramme du nouveau sous-programme relatif à la saisie et à l'édition séquentielles des tâches. D'où le graphe de la figure 10.

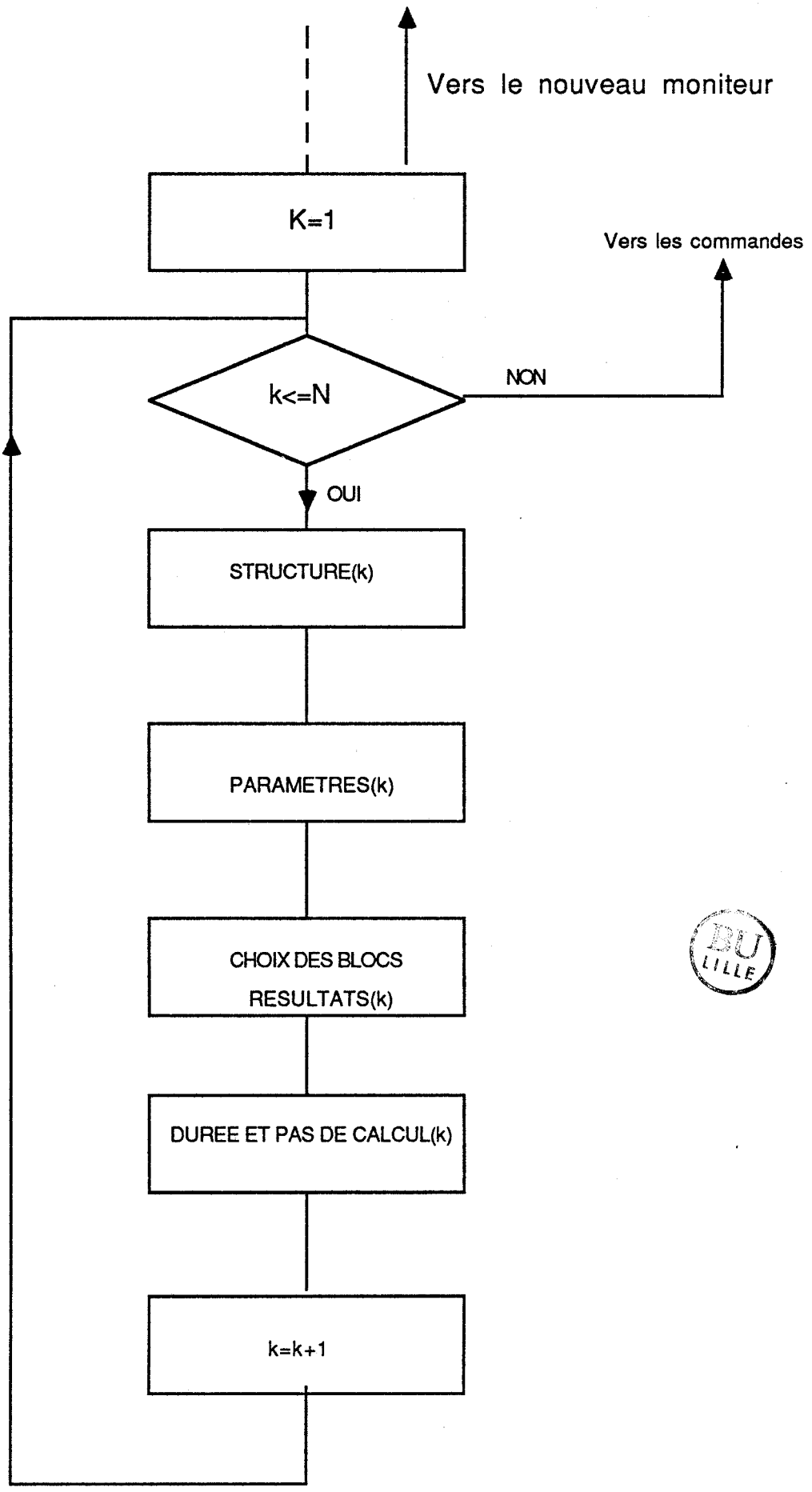


FIGURE 10

Les différents blocs du graphe de saisie et d'édition des tâches sont composés de programmes hybrides dont l'exécution de chacun permet :

- l'édition des réseaux relatifs à chaque processus,

- l'édition des coefficients ou paramètres associés à chaque bloc fonction du réseau dans chacun des processus à exécuter,

- l'édition des blocs associés aux variables dont les sorties permettent l'observation et l'analyse du comportement de chaque processus,

- l'édition de la durée et du pas de simulation de chaque processus.

La configuration du programme éditeur de réseaux et le programme interpréteur de blocs aident l'utilisateur à la constitution d'une table en mémoire pour chaque processus par le biais des tableaux précédemment définis.

L'interpréteur de blocs contrôle la liste des blocs prédéfinis pour déterminer les composantes de chaque processus ou modèle afin de la convertir plus tard à la demande dans des algorithmes exécutables par le calculateur.

Il est à noter que les composantes de type continu et celles de type discret descriptives du réseau de chaque processus peuvent être mélangées sans nécessairement un programme supplémentaire permettant leur couplage.

Chaque bloc défini peut être utilisé comme une composante de réseau de processus aussi souvent qu'il est nécessaire.

Le programme éditeur de paramètres offre à l'utilisateur la possibilité de coefficienter toute composante d'un réseau mettant en relief son importance dans la définition et la stabilité du processus. Nous verrons dans le prochain chapitre que cette possibilité se traduit en définitive par un moyen de réglage par variations successives de paramètres grâce à un sous-programme additionnel, nous évitant ainsi l'utilisation de la méthode de CAUCHY /16/ pour étudier les systèmes modélisés par une équation différentielle du second ordre à coefficients variables.

Pour les processus dont l'étude du comportement passe par l'observation de plusieurs variables ou variables de sortie de bloc, le programme éditeur des résultats de la simulation est doté d'une structure permettant quatre sorties différentes pour dix processus ou modèles différents, soit un nombre total de 40 sorties observables.

Enfin, pour ce qui concerne les programmes favorisant les opérations purement hybrides, les opérations de contrôle de séquences, les opérations d'échange et de synchronisation, l'ensemble dans le but d'un traitement parallèle des tâches, ils seront décrits dans le prochain chapitre.

CONCLUSION

Nous avons développé dans ce chapitre l'un des objectifs essentiels du moniteur hybride, c'est-à-dire, la saisie et l'édition de plusieurs tâches tout en conservant au logiciel son caractère temps réel aussi bien pour les transformations nécessaires qui ont été apportées que pour les traitements futurs qui seront engagés et décrits par les instructions macro ou sous-programmes hybrides dans le but des échanges entre processus symbolisant ainsi le parallélisme dans le traitement.

Le moniteur dispose de plusieurs commandes sous forme d'étiquette (LAB) permettant ainsi d'assurer la gestion et le contrôle temps réel des tâches.

Nous pouvons aisément dire de ce fait qu'il est doté d'une fonction multiprocesseur bien que le logiciel soit fondé sur une structure monoprocesseur.

CHAPITRE III

EXTENSION DU LOGICIEL DE BASE POUR L'EXECUTION PARALELLE : CONCEPT DE PRODUCTEUR-CONSOMMATEUR DANS LE TRAITEMENT EN MODE PARALELLE

INTRODUCTION

Dans la structure du moniteur hybride, plusieurs points de vue doivent être envisagés :

- notons, tout d'abord, la nécessité des macro-instructions dont les codages dans le cas présent sont basés sur l'utilisation à la fois des étiquettes ou des instructions du type "GOTO" et des sous-programmes, ce qui évite la précision des adresses absolues des points de branchement, la structure du langage Fortran assurant d'elle-même cette gestion.

- la précision du concept d'événements pour permettre la résolution des problèmes relatifs à la synchronisation des échanges d'informations entre les parties numériques et les parties analogiques du simulateur ; en effet, une instruction ne peut être déclenchée que par un seul événement ; il est cependant possible de composer plusieurs événements dans le but d'exécuter une seule instruction. Ce problème se résoud facilement grâce à l'introduction de primitives supplémentaires.

- enfin, au niveau de la programmation hybride, certaines données sont délicates à manipuler ; en effet, la fonction multiprocesseur est décrite par plusieurs programmes hybrides distincts qu'il est nécessaire de délimiter clairement. De plus, certains paramètres peuvent être transmis de l'un à l'autre.

Cette transmission nécessite de repérer les paramètres grâce à des pointeurs dans le tableau des paramètres et donc d'avoir accès à des informations de type purement informatique ; le même genre de problème se pose à l'intérieur d'un programme hybride où existent des ruptures de séquences ; les

points de branchement peuvent être repérés, soit par un déplacement relatif par rapport aux points de rupture, soit par l'étiquette du point de branchement.

Pour mieux préciser la nécessité des macro-instructions (illustrées par les sous-programmes) dans une programmation hybride, nous allons définir de façon succinte une instruction hybride et un programme hybride.

A. INSTRUCTION HYBRIDE

Une instruction comporte un certain nombre d'éléments caractéristiques de la tâche à accomplir.

Aussi, il convient de définir, d'une part, son mode d'exécution et, d'autre part, le traitement à effectuer.

Le mode peut être soit du type séquentiel, soit du type déclenché ; dans ce dernier cas, il faut préciser l'étiquette de déclenchement du sous-programme.

Le traitement est défini par une étiquette d'opération qui donne accès à la bibliothèque des sous-programmes qui réalisent l'exécution des opérations hybrides élémentaires. La description du traitement est complétée si nécessaire par un ensemble de paramètres.

B. PROGRAMME HYBRIDE

D'une façon générale, les sous-programmes relatifs à la résolution d'un problème sont regroupés en listes ou "programmes hybrides". Afin de faciliter l'analyse et le traitement des problèmes hybrides, nous avons envisagé la possibilité pour un même événement de mettre en oeuvre plusieurs tâches éventuellement indépendantes.

Pour illustrer ce point de vue, citons l'exemple du programme de "contrôle du clavier" grâce à la fonction "TESCON(I)" qui met en oeuvre, à la fois, l'opération de mise en GEL d'un traitement et l'archivage des résultats. Ce programme est totalement indépendant des programmes relatifs au problème de traitement en mode parallèle étudié.

De cette façon, il est possible d'exécuter plusieurs programmes en parallèle et de façon quasi simultanée.

Cette méthode permet de réaliser par le logiciel la fonction de multiprocesseurs ; compte tenu des besoins usuels relatifs aux traitements hybrides, le nombre de ces programmes est égal à quatre au maximum. Nous adopterons en conséquence pour ce chapitre une présentation en quatre parties :

. dans la première partie, nous donnerons la structure simplifiée du programme de gestion,

. nous présenterons dans la seconde partie quelques éléments de structure des programmes à exécution parallèle,

. une troisième partie traite de la méthode de passage automatique des paramètres,

. dans la quatrième partie, nous mettons en évidence la technique de forçage des valeurs dans une boucle de calcul.

I. STRUCTURE DU PROGRAMME DE GESTION

La structure globale du programme de gestion découle de l'analyse séquentielle des tâches qu'il doit accomplir.

Tout d'abord, les interruptions effectives hormis celles issues du programme de "contrôle clavier" sont programmées par l'intermédiaire des boucles de temporisation.

Le programme de gestion analyse ces interruptions et les compare aux étiquettes de déclenchement, en cas de concordance, il y a réalisation du ou des sous-programmes et exécution des opérations hybrides qu'elles décrivent ; les instructions hybrides constituent ainsi une source d'informations supplémentaires pour le programme de gestion au même titre que les événements.

Afin de simplifier l'organigramme descriptif de ce programme, nous n'avons pas décrit la fonction de multiprocesseurs ; nous proposons de le détailler dans l'organigramme général du moniteur. Le schéma de la figure 11 met en évidence les phases principales du programme de gestion.

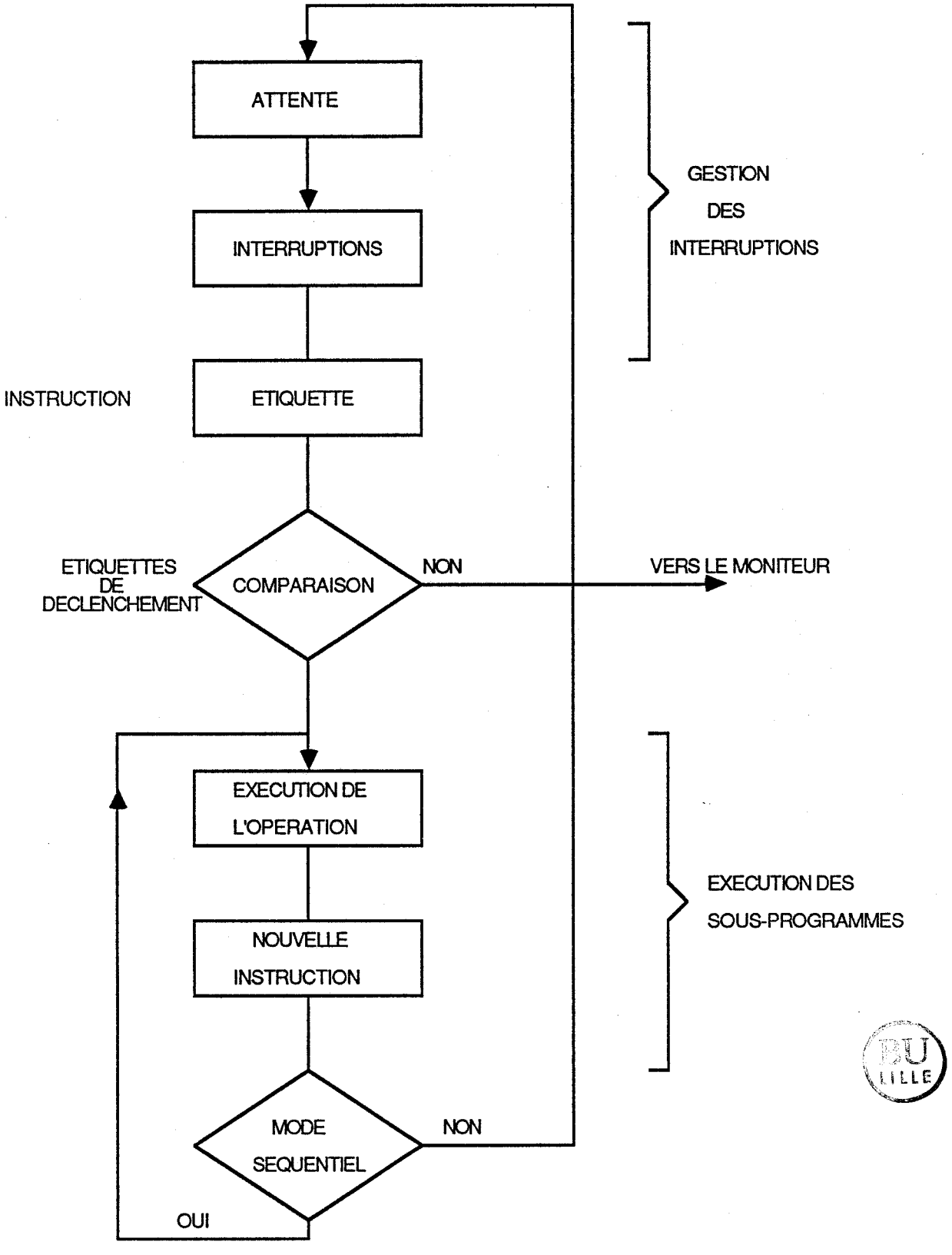


FIGURE 11

II. STRUCTURE DU PROGRAMME A EXECUTION PARALLELE

L'organisation interne du programme à exécution parallèle obéit à deux contraintes relatives à la fois à l'indépendance et à la simultanéité de certains traitements. De ce fait, nous avons réalisé la fonction multiprocesseur grâce à l'écriture de plusieurs programmes hybrides ; chaque programme correspond à un traitement particulier et peut être indépendant des autres.

L'indépendance dans les traitements ne concerne en fait que l'exécution ; un programme aura donc la possibilité d'utiliser des paramètres définis dans d'autres programmes ou encore de transmettre des informations vers le sous-programme de calcul ; par ce biais, on aboutit à la notion de producteur/consommateur dont l'exemple ci-dessous illustre le mécanisme /21,23/.

EXEMPLE :

Soient deux tâches T_1 et T_2 qui se communiquent l'information "Résultat" à travers une table où sont rangés les résultats d'un processus.

La tâche T_2 (Producteur) fournit un résultat à la tâche T_1 (Consommateur) qui va l'utiliser comme condition initiale pour démarrer sa propre évolution.

Les valeurs initiales des deux sémaphores sont allors données sous la forme : (Plein, Vide)

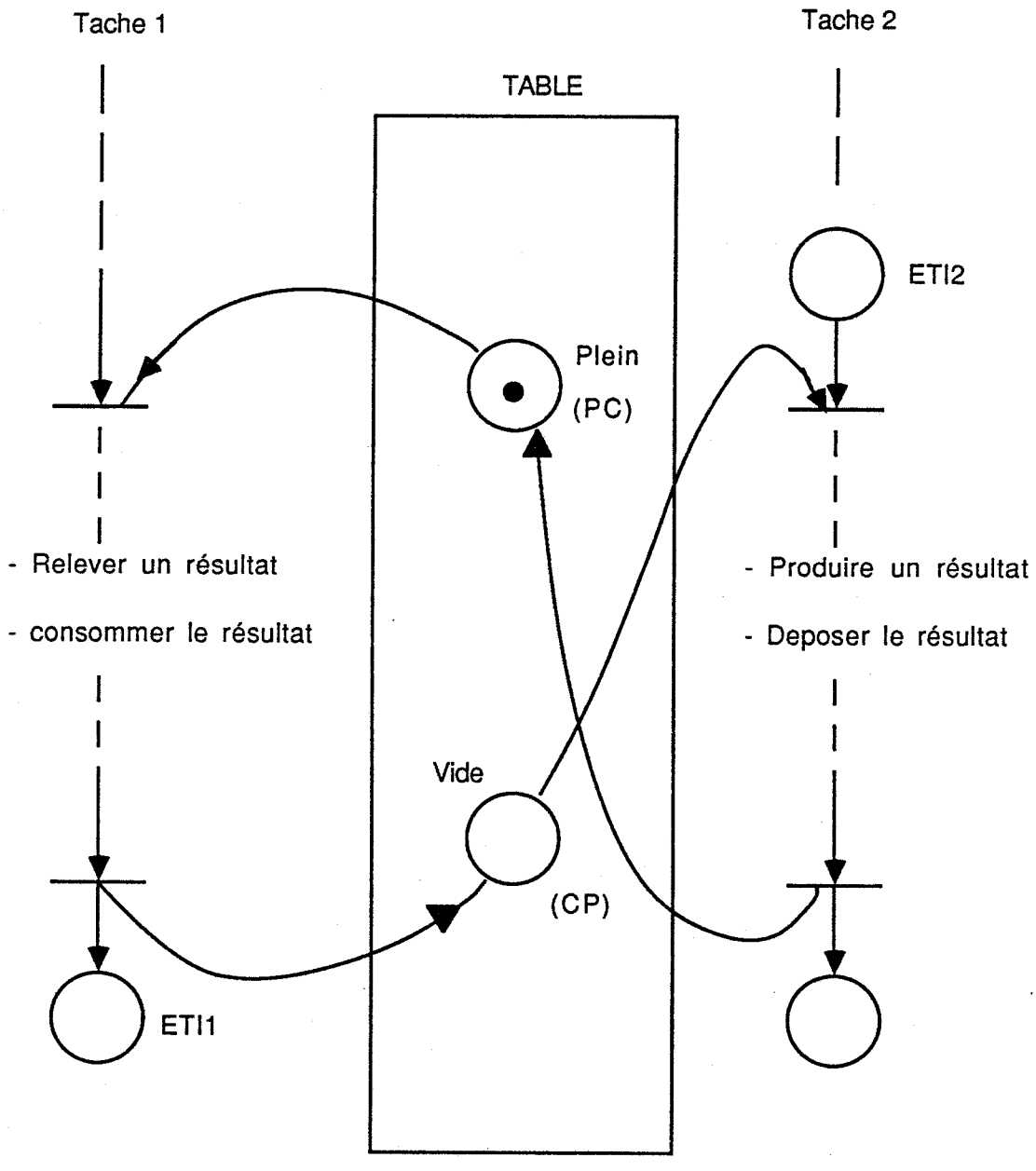


FIGURE 12

Dans l'exemple i-dessus, les tâches 1 et 2 représentent deux modèles de processus dont la description et la saisie sont conformes au nouveau mode d'édition décrit précédemment.

Le déroulement des séquences de l'opération de communication est décrit suivant la partie de programme ci-après.

```
READ (5,*) IM                          Recherche de la tâche n° 1

INTER = .FALSE.                        Validation de la tâche par un prédicat

CALL SIMUL (IM,INTER)                  Lancement de la tâche IM

IF(KPT. EQ. N) GOTO ETI 2              N boucles de calcul

GOTO ETI 1                             Poursuivre la simulation

ETI 2 READ(5,*)IH                      Recherche de la table IH

CALL RESULT(IM,IH)                     Production de résultat par la tâche IM

OUT(IM) = VAL                          Dépôt du résultat par affectation

INTER = .TRUE                          Inhibition de la tâche IM

CALL SIMUL (IM,INTER)

IM = IM ± 1                            Recherche de la tâche n° 2

INTER = .FALSE.                        Validation de la tâche n° 2

CALL CONDIT (IM)                       Mise en CI de la tâche n° 2

CALL MODCAL (IM, INTER, VAL)           Lancement de la tâche n° 2 avec le
                                       résultat VAL produit par la tâche n° 1

ETI 1 : LAB = P                        Poursuite du traitement
```

Peut-on parler d'un réel parallélisme dans l'exécution des tâches 1 et 2 ?

En premier lieu, on est tenté de répondre "NON" puisque les séquences "Production/Dépôt de résultat et Prélèvement/Consommation" du résultat ne peuvent naturellement se dérouler que l'une après l'autre, sauf dans le cas où l'on dispose de deux processeurs indépendants, chacun exécutant dans le même temps une des deux séquences.

Cependant, si on considère que la tâche 2 produit le résultat, le dépose et continue son traitement (GOTO ETE1), permettant ainsi à la tache 1 de prélever le résultat, le consommer et de poursuivre aussi son traitement grâce à l'usage de la même étiquette, on peut alors dire qu'il y a un certain parallélisme proche de celui des deux processeurs évoqué ci-dessus. Aussi, dans l'exemple traité, nous parlerons plutôt d'un pseudo parallélisme que d'un réel parallélisme dans l'exécution des tâches 1 et 2.

* Dans le cas où les tâches s'excluent mutuellement, la communication se réduit à une liaison de partage de ressources à un seul état ou encore liaison de type exclusion mutuelle.

Lorsque les deux tâches T_1,T_2 se trouvent dans un tel cas, elles se partagent l'utilisation de la même information (Résultat) de la manière suivante :

- la première comme condition initiale pour démarrer,

- la seconde comme paramètre de réglage pour une nouvelle boucle de calcul.

Ainsi, le sémaphore "Résultat" peut être interprété comme une variable boolenne initialisée à 1.

La représentation du déroulement des tâches peut être illustrée par le schéma suivant :

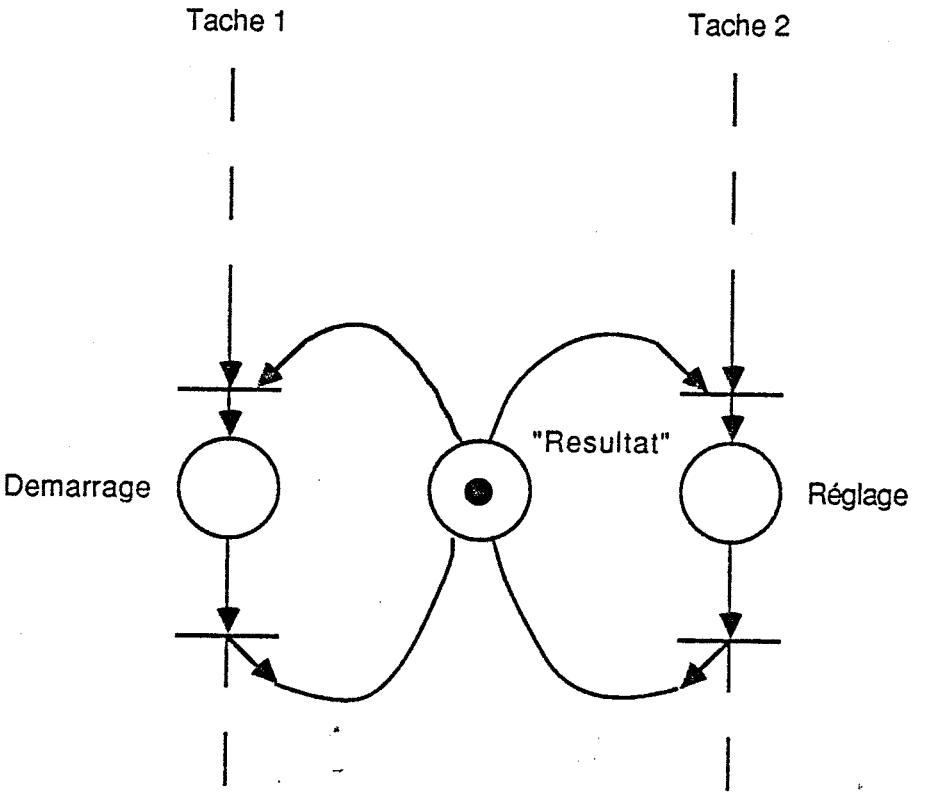


FIGURE 13

Comme précédemment, les tâches 1 et 2 sont décrites conformément aux modèles TUTSIM

d'où la partie du programme suivante :

```
READ(5,*)IM                          Recherche de la tâche 1

INTER = .FALSE.                      Validation par un prédicat

CALL SIMUL(IM,INTER)                 Lancement de la tâche 1
```

IH = K (1 = K < 5) CALL RESULT (IM,IH)	Production d'un résultat
IF(OUT(IH).GT.0.0) GOTO ETI 1	Choix conditionnel du résultat
INTER = .TRUE. CALL SIMUL (IM,INTER)	GEL de la tâche 1
IM = IM $\pm$ 1 INTER = .FALSE. CALL REGPAR (IM,IB,KPAR,OUT(IH))	Recherche de la tâche 2 Validation de la tâche 2 par le prédicat Affectation du résultat OUT(IM) au paramètre KPAR du bloc IB dans la tâche 2
CALL SIMUL (IM,INTER)	Lancement de la tâche 2 après réglage
ETI1: CALL CONDIT (IM,INTER)	Mise en CI de la tâche 1
INTER = .FALSE. CALL SIMUL (IM,IH)	Exécution d'une nouvelle boucle de calcul par la tâche 1
ETI2 : LAP = P	Poursuite du traitement

III. METHODE DE PASSAGE AUTOMATIQUE DES PARAMETRES

La saisie d'un processus passe par l'enregistrement d'un certain nombre de variables auxquelles sont affectés des coefficients multiplicatifs (paramètres). Un sous-programme d'édition des paramètres permet de saisir de manière globale et figée ces paramètres.

Le mécanisme de réglage qui aboutit à l'obtention des solutions optimales pour un processus en cours d'exécution grâce aux modifications successives des paramètres par injection de nouvelles valeurs peut désormais se dérouler de manière automatique. En effet, il n'est plus nécessaire d'arrêter le processus en cours de simulation pour faire appel au sous-programme d'édition des paramètres en vue d'un éventuel réglage du processus. Ces paramètres étant af-

fectés à des blocs fonctions descriptifs du réseau, nous avons écrit deux sous-programmes qui pour un processus donné permettent :

- l'un la saisie de nouvelles valeurs de paramètres,
- le second, l'affectation de ces valeurs à des blocs précis.

L'ensemble de ces opérations se déroule de manière instantanée pendant la phase de simulation ou en mode calcul. Toute la procédure d'affectation automatique des paramètres est schématiquement illustrée par une partie de l'organigramme de la figure 14.

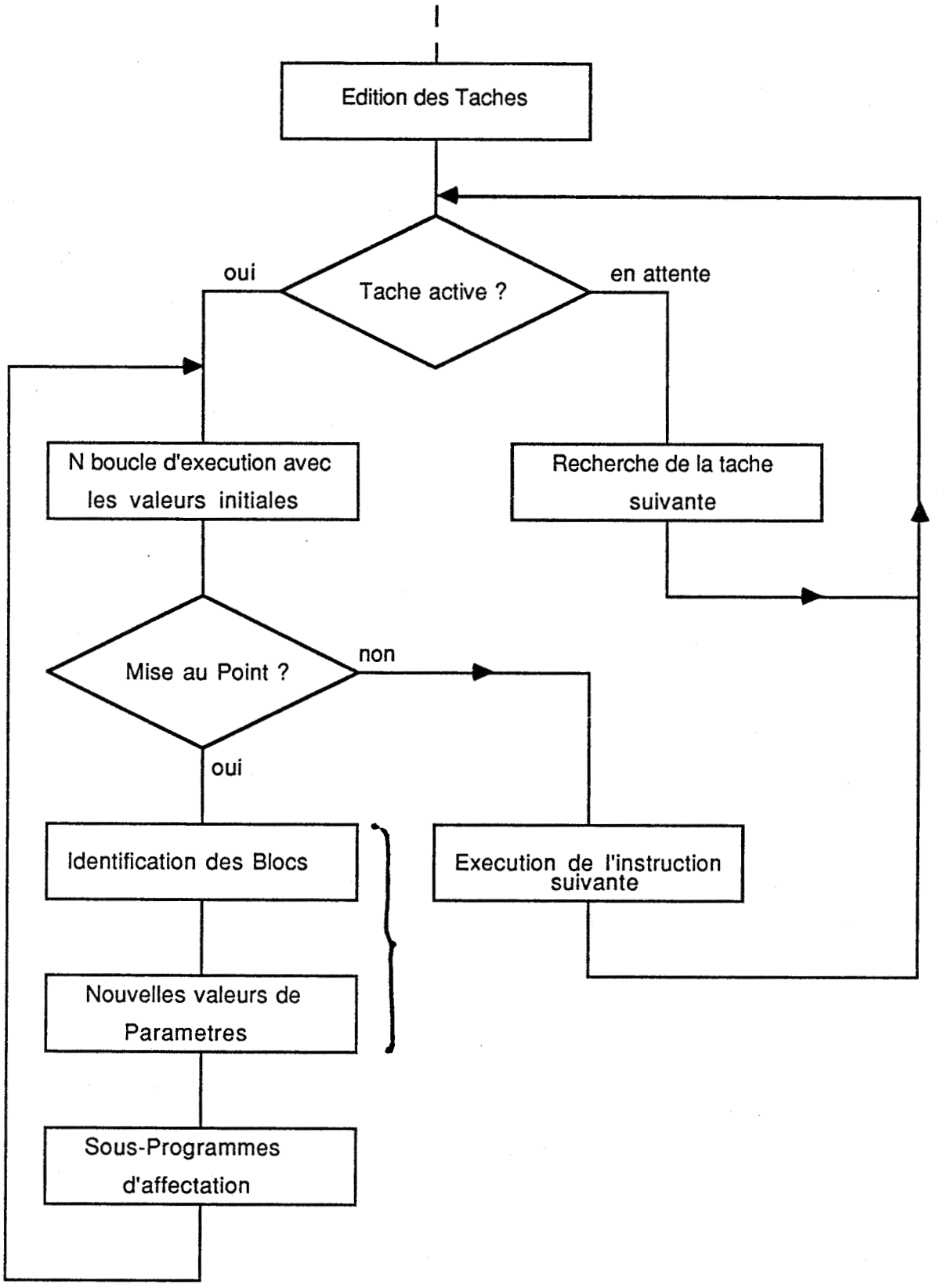


FIGURE 14

IV. REGLAGE DE PROCESSUS PAR LA TECHNIQUE DE FORCAGE

IV.1 INTRODUCTION

L'utilisation des paramètres pour le réglage d'un processus par le simulateur est analogue au système de commande multivariables. Pour une fonction ou un état donné, c'est la réponse du processus aux modifications de la ou des consignes (paramètres). Il convient cependant de distinguer deux types de forçages :

- le premier lié à une partie décisionnelle dans une unité de production permet la modification des modes de marches et donc un concept technologique différent.

Pour illustrer ce concept, prenons l'exemple de marquage de place dans un graphe de Pétri pour modéliser les systèmes de production ; en effet, la condition d'activation d'une transition "T" est relative à la fois au marquage de la place P en amont et à l'événement "E" associé /17,22,24/.

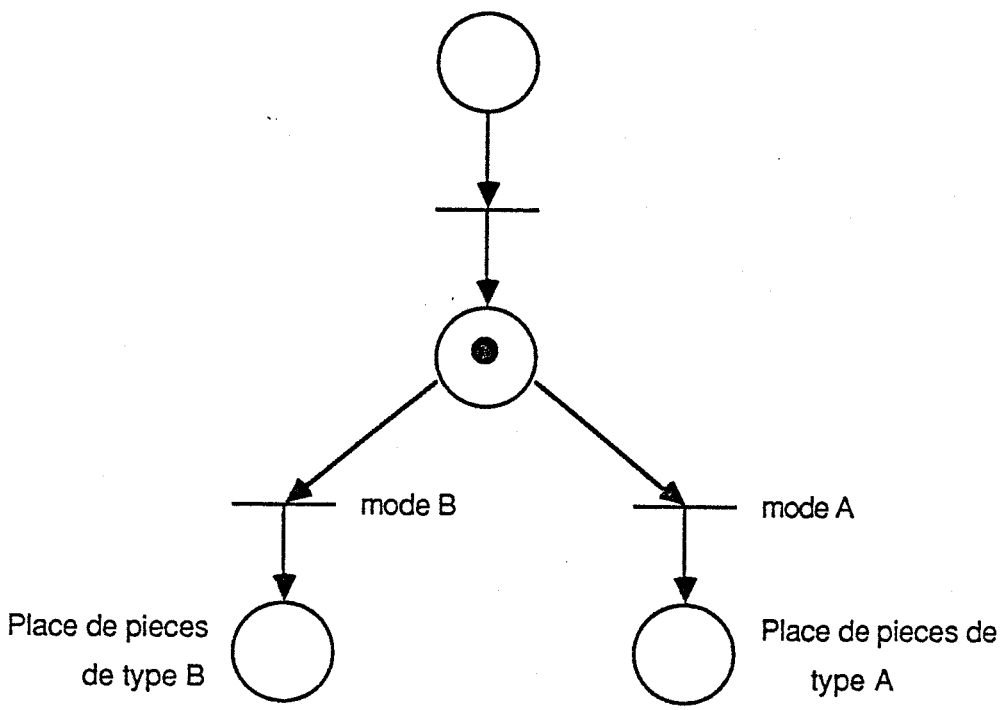


FIGURE 15

Les événements associés aux modes pouvent être rattachés à la partie décisionnelle modélisée par des places dont la marque ne rend pas forcément franchissable la transition ; elle reste alors inhibée tant que l'événement E n'est pas arrivé ; le forçage dans ce cas consiste à la mise à "1" de E.

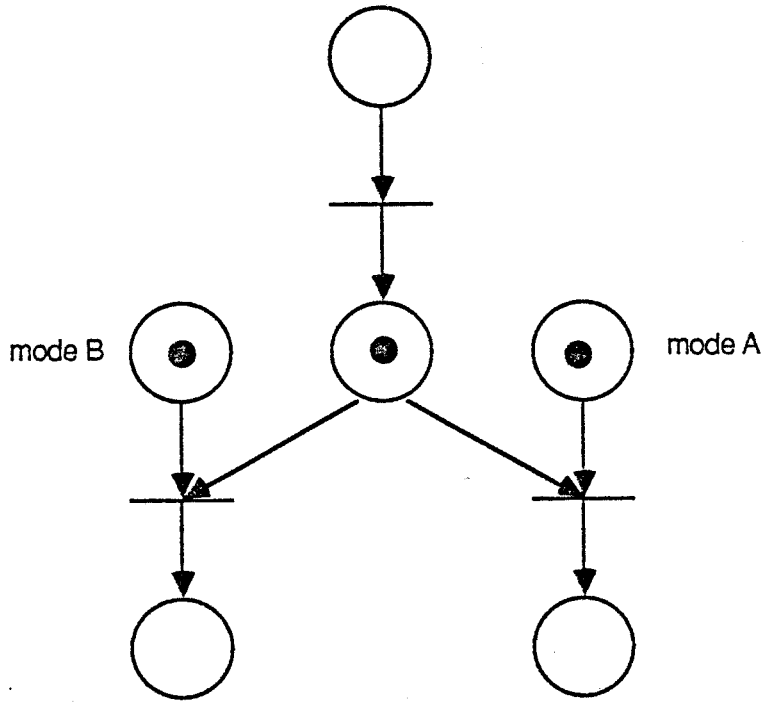


FIGURE 16

- Le deuxième type de forçage est relatif aux différentes contraintes auxquelles on soumet de manière arbitraire, soit l'évolution d'un processus, soit les résultats issus de la simulation.

Certaines de ces contraintes sont elles-mêmes soumises à d'autres comme par exemple la délimitation d'une zone au delà de laquelle la stabilité de simulation du processus est compromise.

Nous citons l'exemple d'un processus soumis à une perturbation modélisée par la fonction U(t) délivrée par un générateur de bruit. La contrainte pour une répartition uniforme du bruit est donnée par l'échelle de distribution de

U(t), soit l'intervalle fermé |-1,+1| tel que (-1 < U(t) ⩽ +1).

Pour ce qui concerne le forçage des résultats pendant la phase de simulation du processus, il consiste, soit à mettre un tableau de résultats à zéros, soit à échanger les résultats de deux tableaux d'un même processus ou d'un même élément de table de deux processus différents.

Avant forçage

NRPL (K,1)
0.505
0.510
0.600
⋮
0.800

n pas de simulation

Après forçage

NRPL (K,1)
0.000
0.000
⋮
0.000

Exemple dans le cas de deux processus différents

NRPL (K,1)
0.505
⋮

NRPL(K±1,2)
0.805
⋮

Avant forçage

NRPL (K,1)
0.805

NRPL(K±1,2)
0.505

Après forçage

IV.2 EXECUTION DES FORCAGES PAR LE LOGICIEL

Le logiciel offre la possibilité de réaliser trois autres extensions dans la ligne du second type de forçage décrit précédemment et portant sur :

- le forçage du réseau descriptif du modèle conduisant à une modification de structure,

- le forçage des paramètres,

- le forçage de pas de calcul.

Notons que ces différents forçages peuvent être réalisés pendant la phase de simulation sur un même processus ou sur plusieurs processus indépendants de deux manières différentes, c'est-à-dire, qu'ils peuvent être soit de type séquentiel et simultané, soit de type séquentiel et indépendant les uns des autres, l'ensemble dans un ordre quelconque.

Nous donnons deux exemples pour illustrer ce type de fonctionnement dans les deux cas soulevés.

1er cas : Un seul et même processus

Soit un processus P1 dont l'exécution passe par le forçage de type séquentiel et simultané, il est donc soumis à l'évolution décrite schématiquement par l'organigramme suivant

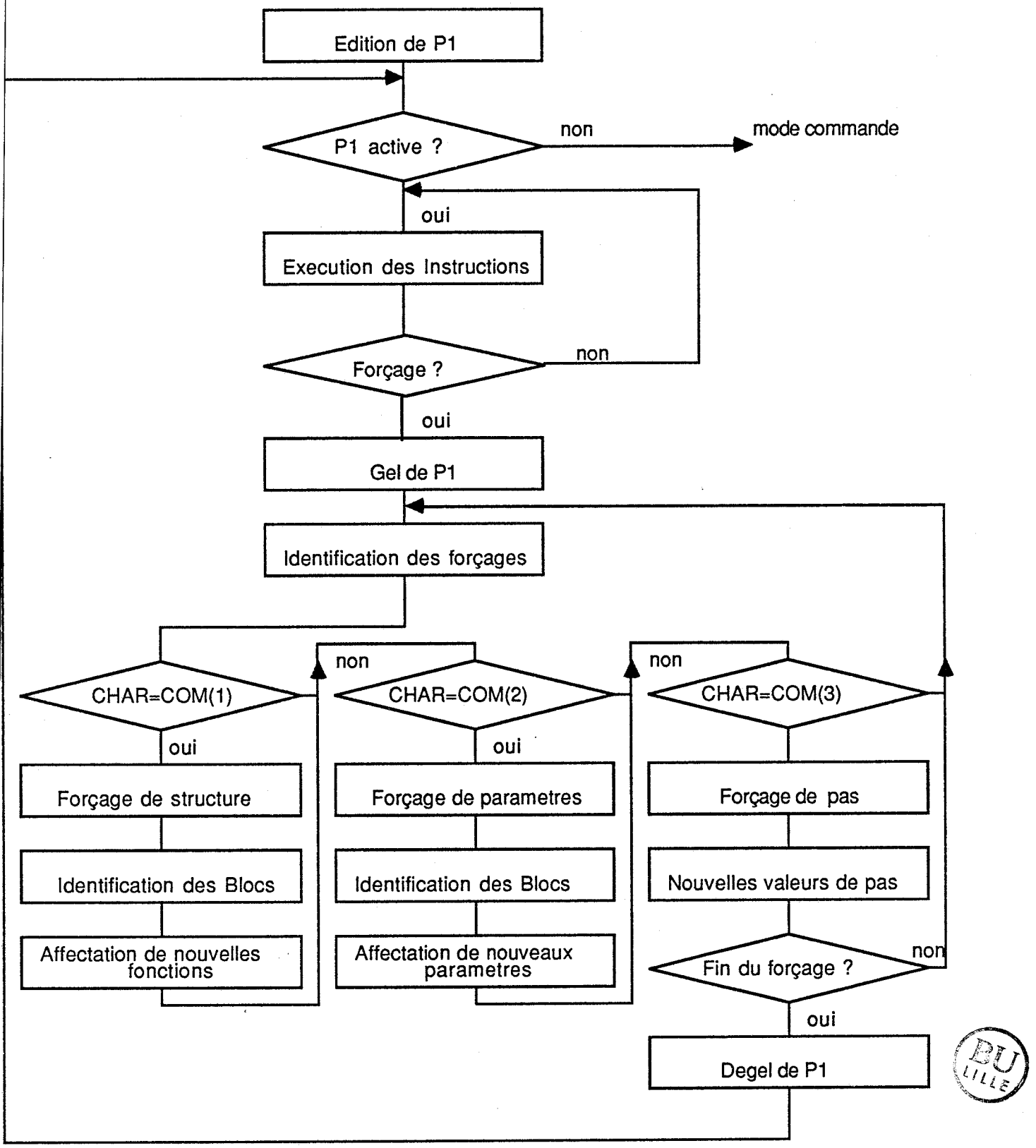

Edition de P1
P1 active ?
non
mode commande
oui
Execution des Instructions
Forçage ?
non
oui
Gel de P1
Identification des forçages
CHAR=COM(1)
non
CHAR=COM(2)
non
CHAR=COM(3)
oui
oui
oui
Forçage de structure
Forçage de parametres
Forçage de pas
Identification des Blocs
Identification des Blocs
Nouvelles valeurs de pas
Affectation de nouvelles fonctions
Affectation de nouveaux parametres
Fin du forçage ?
non
oui
Degel de P1

FIGURE 17

La saisie du processus P1 se déroule de la même manière que celle des tâches 1 et 2 de l'exemple précédent. D'où la partie du programme ci-dessous :

```
READ (5,*) IM ..................... Initialisation de la tâche IM

IF (IM.EQ.P1)  GOTO ETI1
GOTO ETI2 ......................... Retour au mode commande

ETI1 : INTER =. FALSE .
                                    Activation de P1
CALL SIMUL (IM, INTER)

       IF (CHAR.EQ.F) GOTO ETI3 ....  Requête de forçage

ETI3 : IF (CHAR.EQ.COM (1) )  THEN
       INTER =.TRUE.
                                    GEL DE P1
       CALL SIMUL (IM, INTER)

       CALL INSTR (IM) .............  Froçage de structures

       ELSE IF (CHAR.EQ.COM(2) ) THEN
         INTER =.TRUE.
         CALL SIMUL (IM, INTER)
         CALL IMPAR (IM) ...........  Forçage de paramètres


       ELSE IF (CHAR.EQ.COM(3) ) THEN
         INTER = .TRUE.
         CALL SIMUL (IM, INTER)
         CALL INTIMG (IM) ..........  Forçage de pas


       ELSE

       GOTO ETI 1
            ENDIF

ETI2   :   GOTO 21 .................  Retour au mode commande
```

2ème cas : Deux processus P1 , P2 dont les traitements sont soumis au forçage de type séquentiel et indépendant. L'organigramme de principe est donné par le schéma suivant :

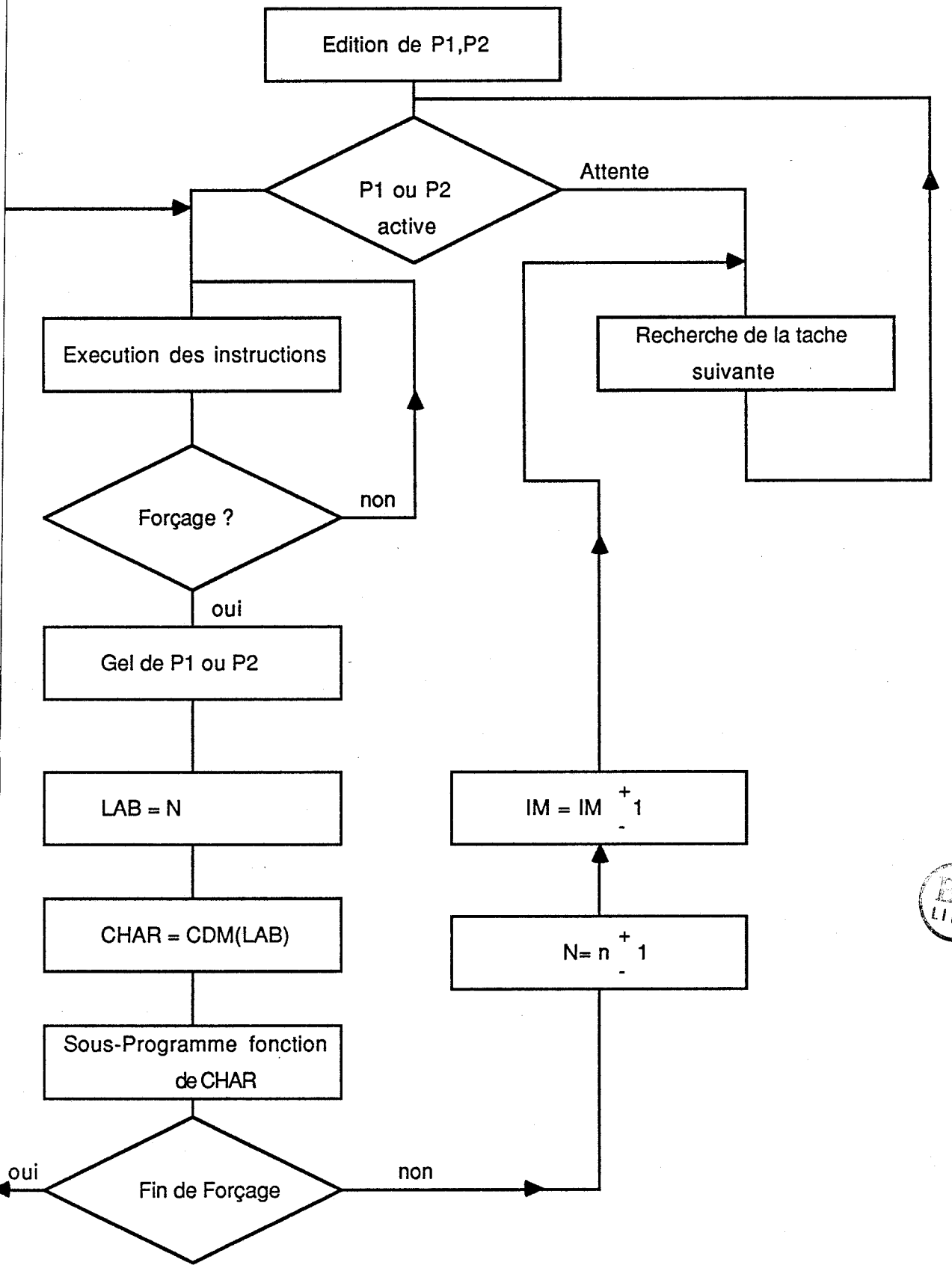


FIGURE 18

La saisie de P1 et P2 est conforme au cas précédent ; la séquence de programme relative à ce type de traitement est la suivante :

```
READ(5,*)IM                          Initialisation de IM=f(P1,P2)

  ETI1: IF(IM.EQ.P1.OR.IM.EQ.P2) THEN   Validation du prédicat et
          INTER=.FALSE.                 activation de P1 ou P2 au choix de
          CALL SIMUL(IM,INTER)          l'utilisateur

              IF(CHAR.EQ.F) GOTO ETI2   Requête de forçage
                     GOTO ETI1          Poursuite de la simulation de P1 ou P2

          ELSE

          GOTO ETI3

              ETI2: INTER=.TRUE.    ┐
              CALL SIMUL(IM,INTER)  ┘   GEL de P1 ou P2
              LAB=N                     Identification du forçage
              CHAR=COM(LAB)
              CALL CHANGE(CHAR)         Fin du 1er forçage de P1

          IM=IM ± 1   ┐
          N=N ± 1     ┘                 Identification du 2ème forçage dans P2

          GOTO ETI1                     Réactivation de P1 ou P2

ETI3: GOTO 21                           Retour au mode commande
```

CONCLUSION

Tout d'abord, les programmes hybrides présentés dans ce chapitre ont été définis de manière à permettre aux instructions du moniteur la réalisation de la fonction multiprocesseur grâce à la transmission des paramètres d'une tâche vers l'autre.

Dans le même sens, le concept d'événement a permis de résoudre les problèmes relatifs à la synchronisation des échanges d'informations entre les processus tout en conservant la structure temps réel dans les traitements.

Il faut noter que le domaine de variation de certains paramètres est limité, ce qui implique certaines contraintes d'échelle et de précision comme nous l'avons mentionné.

En second lieu, pour permettre à l'utilisateur un contrôle aisé des processus et leur réglage sous forme de contraintes arbitraires, plusieurs points de forçage ont été introduits dans le programme moniteur.

Enfin, l'emploi systématique d'étiquettes et des instructions élémentaires du genre "GOTO" simplifie considérablement la détermination des points de branchement relatifs aux instructions de rupture de séquence conditionnelle et inconditionnelle tout en facilitant par ailleurs dans certains cas le repérage de paramètres.

CHAPITRE IV

EXEMPLES D'APPLICATION

INTRODUCTION

Nous proposons dans ce chapitre d'illustrer suivant trois exemples particuliers les techniques spécifiques de la simulation hybride découlant directement des différentes possibilités d'utilisation du logiciel présenté dans ce mémoire.

L'analyse et la simulation des phénomènes nécessitant un couplage analogique/numérique passe obligatoirement par un choix judicieux des opérateurs utilisables (analogiques) et des algorithmes fonctionnels (numériques).

En outre, il est généralement possible de décomposer le problème en mettant en évidence plusieurs niveaux de programmation ; c'est dans ce sens qu'il faut considérer qu'un programme hybride résulte de l'association globale et synchronisée d'un cablage analogique, d'un "cablage numérique" et d'un ensemble de programmes de traitement et d'échanges réalisés sur la base du logiciel développé.

Pour illustrer ces points de vue tout en mettant en évidence le procédé simple et rapide de calcul hybride que permet l'utilisation du logiciel, nous proposons d'étudier dans une première partie les problèmes relatifs à la cinétique chimique . / 25 /

La seconde application concerne la simulation de la trajectoire du mouvement du siège éjectable à partir d'un avion en vol ; l'intérêt de cette étude porte sur l'application des sous-programmes qui permettent le passage séquentiel et automatiques des paramètres pour déterminer la date (temps) du passage du siège par la verticale stabilisateur (Xs, Ys) dans le repère avion / 26, 27, 28 /

En dernier lieu, un problème régi par un modèle de type à "constantes réparties" relatif à la résolution d'équation aux dérivées partielles permet d'évaluer globalement l'intérêt d'un tel système de simulation. Dans ce cas, le logiciel permet de résoudre simplement un problème de parallélisme ralativement complexe.

I. RESOLUTION D'UN PROBLEME DE CINETIQUE CHIMIQUE.

I.1. THEORIE DE LA CINETIQUE CHIMIQUE

La cinétique d'un système chimique est caractérisée par les relations entre la vitesse de transformation, le temps de réaction et la concentration des substances du mélange réactionnel.

Prenons par exemple deux substances A et B qui entrent en réaction ; l'équation générale du mélange réactionnel est telle que :

$$\alpha A + \beta B \xrightarrow{K} \gamma C + \delta D$$

L'application de la loi d'action des masses caractérise le fait que la vitesse de la réaction est proportionnelle au produit des concentrations des corps initialement mis en présence ; elle est donnée par la formule de Van't Hoff suivant l'expression :

$$V = K [A]^p [B]^q \quad \text{où}$$

- K est une constante de la vitesse de réaction.
- p + q, l'ordre de la réaction.
- $[A]$ et $[B]$, les concentrations courantes des substances A et B.

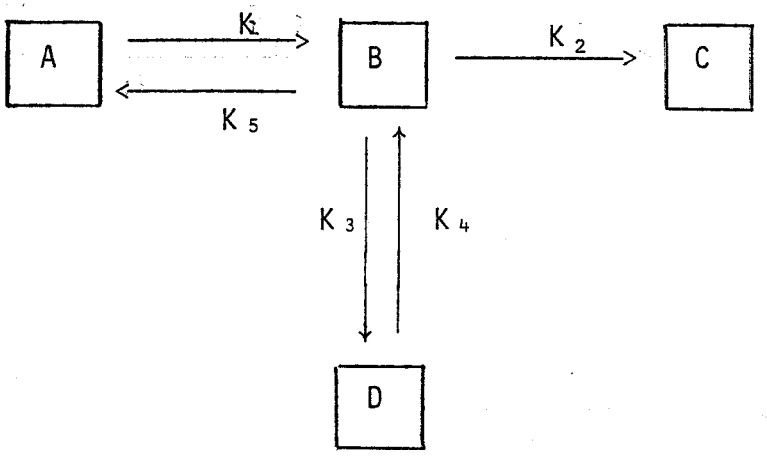
On considère généralement $p = \alpha$ et $q = \beta$

Par ailleurs, la vitesse de la réaction dépend de la molarité de la concentration qui est fonction du nombre de molécules intervenant à chaque stade de la réaction ; d'où l'équation différentielle donnant l'expression de la vitesse en fonction des concentrations :

$$V = \frac{1}{\gamma} \frac{d[C]}{dt} = \frac{1}{\delta} \frac{d[D]}{dt} = -\frac{1}{\alpha} \frac{d[A]}{dt} = -\frac{1}{\beta} \frac{d[B]}{dt}$$

I.2. POSITION DU PROBLEME

Le problème usuel en cinétique chimique est la recherche des concentrations maximales d'un ou de plusieurs produits intermédiaires intervenants dans une série de réactions consécutives réversibles ou non. Aussi nous avons envisagé le cas d'un modèle de réaction suivant le schéma ci-dessous :



La réaction est parallèle, réversible et du premier ordre.

Au terme de la réaction les produits A et B se décomposent en :

* un produit intermédiaire : D

* un produit final : C

Nous supposerons pour simplifier les calculs que les constantes de la réaction de dégradation des produits sont fixées. Le problème se réduit finalement à chercher en fonction du temps la concentration maximale du produit C par variations successives des valeurs des paramètres K_1 et K_2.

La programmation des paramètres et la simulation hybride peuvent être envisagés sur plusieurs niveaux en décomposant l'analyse du problème en trois phases :

- Dans une première phase nous étudierons le modèle mathématique descriptif de l'évolution temporelle des concentrations.

- Dans la deuxième phase nous définirons les blocs fonctionnels et leur cablage afin d'obtenir le réseau descriptif du modèle.

- Enfin nous définirons les commandes et les sous-programmes hybrides nécessaires à la variation des paramètres K_1, K_2 et à la simulation de l'ensemble.

I.3. MODELE MATHEMATIQUE DE L'EVOLUTION TEMPORELLES DES CONCENTRATIONS

L'application des lois de la cinétique chimique à chacun des produits du mélange réactionnel aboutit à l'obtention d'un système d'équations différentielles décrit par la relation ci-dessous

$$\frac{d[A]}{dt} = - K_1 [A] + K_5 [B] \qquad (1)$$

$$\frac{d[B]}{dt} = K_1 [A] - K_5 [B] - K_3 [D] - K_2 [B] + K_4 [D]$$

$$\frac{d[C]}{dt} = K_2 [B] \qquad (3)$$

$$\frac{d[D]}{dt} = K_3 [B] - K_4 [D] \qquad (4)$$

Avant d'établir le réseau correspondant à chaque équation, il convient de faire une remarque sur le choix des blocs fonctionnels représentatifs des constantes de vitesse K_1, K_2 ..... K_5 de la réaction.

REMARQUE : Etant donné que la recherche du maximum de la concentration du produit C passe par la variation de paramètres, nous allons attribuer aux constantes de vitesse K_1, K_2 un bloc descriptif de la fonction "GAIN", les autres étant considérés comme des constantes réelles.

I.4. RESEAU REPRESENTATIF DU SYSTEME D'EQUATIONS

Les symboles utilisés pour chaque bloc sont donnés dans le Chapitre I.

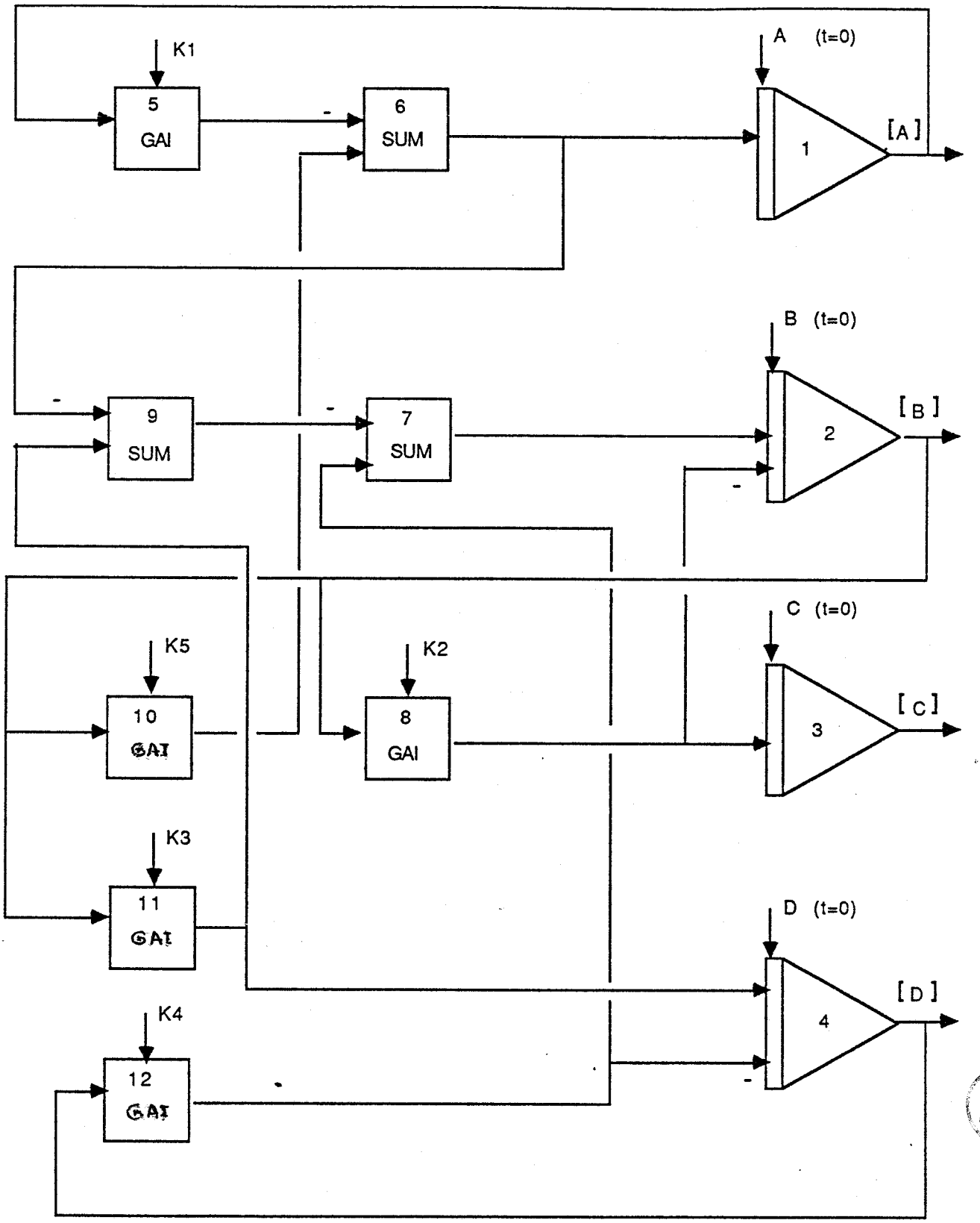


FIGURE 19

I.5. PHASE DE SIMULATION ET RECHERCHE DU MAXIMA DE CONCENTRATION

I.5.1. INTRODUCTION

En théorie le maximum de la concentration (C) est atteint lorsque sa dérivée $\frac{d}{dt}$ (C) s'annule ; or pour ce qui nous concerne ce maxima est fonction du couple de valeurs de gain (K_1, K_2) ; il est donc possible de relever une famille de points d'enregistrement de l'évolution de la concentration (C) en fonction du temps et de K_1 (K_2, fixe) ou de K_2 (K_1, fixe).

Ce problème est résolu par le logiciel en faisant des appels successifs au sous-programme de variation automatique de paramètres et en identifiant successivement le bloc de K_1 et le bloc de K_2.

Par ailleurs l'exécution du réseau par le moniteur de travail du logiciel nécessite la donnée de certaines valeurs numériques initiales ; aussi nous proposons par hypothèse les valeurs numériques suivantes si l'on considère que l'observation du processus s'étend sur une durée de 10 heures et que les constantes de vitesse s'expriment en unité de litre/heure et les concentrations en Kmol. /m^3.

$K_1 = 0,4$ $\quad$ $[A]\ (0) = 7$

$K_2 = 0.16$ $\quad$ $[B]\ (0) = 2$

$K_3 = 0.13$ $\quad$ $[C]\ (0) = 0$

$K_4 = 0.08$ $\quad$ $[D]\ (0) = 0.5$

$K_5 = 0.10$

I.6. SAISIE DU RESEAU

Conformément à la numérotations des blocs du schéma de la figure N° la structure, les paramètres et les sorties du réseau peuvent être édités de la manière suivante :

a) Structure :

```
 1, INT, 6
 2, INT, 7
 3, INT, 8
 4, INT, - 12, 11
 5, GAI, 1
 6, SUM, - 5, 10
 7, SUM, - 9, 12
 8, GAI, 2
 9, SUM, - 6, 11
10, CON, 2
11, CON, 2
12, CON, 4
```

b) Paramètres

```
0,4
1,7
2,2
3,0
4,0.5
8,0.16
10,0.10
11,0.13
12,0.8
```

d) Blocs descriptifs de l'évolution temporelle des concentrations.

X_{11} : 0, 0, 10

Y_{11} : 1, 0, 10

Y_{12} : 2, 0, 10

Y_{13} : 3, 0, 10

Y_{14} : 4, 0, 10

e) Durée d'observation du processus

0, 1, 10

I.7. LOGIQUE DE RECHERCHE DE LA CONCENTRATION MAX.

La détermination du maximum de la concentration [C] est assurée par l'exécution du réseau à travers 2 programmes dont nous donnons une brève description.

La première partie commence par l'initialisation des intégrateurs à partir des valeurs numériques des conditions initiales et par l'exécution de n boucles de calcul avec le premier couple de valeurs K_1, K_2.

A la fin de cette première exécution, il y a affichage du contenu du bloc N°3 (Y_{13}) représentatif de l'enregistrement de la concentration du produit [C] puis incrémentation de K_2 suivie d'une nouvelle boucle de calcul.

La deuxième partie qui commence par un ordre d'arrêt, identifie le bloc N°8 (K_2) , incrémente son contenu, réinitialise les intégrateurs, puis exécute une nouvelle boucle de calcul à la fin de laquelle il y a comparaison des valeurs de concentration du produit [C] .

La figure N° 20 illustre l'organigramme de principe de l'ensemble.

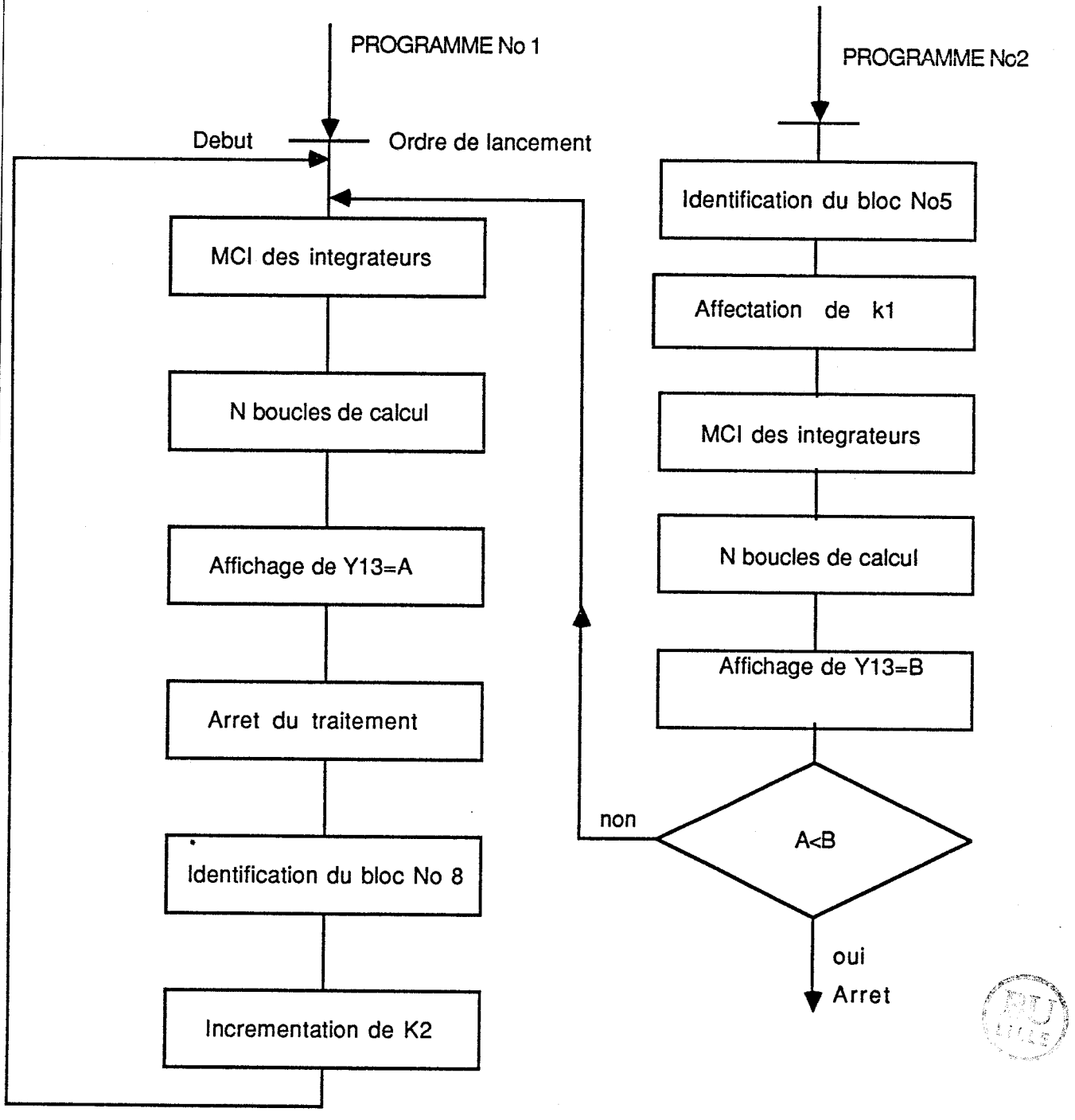

PROGRAMME No 1
Debut
Ordre de lancement
MCI des integrateurs
N boucles de calcul
Affichage de Y13=A
Arret du traitement
Identification du bloc No 8
Incrementation de K2
PROGRAMME Nc2
Identification du bloc No5
Affectation de k1
MCI des integrateurs
N boucles de calcul
Affichage de Y13=B
A<B
non
oui
Arret

FIGURE 20

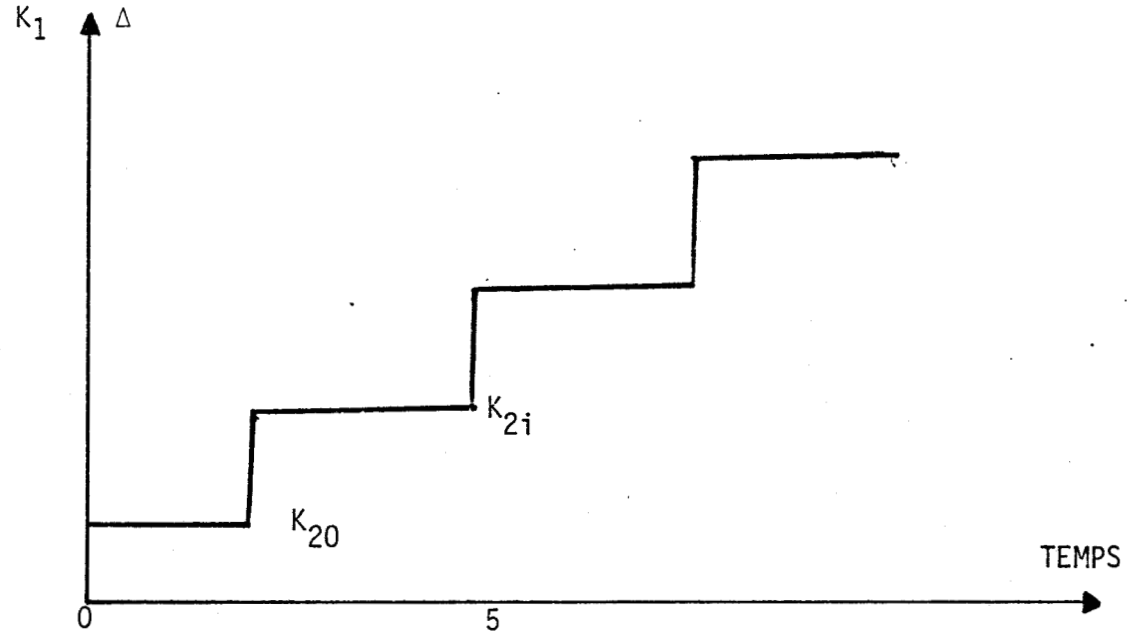


FIGURE 21a EVOLUTION DE K1

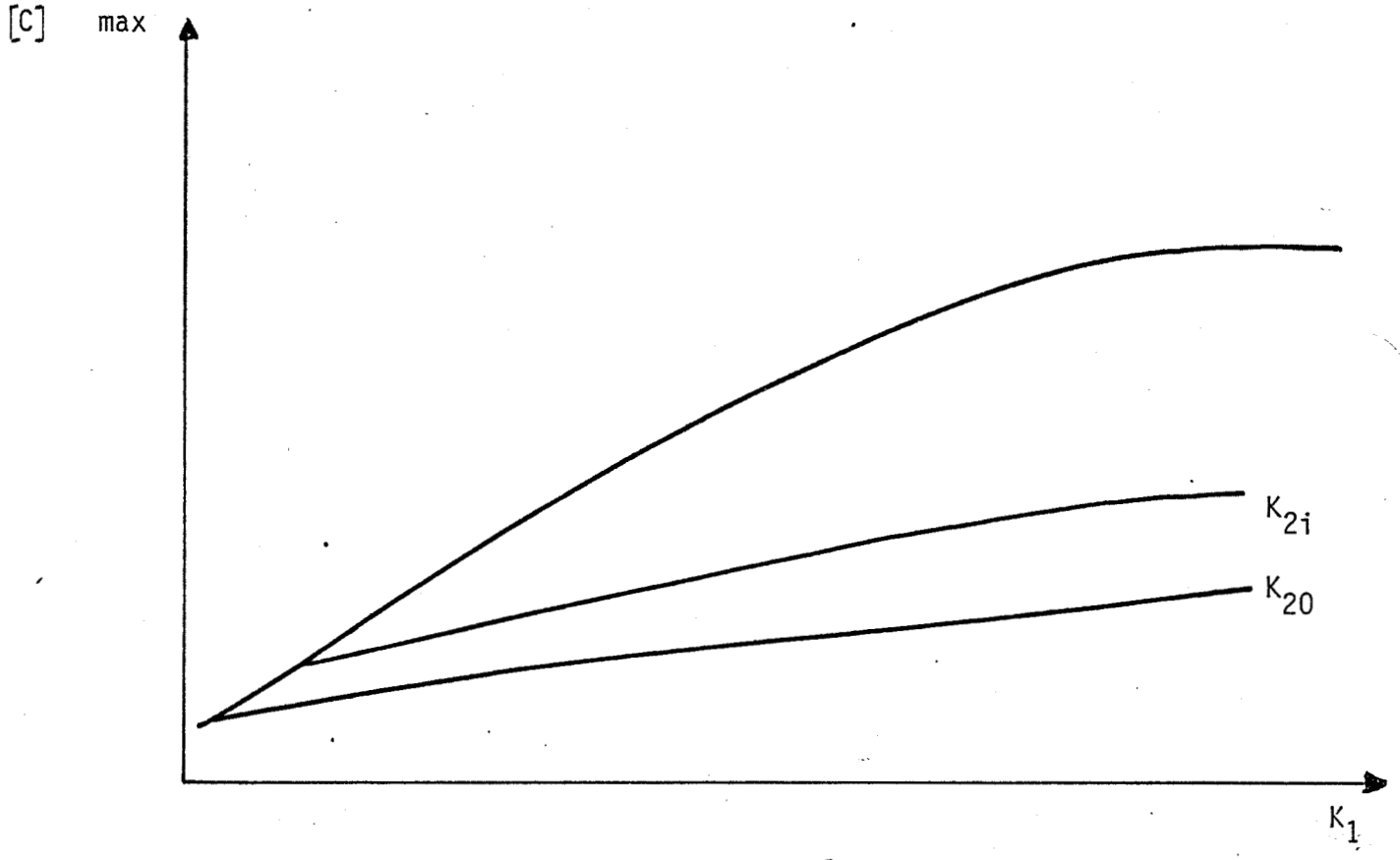
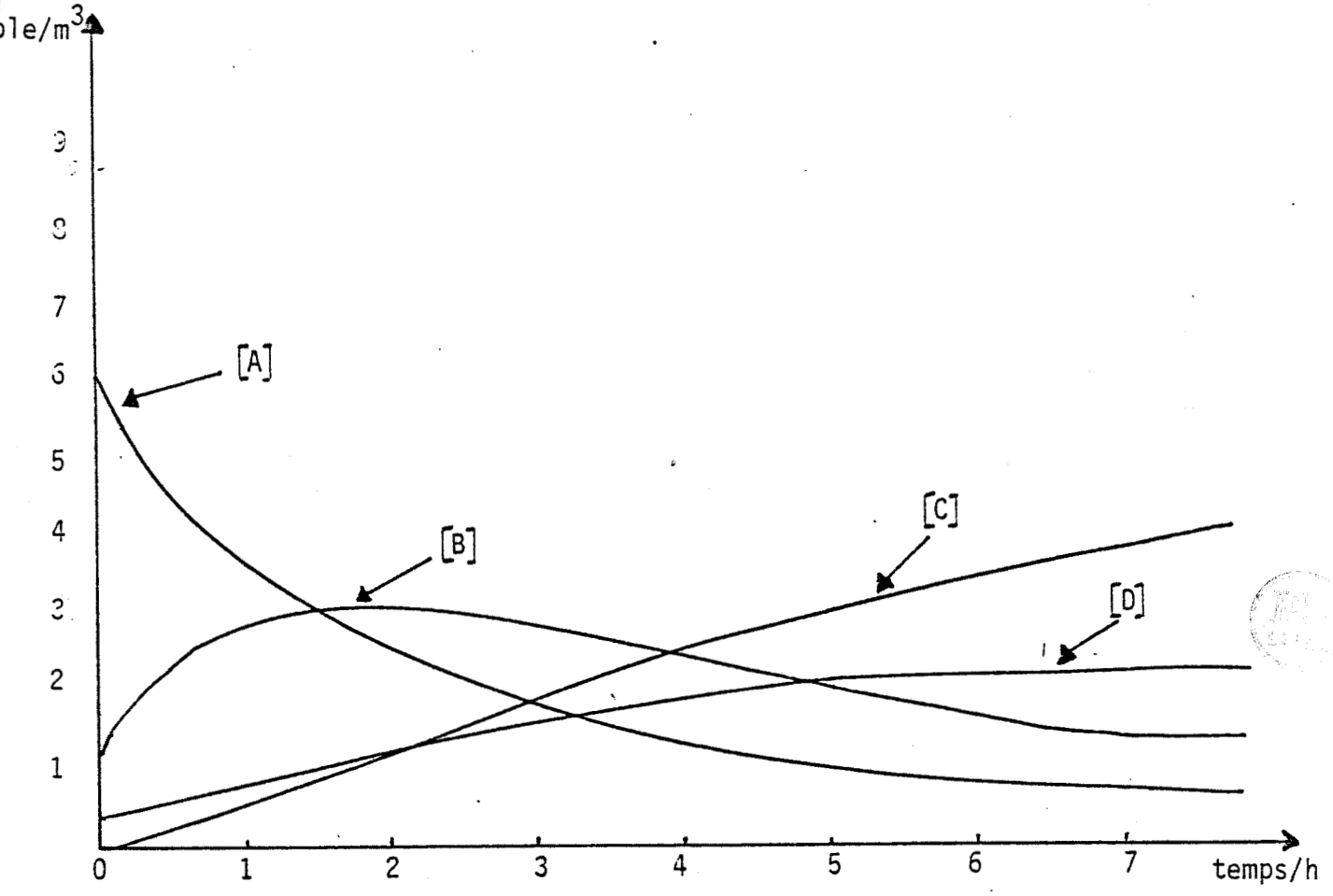


FIGURE 21b COURBES $[C]\ max = f_{k_2}(k_1)$

I.8 RESULTATS DE LA SIMULATION

La figure 22 illustre l'évolution des concentrations en fonction du temps pour des valeurs de K_1, K_2, K_3, K_4 et K_5 fixes.



EVOLUTION DES CONCENTRATIONS |A| ,
|B|,|C|,|D| EN FONCTION DU TEMPS

FIGURE 22

II RESOLUTION DU PROBLEME DE L'EJECTION DU PILOTE D'UN AVION

II.1. DESCRIPTION DU PROBLEME

La technique de sauvetage la plus répandue dans le domaine de l'aviation militaire repose sur le principe d'éjection du siège lorsque le pilote se trouve dans une situation de perte totale du contrôle de l'avion. La formulation du problème doit tenir compte de plusieurs points de vue dont les trois principaux concernent :

- les phases de vol de l'avion (intégration de l'altitude, des paramètres et du niveau de vol "FL").

- l'environnement atmosphérique (intégration de la trainée.... atmosphérique).

- le système siège-pilote (intégration de la masse et de la vitesse d'éjection).

La formulation globale du problème revient à considérer l'ensemble "siège-avion" en mouvement relatif dans un repère lié à l'avion de manière à définir les équations qui régissent le mouvement du siège afin d'en faire une résolution paramétrique par simulation dans l'objectif d'éviter toute possibilité de collision entre le siège et la queue de l'avion.

La figure N° 23 illustre les données du problème.

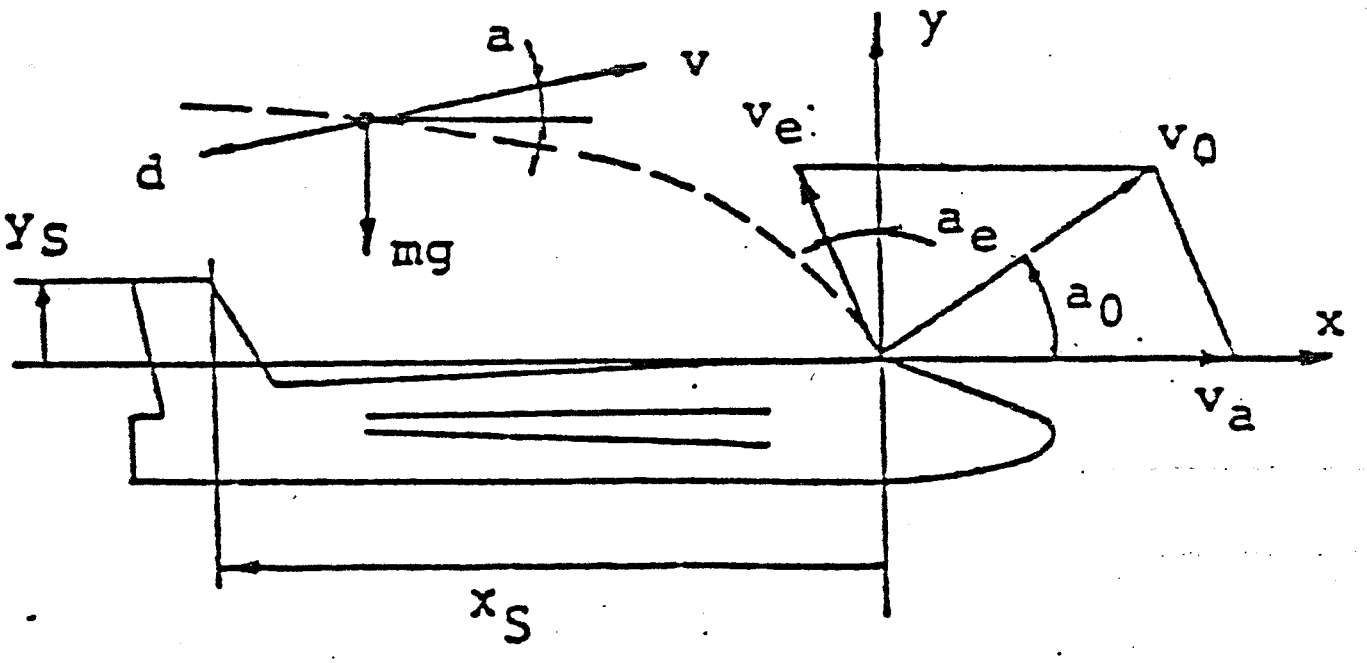


FIGURE 23

- Va = vitesse de vol supposée constante
- Ve = vitesse d'éjection du siège par rapport à l'avion
- ae = angle d'éjection par rapport à l'avion
- V = vitesse absolue du siège
- a = angle absolue du siège
- V_0, a_0 = veleurs initiales de la vitesse et de l'angle absolues
- d = trainée atmosphérique
- ha = niveau de vol supposé constant
- m = masse totale du siège et du pilote
- x_s,y_s = coordonnées du stabilisateur vertical de l'avion

Il est à noter que toutes les variables et constantes sont exprimées dans le système d'unité M.K.S.

II.2. SIMULATION DE L'EVOLUTION TEMPORELLE DE LA TRAJECTOIRE

Outre les hypothèses relatives à la vitesse "Va" et au niveau du vol "ha", nous considérons que l'avion est dans une phase de vol en palier ; dans ces conditions, le mouvement du siège ne dépend que de l'action des seules forces de la pesanteur et de la trainée atmosphérique dont l'intensité est proportionnelle au niveau "ha".

L'étude mathématique des équations qui régissent l'évolution temporelle de la trajectoire aboutit au système différentiel suivant :

$$\frac{dx}{dt} = V \cos a - Va$$

$$\frac{dy}{dt} = V \sin a \qquad (1)$$

$$\frac{dV}{dt} = -(d/m + g \sin a)$$

$$\frac{da}{dt} = -\frac{g \cos a}{V}$$

Où une première condition initiale est donnée par le système.

$$x_0 = 0$$

$$y_0 = 0$$

$$V_0 = \sqrt{(Va - Ve \sin ae)^2 + (Ve \cos ae)^2} \qquad (2)$$

$$a_0 = \operatorname{arctg} \frac{Ve \cos ae}{Va - Ve \sin ae}$$

Le réseau de blocs descriptif du modèle mathématique est précisé par la figure N° 24.

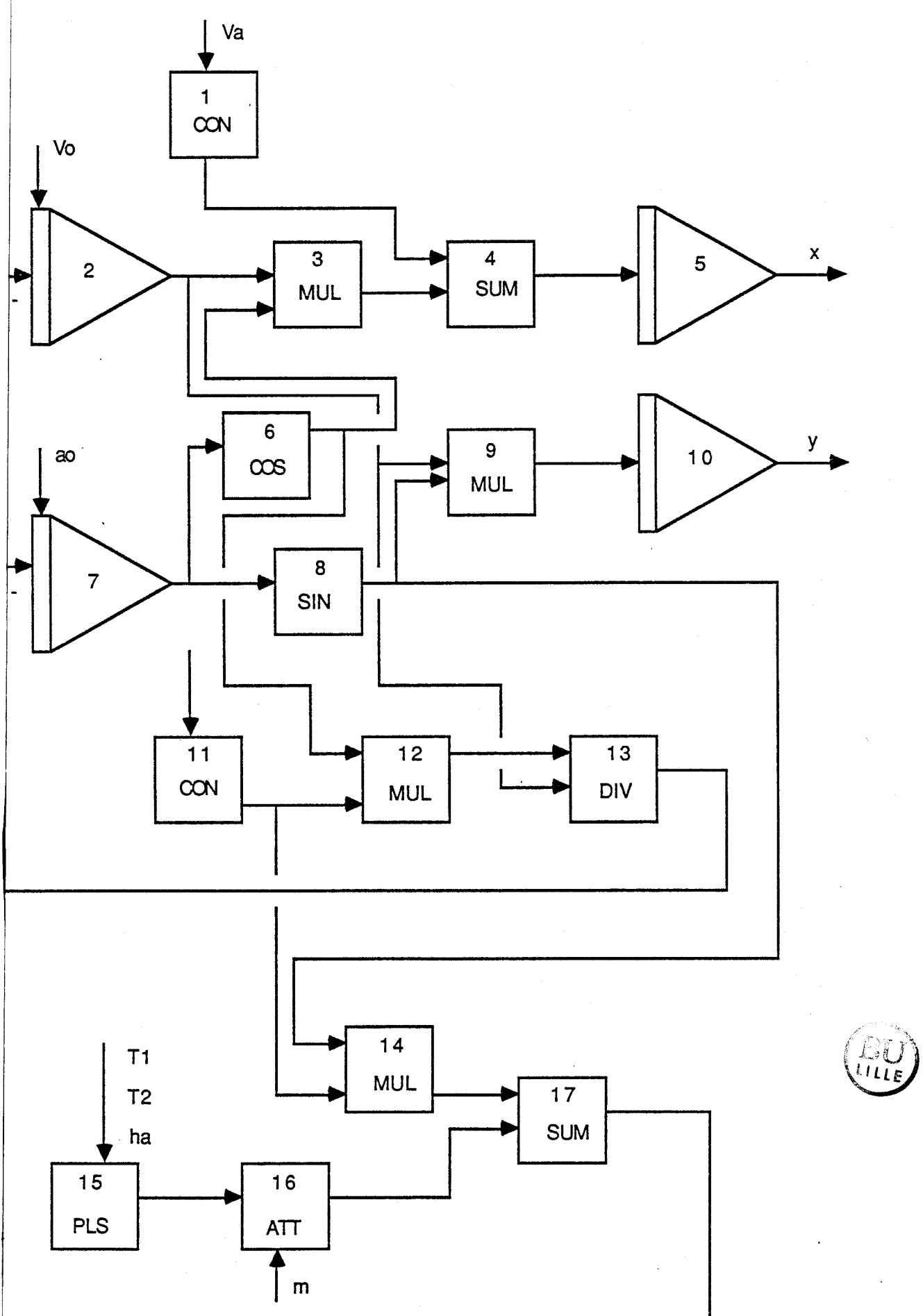


FIGURE 24 : RESEAU DU MODELE

II.2.1. CHOIX DU MODE DE RESOLUTION

Avant de décrire les fonctions à réaliser par les programmes hybrides, il est nécessaire de préciser l'objectif de la simulation envisagé ; en effet la solution du problème étudié est en fonction au moins des paramètres ha, Va, ae et Ve ; aussi plusieurs simulations par variations successives de ces paramètres sont nécessaires à la recherche du maxima de la position x,y du siège telle que :

$$x(t) = X_s$$

et

$$y(t) = Y_s$$

correspondent à la date "t" du passage du siège par les coordonnées du stabilisateur de l'avion.

Pour être adapté à la logique de travail sous le logiciel, nous supposerons un générateur d'impulsion qui délivre des impulsions d'amplitude proportionnelle à la trainée atmosphérique. Ce qui réduit le nombre de paramètres déterministes de la "date" et donc de la solution à deux variables, soient la vitesse Va de l'avion et la vitesse Ve d'éjection du siège.

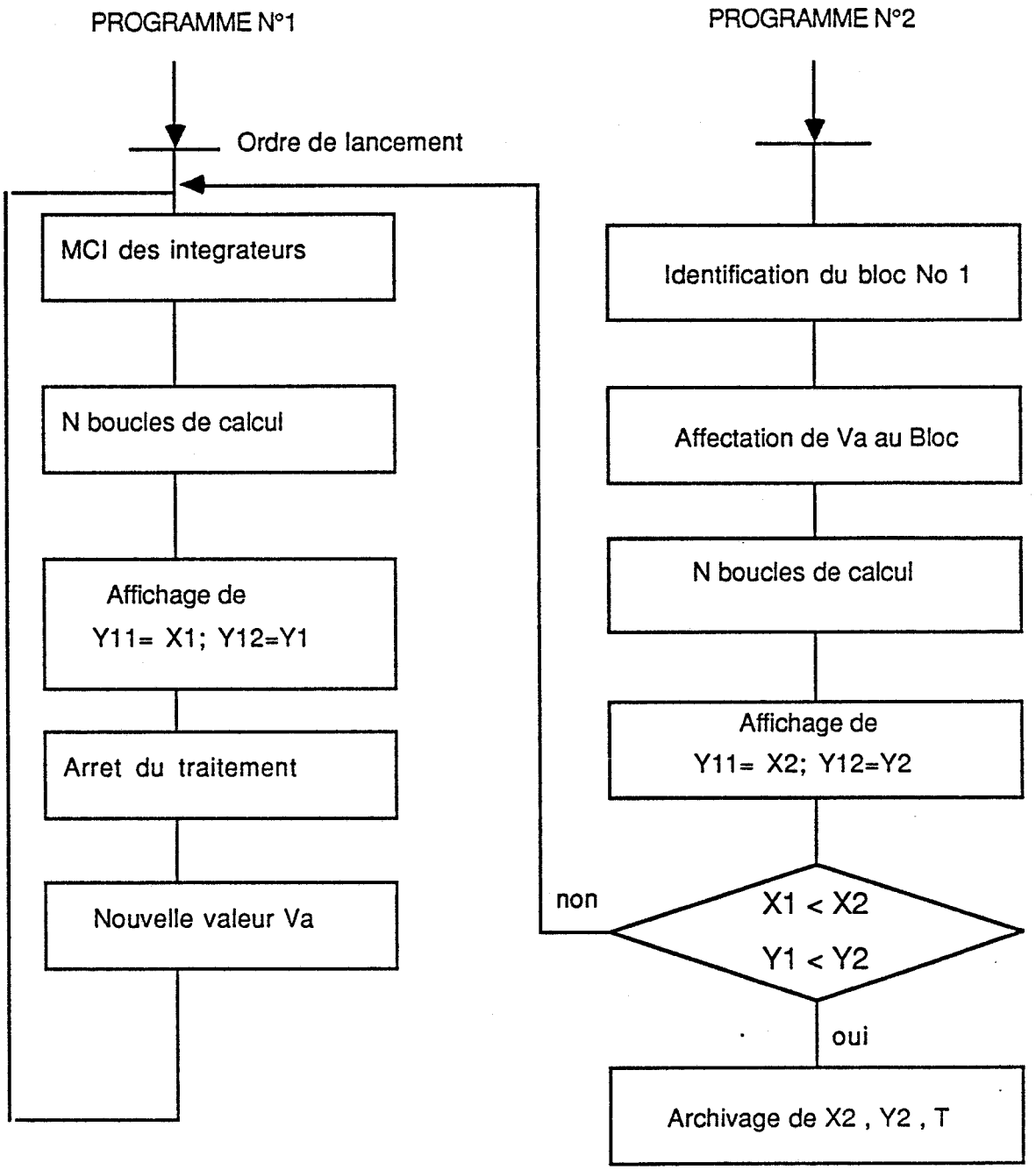
II.2.1.1. PROGRAMMES HYBRIDES

Dans cet exemple deux fonctions sont à réaliser par programmation.

En premier lieu, une fonction dans laquelle les programmes hybrides assurent d'une part le lancement de la résolution du réseau de blocs et d'autre part l'arrêt de celle-ci au gré de l'utilisateur.

Dans un second temps, il convient de procéder à une variation périodique des composantes Va et Ve et de relancer la résolution jusqu'à l'obtention du maximum de x et y puis relever le temps correspondant.

L'organigramme de principe est illustré par la figure suivante :


PROGRAMME N°1
Ordre de lancement
MCI des integrateurs
N boucles de calcul
Affichage de
Y11= X1; Y12=Y1
Arret du traitement
Nouvelle valeur Va
PROGRAMME N°2
Identification du bloc No 1
Affectation de Va au Bloc
N boucles de calcul
Affichage de
Y11= X2; Y12=Y2
non
X1 < X2
Y1 < Y2
oui
Archivage de X2 , Y2 , T

II.2.2. PHASE D'EDITION POUR LA SAISIE DU RESEAU

Tout d'abord, il convient de faire une hypothèse conformément aux données du problème en vue d'une application numérique ; en effet, les conditions initiales sous forme d'expressions arithmétiques peuvent être réduites à des constantes si l'on considère que la vitesse de vol Va est au moins 10 fois supérieure à celle d'éjection du siège.

Donc $Ve << Va \Rightarrow$

$$V_0 = \sqrt{(Va - Ve \sin ae)^2 + (Ve \cos ae)^2} \neq \sqrt{Va^2} = Va$$

$$a_0 = - \operatorname{arctg} \frac{Ve \cos ae}{Va - Ve \sin ae} = 0$$

D'où nous proposons les valeurs numériques suivantes :

$g = 9{,}81$
$m = 140$
$Va = 150$
$Ve = 15$

$V_0 = Va$
$x_0 = 0$
$y_0 = 0$
$a_0 = 0$

II.2.2.1. STRUCTURE DU RESEAU

```
..
1, CON
2, INT, -17
3, MUL, 2, 6
4, SUM, -1, 3
5, INT, -4
6, COS, 7
7, INT, - 13
8, SIN, 7
9, MUL, 8, 2
10, INT, -9
11, CON
12, MUL, 11, 6
13, DIV, 12, 2
14, MUL, 8, 11
15, PLS
16, ATT, 15
17, SUM, 16, 14
```

II.2.2.2. PARAMETRES DU RESEAU

```
 1, 150
 2, 150
11, 9.81
15, 1, 200, 5
16, 140
```

II.2.2.3. BLOCS "RESULTATS" DE LA SIMULATION

X_{11} : 0, 0, 200

Y_{11} : 5, 0, 50

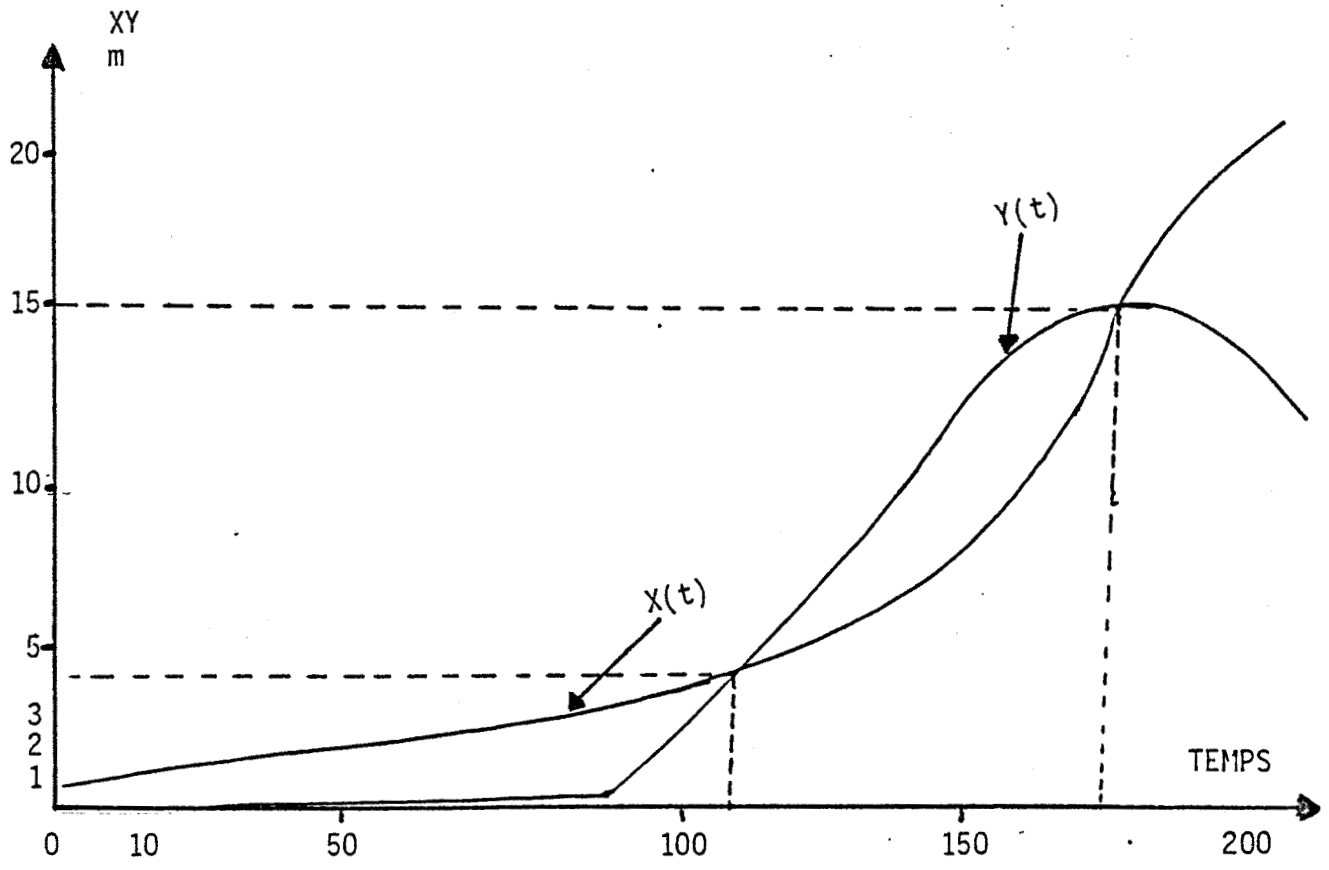
Y_{12} :10, 0, 20

II.2.2.4. DUREE DE LA SIMULATION

0.5, 200

II.2.2.5 RESULTATS

La figure 25 représente l'évolution de la trajectoire du siège pour une vitesse de vol Va égale à 150 pieds/s.



EVOLUTION DE LA TRAJECTOIRE DU SIEGE
ET DATE DE PASSAGE PAR LES COORDONNEES
STABILITRICES

X_s = 17m
Y_s = 4,2m
T = 174s

FIGURE 25

III. RESOLUTION D'UN PROBLEME REGI PAR UN MODE DU TYPE A "CONSTANTES REPARTIES"

Dans cet exemple, nous étudions un processus thermique très simple dont le modèle mathématique est régi par une équation aux dérivées partielles paraboliques de la forme :

$$N_1 \quad \frac{\partial^2 \phi}{\partial x^2} + \alpha \frac{\partial \phi}{\partial x} = \beta \frac{\partial \phi}{\partial t}$$

où $\alpha, \beta, x, \phi \in R$ et $t \in T$ tq $T \in [o, \infty[$

L'équation comporte 2 variables indépendantes soient :

- une variable d'espace x

et

- une variable temps t

La résolution d'un système de cette nature peut être envisagée selon 3 méthodes différentes comme suit :

- a) Dans des cas très particuliers, une résolution formelle en continue peut être envisagée en utilisant la transformée de la place à 2 variables ; il est également possible de construire un modèle electrique à 2 dimensions constitué de résistances et condensateurs ; ceci implique dans ce cas le choix de capacités très faibles associées à une échelle temps beaucoup trop rapide. Cette technique est donc difficile à généraliser.

- b) La deuxième méthode possible est une discrétisation de la variable d'espace x ; dans ce cas on obtient un système d'équations différentielles linéaires ou non linéaires selon la nature des différents α et β. L'obtention d'un tel modèle ne peut être valable dans le fait que par le biais d'une approximation car en discrétisant à la fois x et t, il est possible de définir un modèle récurrent et d'envisager ainsi une résolution purement numérique.

- c) Enfin si l'on cherche à discrétiser uniquement la variable temps, alors on peut obtenir un modèle décrit par un système différentiel par rapport à la variable x ; dans ce cas, on envisagera une solution par simulation numérique répétitive dans laquelle l'évolution dynamique de rang i du processus ne dépendra que des informations de rang i-1.

C'est donc sur la base de ces 2 dernières méthodes que nous proposons de résoudre le système décrit par la relation N_1 grâce à une gestion numérique par les sous programmes hybrides du simulateur.

III.1 DESCRIPTION DU PROBLEME ETUDIE

Pour illustrer notre choix nous proposons de résoudre un problème de conduction thermique mieux connu sous le nom de "choc thermique". /29/

Il s'agit en effet d'étudier par simulation la dynamique de la température d'un mur d'épaisseur constante soumis à un échelon de température à partir d'un état initial supposé en équilibre.

L'application de la loi de Fournier relative à la conductibilité thermique traduit l'évolution dynamique de la température d'une masse suivant un modèle de la forme :

$$N_2 \qquad \text{Div} \left[\lambda \, , \, \text{Grad.} \, \theta \right] = \rho C \frac{\partial \theta}{\partial t} \quad \text{où}$$

- ρ est une constante représentant la masse volumique du mur
- C la chaleur massique du mur
- λ le coefficient de conductibilité qui dans le cas général est une fonction de la température.

A titre d'hypothèse simplificatrice, le mur sera considéré de dimension infinie de façon à négliger les effets limites et à n'étudier le problème que dans la seule dimension relative à l'épaisseur du mur. Dans la même optique, il est possible d'étudier le problème en envisageant 2 modes de la dynamique de la température selon que le coefficient λ est :

- supposé constant, ou bien :
- considéré dans sa forme générale de fonction non constante de la température.

Dans ces conditions, l'équation de la relation N_2 peut être écrite sous forme d'un système simplifié :

* λ est une fonction affine de la température suivant l'expression :

$$\lambda = a\,\theta + b$$

Dans ce cas on obtient le système =

$$\lambda = a\,\theta + b$$

$$N_3 \qquad b\,\frac{\partial^2\theta}{\partial x^2} + a\,\frac{\partial\theta^2}{\partial x} = \rho c\,\frac{\partial\theta}{\partial t}$$

*λ est une constante

Dans ce cas le système devient :

$$\lambda = b$$

$$N_4 \qquad b\,\frac{\partial^2\theta}{\partial x^2} = \rho c\,\frac{\partial\theta}{\partial t}$$

III.2 CHOIX D'UN MODE DE RESOLUTION

Dans l'optique de la méthode de résolution proposée, la discrétisation de la variable temps dans l'un ou l'autre système d'équations des relations (N_3, N_4) permet l'écriture de chacun des systèmes sous deux formes suivant que l'on procède à une discrétisation dans le sens croissant ou rétrograde de la variable.

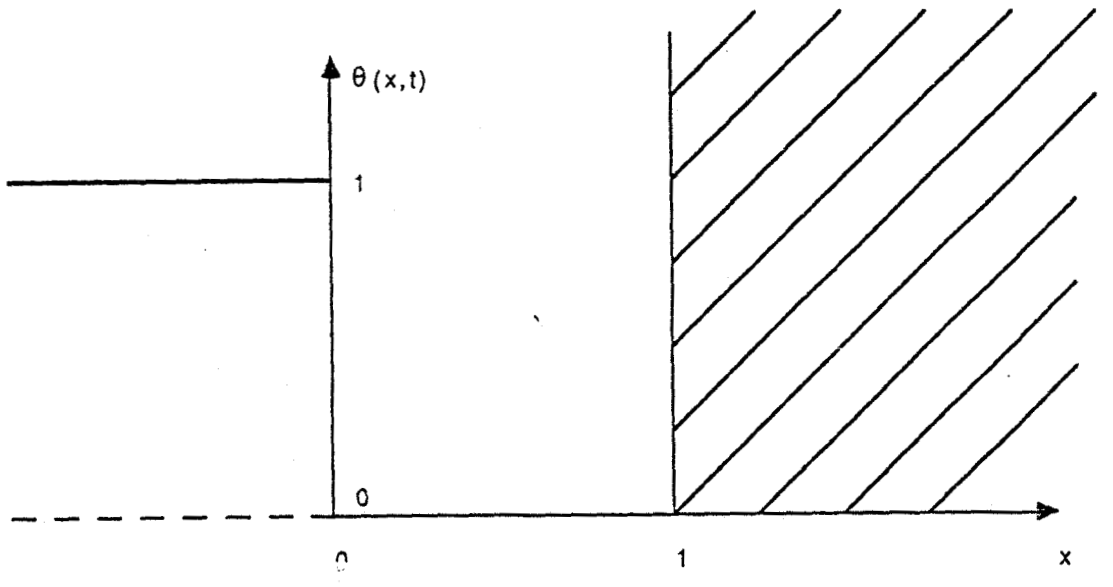
L'application de cette méthode à l'équation de la relation N_4 conduit tout naturellement à une représentation sous sa forme différentielle suivant x telle que :

N_5

$$1 \quad \frac{\lambda}{\rho c}\,\frac{d^2\theta_j}{dx^2} = \frac{\theta_j - \theta_j - 1}{\Delta t} \qquad \text{si t croissant}$$

$$2 \quad \frac{\lambda}{\rho c}\,\frac{d^2\theta_j+1}{dx^2} = -\,\frac{\theta_j+1 - \theta_j}{\Delta t} \qquad \text{si t rétrograde}$$

Dans ces conditions θ représente la température en variables réduites évoluant dans l'intervalle $|0,1|$ pour tout $x \in |0, 1|$. Nous en déduisons les conditions aux limites variables à tout instant ti donné suivant le graphe ci-dessous :



- Etat initial :

$$\theta(0,0) = 0 \; ; \quad \theta(1,0) = 0 \; ; \quad \theta(x,0) = 0$$

- Etat limite pour $t \in [0^+, \infty [$:

  - sur la surface intérieur du mur : $\theta(0,t) = 1$
  - à l'extérieur du mur : $\theta(1,t) = 0$

Etant donné que la stabilité du modèle correspond à celle du processus ligne, le choix de résolution par simulation de l'une ou l'autre équation de la relation N_5 peut être déduit d'une étude de stabilité du système décrit par l'équation aux dérivées partielles de la relation N_1.

Le système physique ici considéré comporte un régime limite stable correspondant à l'équilibre thermique s'établissant par l'intermédiaire du mur ; aussi il est évident que seule la résolution suivant une discrétisation dans le sens croissant de la variable t permet de tenir compte de ces conditions limites et de faire tendre nécessairement le comportement dynamique du modèle vers un régime stable.

Aussi nous proposons de résoudre le modèle à partir de l'équation 1 de la relation N_5 suivant 2 hypothèses :

- D'abord la valeur de la condition initiale sur la dérivée $\frac{d\theta}{dx}$ est préalablement nulle.

- La valeur du pas d'intégration ou d'échantillonnage suivant la variable temps est choisie très faible ; ce qui permet d'assimiler l'évolution de la variable x à celle de la variable temps afin d'adapter la résolution du modèle aux conditions de travail sous le logiciel. D'où l'introduction d'une équation supplémentaire traduisant la relation de proportionnalité.

Dans ces conditions, nous obtenons le système suivant :

$$t = \alpha x$$

$$N_6 \qquad \frac{\lambda}{\rho C} \frac{d^2\theta}{dx^2} = \frac{d\theta}{dt} = \alpha \frac{d\theta}{dx}$$

L'équation du modèle devient :

$$N_7 \qquad \frac{\lambda}{\rho C} \frac{d^2\theta}{dx^2} = \alpha \frac{d\theta}{dx} \qquad \text{avec } [\dot{\theta}(0, x_i)] = 0$$

La solution de $\theta(x,ti)$: $\theta(x, xi)$ obtenue réinjectée comme condition initiale au pas de résolution xi+1 correspondant à n boucles d'intégration jusqu'à l'obtention de la contrainte finale $\theta(1, xi)$ qui permet de valider ou de rejeter la solution.

En définitive la recherche de la valeur initiale de la dérivée sur chaque pas d'intégration suivant est suivie d'une phase d'enregistrement, de réinitialisation par affectation de la valeur enregistrée et enfin de relancement pour le pas suivant.

L'allure à titre indicatif des résolutions peut être représentée dans l'espace (θ, x, t) par la figure N° 26.

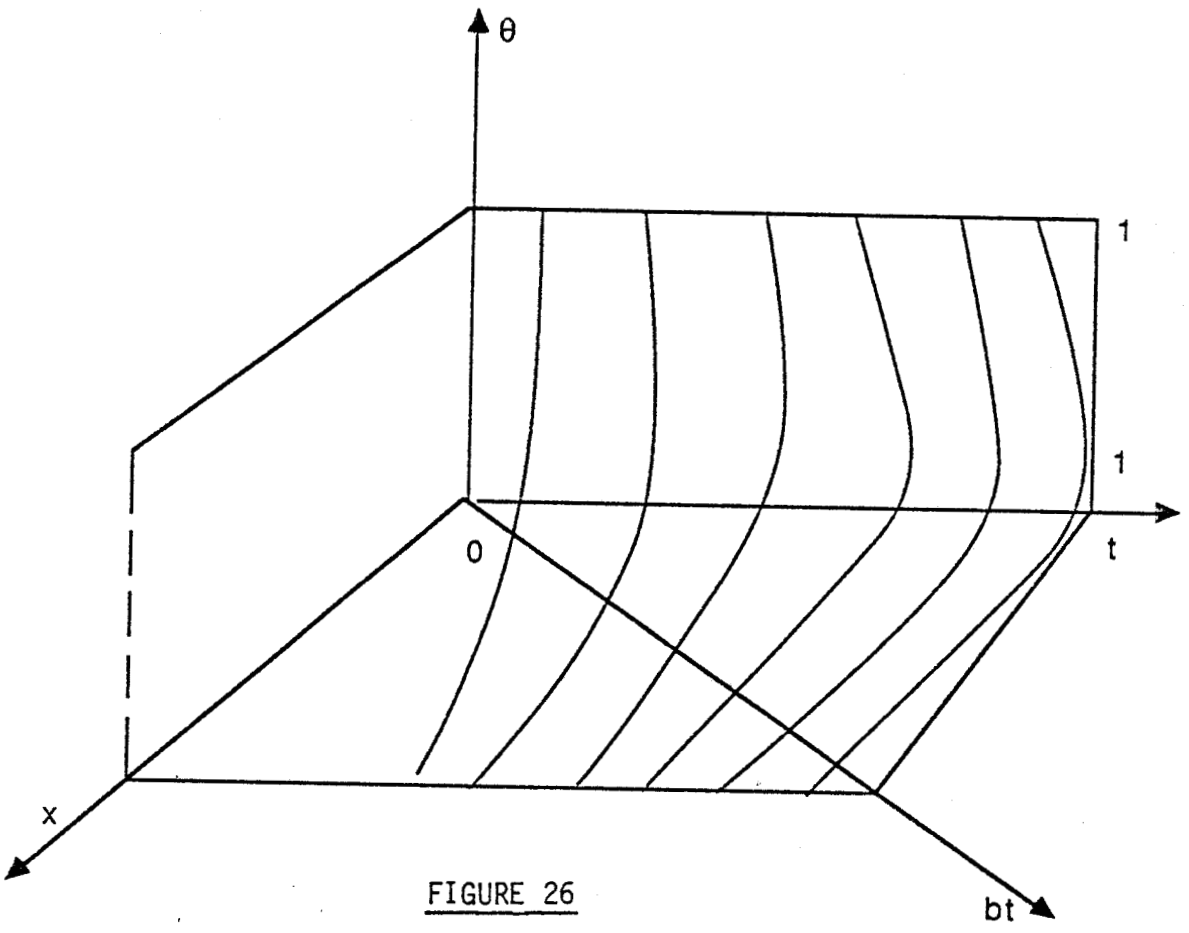


FIGURE 26

III.3 PROGRAMMATION HYBRIDE

La programmation hybride de résolution assure 2 fonctions essentielles dont la première consiste d'une part à gérer la recherche de la condition initiale et d'autre part à enregistrer les valeurs de la température θ dans un fichier annexe.

La seconde fonction permet le passage d'un pas de n intégrations au pas suivant après modification de la valeur de $\dot{\theta}$ à partir de l'enregistrement $\theta(x, xi - 1)$ du pas précédent.

Donc toute la partie de programme se résume à un procédé de communication d'informations (enregistrements de θ) d'un pas au pas suivant à travers 2 tables de résultats [NRPL (n,y)] qui jouent le rôle d'une bascule.

L'organigramme de principe est illustré par la figure N° 27.

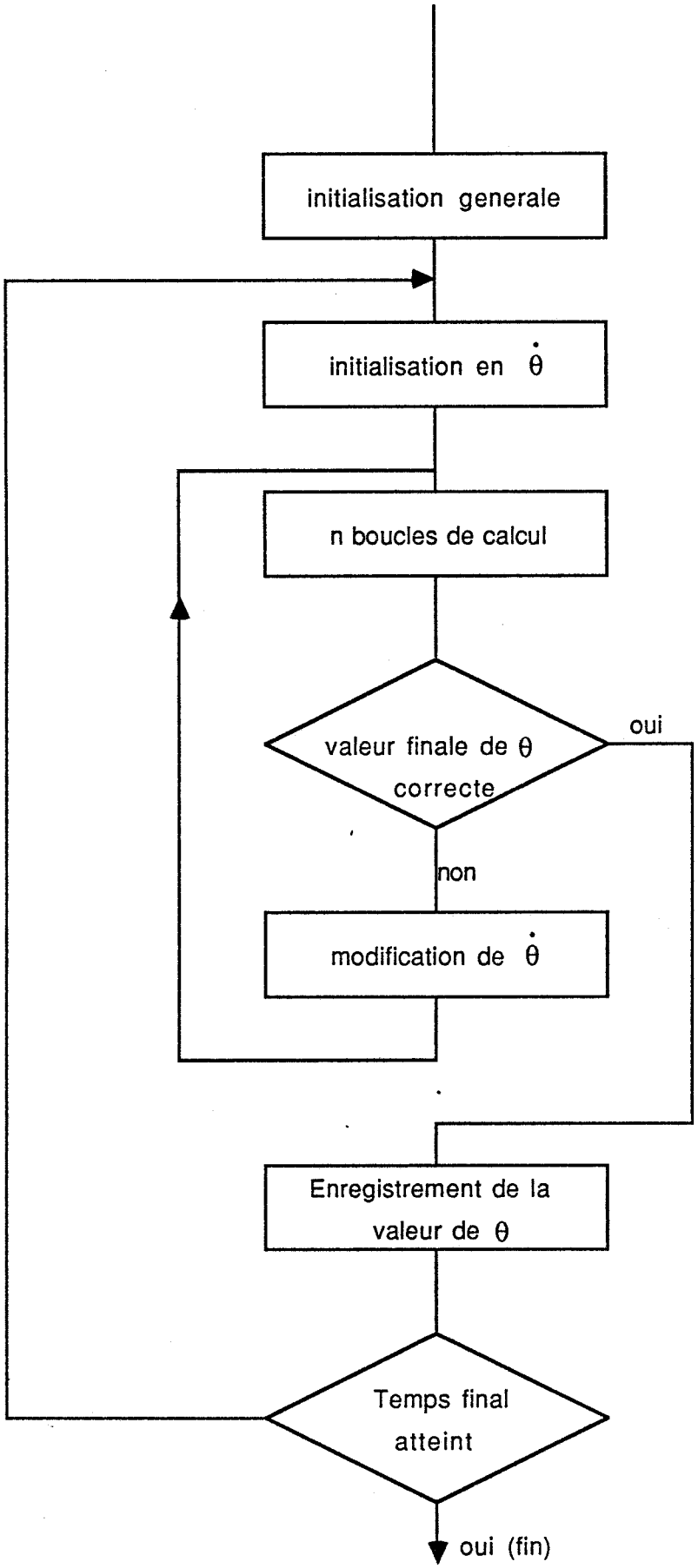


FIGURE 27

III.4 RESEAU DE BLOCS DESCRIPTIFS DU CABLAGE DU MODELE

Le modèle étant décrit par l'équation de la relation N_7, nous proposons les valeurs numériques suivantes en vue de l'édition de son réseau conformément à la logique de travail sous le logiciel

$\alpha = 1$, $\rho = 1$, $c = 1.05\ 10^3$, $\lambda = 1$ avec
$\theta(0,0) = 0.01$

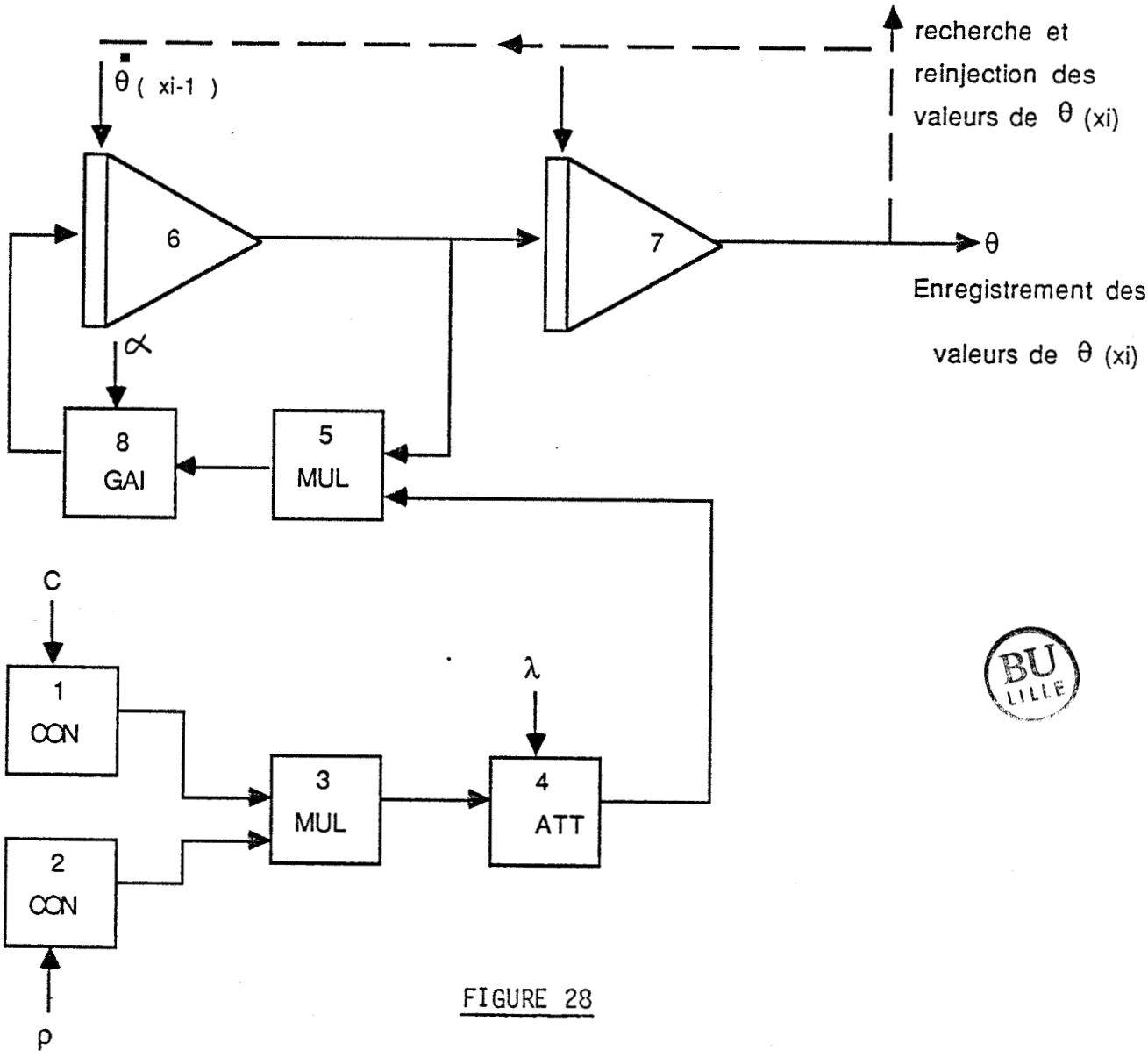


FIGURE 28

III.5 EDITION DU RESEAU

a) Structure

1, CON
2, CON
3, MUL, 1, 2
4, ATT, 3
5, MUL, 4, 6
6, INT, 5
7, INT, 6
8, GAI, 5

b) Paramètres

1, 1,05 10^3
2, 1
4, 1
6, 0.01
8, 1

c) Blocs de sortie des résultats

X_{11} : 0, 0, 100
Y_{11} : 6, 0.01, 1
Y_{12} : 7, 0, 1

d) Durée d'observation

0.01 , 100

III.6 EXTENSION DE LA RESOLUTION AU CAS OU λ EST NON CONSTANT

Comme nous l'avons évoqué dans le 1er paragraphe, ce cas est applicable soit à un mur non homogène dans lequel la conductibilité est une fonction affine de la température, soit à un mur composé de plusieurs conductibilités thermiques différentes.

Dans ces conditions, l'expression de la conductibilité de la forme :

$$\lambda = a\theta + b$$

L'équation aux dérivées partielles issue de la loi de Fournier contient un terme supplémentaire et s'exprime sous la forme

$$N_1' \qquad \frac{b\,\partial^2\theta}{\partial x^2} + a\left(\frac{\partial\theta}{\partial x}\right)^2 = \rho c\,\frac{\partial\theta}{\partial t}$$

La discrétisation suivant la variable temps dans le sens croissant permet d'écrire :

$$N_2' \qquad \frac{b}{\rho c}\,\frac{d^2\theta_i}{dx^2} + \frac{a}{\rho c}\left(\frac{d\theta i}{dx}\right)^2 = \frac{\theta_i - \theta_i{}^{-1}}{\Delta t}$$

Si nous adoptons une résolution suivant les mêmes hypothèses et contraintes aux limites que dans le cas précédent, on aboutit à une représentation du modèle par le système suivant :

$$t = \alpha x$$

$$N_3' \qquad \frac{b}{\rho c}\,\frac{d^2\theta}{dx^2} + \frac{a}{\rho c}\left(\frac{d\theta}{dx}\right)^2 = \frac{d\theta}{dt}$$

D'où la description du modèle mathématique du système par l'équation différentielle :

$$N_4' \qquad \frac{b}{\rho c}\,\frac{d^2\theta}{dx^2} + \frac{a}{\rho c}\left(\frac{d\theta}{dx}\right)^2 = \alpha\,\frac{d\theta}{dx}$$

La modification de l'équation du modèle par rapport au cas où λ est une constante ne concerne en fait que l'introduction du terme supplémentaire $\left(\frac{d\theta}{dx}\right)^2$ et la représentation des blocs descriptifs du cablage de son réseau.

Dans ces conditions le programme hybride de la simulation reste inchangé relativement aux modifications de la valeur de la condition initiale sur la dérivée θ; cependant il intègre une variation supplémentaire relative à celle de la conductibilité thermique λ grâce à la variation du coefficient directeur "a". Cette variation est indépendante de la première et peut être réalisée de façon séquentielle grâce aux sous programmes hybrides de passage automatique de paramètres, traduisant de ce fait une possibilité de résolution pseudo-parallèle par le simulateur.

III.7 REPRESENTATION DU RESEAU DE BLOCS DU MODELE

Le réseau de blocs est déduit de l'équation de la relation N_4' mise sous la forme :

$$\frac{d^2\theta}{dx^2} = \frac{1}{b}\left[\alpha \rho c \frac{d\theta}{dx} - a\left(\frac{d\theta}{dx}\right)^2\right]$$

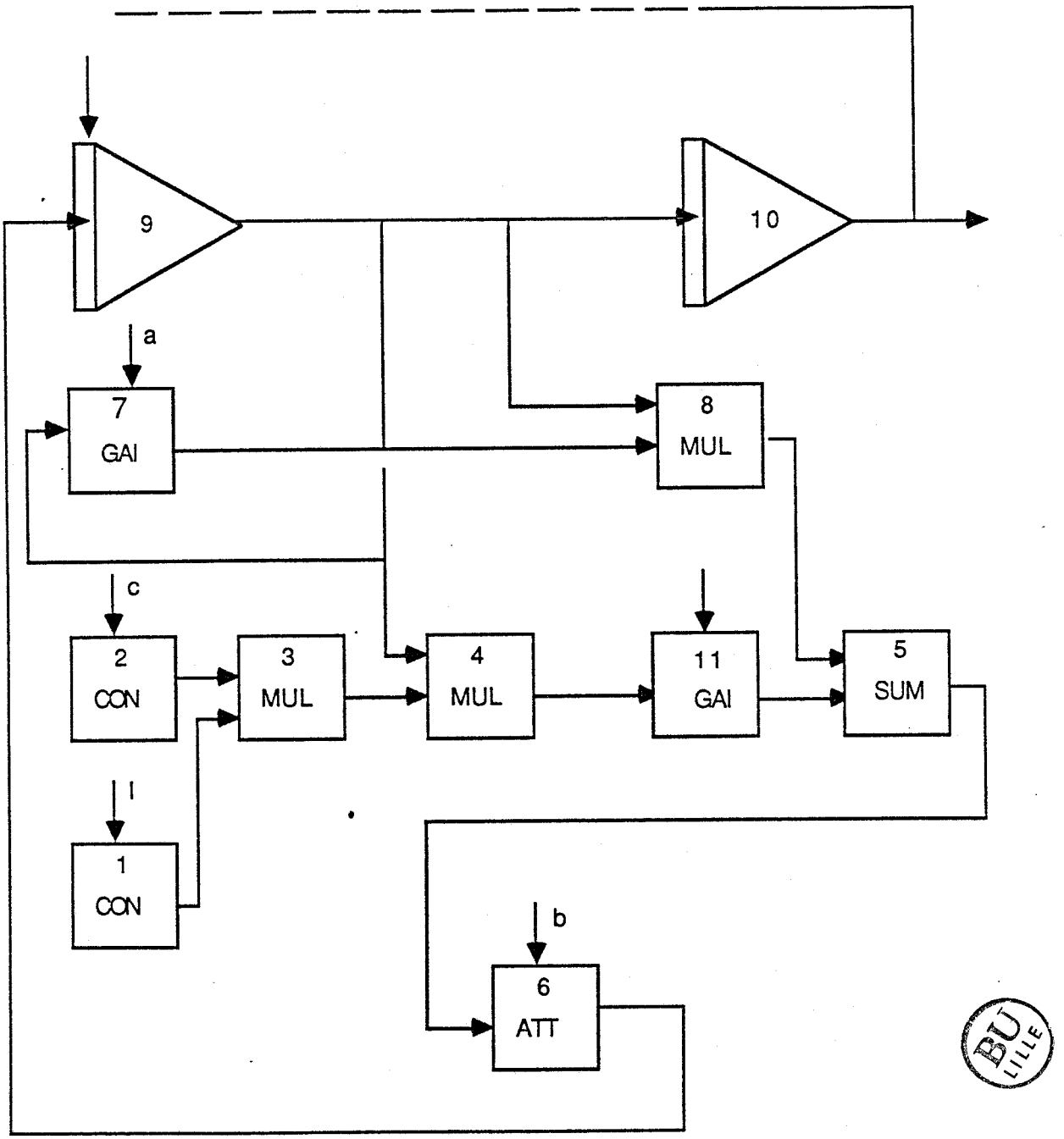


FIGURE 29

Pour la saisie du réseau par le simulateur, les valeurs numériques des paramètres "a" et "α" sont initialisées "1".

a) structure du réseau

```
1,CON
2,CON
3,MUL,1,2
4,MUL,3,9
5,SUM,4,8
6,ATT,5
7,GAI,9
8,MUL,9,7
9,INT,6
10,INT,9
11,GAI,4
```

b) Paramètres du réseau

1, 1
2, 1.5 10^3
6, 1
7, 1
9, 0.01
11, 1

c) Blocs de sortie des résultats

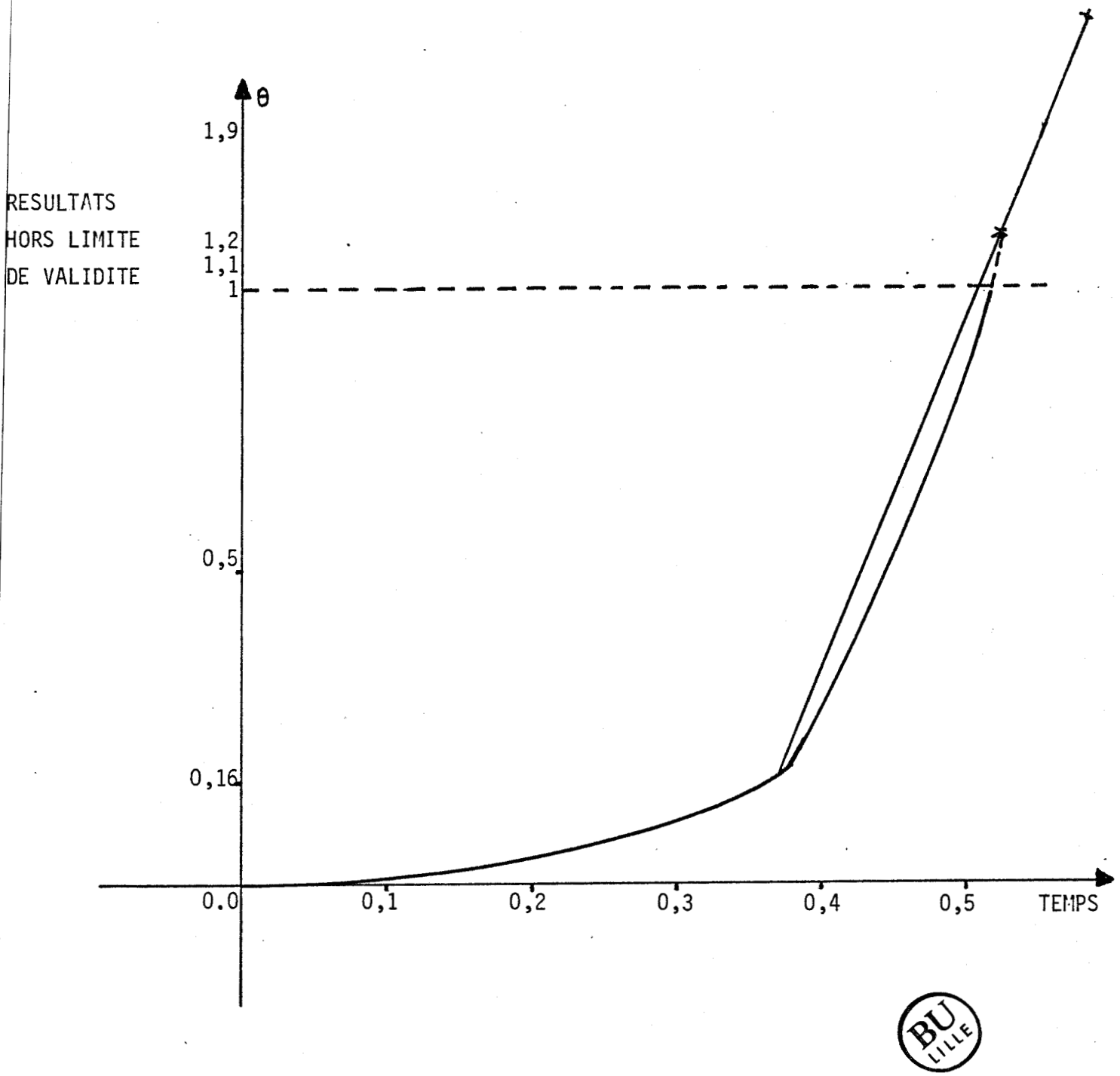
X_{11} : 0, 0, 100
Y_{11} : 9, 0.01, 1
Y_{12} : 10, 0, 1

d) Durée d'observation

0.01, 100

III.8 RESULTATS

La figure 30 représente l'évolution de la température à l'intérieur du mûr avec quelques points représentatifs hors des limites de variabions définies au cours de l'étude de la stabilité du système d'équations. Ceci est dû aux erreurs de discrétisation de la variable x à travers l'équation de proportionnalité supplémentaire $t = \alpha x$.



EVOLUTION DE LA TEMPERATURE A
L'INTERIEUR DU MUR

FIGURE 30

CONCLUSION GENERALE

CONCLUSION GENERALE

Le logiciel présenté dans ce mémoire constitue un outil de simulation adapté à l'étude des processus industriels dont les modèles mathématiques sont décrits par un système d'équations différentielles linéaires ou non.

Dans le cas d'une étude simultanée de plusieurs processus, ce logiciel permet une saisie selon une structure "banalisée", paramètrée par des variables dimensions et par des valeurs numériques affectées aux variables relatives à chaque processus élémentaire considéré. Aussi, pour des procédés industriels complexes, il permet une résolution simple en adoptant pour le modèle de référence une dynamique comparable à celle qui caractérise le procédé lui-même. Le simulateur présente de ce fait une facilité de mise en oeuvre directe des méthodes de saisie, d'analyse et de synthèse des systèmes de modèles sans tomber sous le coup de restrictions généralement dues aux langages utilisés.

De par sa nature hybride, le logiciel permet de traiter simultanément des opérations numériques et analogiques en temps réel de façon quasi parallèle ; le contrôle du processus étant à la fois gérē par des sous-programmes de réglage et par des interventions programmables par l'utilisateur.

Dans ce travail, nous avons étendu le logiciel de base de façon à saisir simultanément plusieurs processus, à les archiver dans des "fichiers modèles" et à autoriser le pseudo-parallélisme de ces processus. Ces extensions nous ont permis de traiter des exemples complexes, tels que les problèmes relatifs au choc thermique, à la simulation de l'éjection d'un pilote d'avion et à la cinétique chimique. S'il est un autre domaine où le logiciel révèle une importance supplémentaire, c'est celui de l'enseignement ; en effet, sa transporta-

bilité permet une implantation assez aisée sur des micros-ordinateurs ou de gros calculateurs. Il s'agit alors d'un outil didactique bien adapté à l'étude des systèmes électriques, mécaniques ou chimiques par simulation.

Cet aspect didactique est consolidé par une structure et un dialogue interne enrichi par des choix optionnels qui conduisent à mettre assez rapidement ce simulateur à la portée d'utilisateurs non spécialisés en informatique.

Au-delà des fonctionnabilités remplies par le simulateur sur le plan de l'enseignement (travaux pratiques de simulation) et de la recherche (simulation des procédés industriels), il est possible d'envisager une perspective d'évolution de ce travail afin de doter le logiciel d'un aspect complémentaire de transparence et de facilité de mise en oeuvre au cours d'un processus de simulation.

On peut, en effet, envisager de rendre plus transparente la gestion du parallélisme et des entrées-sorties. Ceci nécessite la définition d'une série de primitives et l'écriture de deux logiciels complexes :

- un système d'exploitation temps réel pour la simulation multitâches à partir des fichiers processus et d'une bibliothèque de sous-programmes existants, prenant en charge les entrées-sorties, la gestion du parallélisme, les communications entre tâches élémentaires et la supervision de l'interpréteur.

- un compilateur générant un code intermédiaire pour le programme interpréteur.

Le développement de cet ensemble peut permettre une simulation multitâches de manière totalement assistée.

ANNEXE

ANNEXE

Afin de faciliter l'exploitation du logiciel par un utilisateur non initié, nous donnons un exemple d'utilisation portant sur la saisie, l'exécution, le réglage de "N" processus à partir du programme source compilé sur VAX.

Remarque : Toutes les commandes et informations définies sous l'éditeur des tâches par l'utilisateur sont notées en caractères majuscules.

A. APPEL DE L'EDITEUR DES TACHES OU FICHIERS PROCESSUS

RUN PRIMS suivi de <RC>

Cette commande entraîne l'affichage à l'écran du message :

- Voulez-vous exécuter un fichier existant : ?

TAPEZ :

- O pour "oui" Affichage du message ?

Nom du fichier (MAXI 10 caractères) : ?

Exemple : TAPEZ :

PROCE1.SIM

<RC> Retour au mode commande

La simulation dans ce cas se fait à partir du fichier-processus nommé : PROCE1.SIM

- N pour "non" Affichage du message :

Entrer nombre de processus : ?

TAPEZ :

N suivi de <RC> avec $1 < N < 10$

La simulation se fera, dans ce cas, à partir des N processus suivant une structure "banalisée" de description des tâches.

B. DESCRIPTION DES TACHES

STRUCTURE (1)
•————————
•————————
•————————
•————————
•
•
RC

PARAMETRES(1)
•————————
•————————
•————————
•————————
•
•
RC

BLOCS-RESULTATS(1)
•————————
•————————
•————————
•————————
RC

DUREE D'OBSERVATION(1)
•————————
RC

STRUCTURE(2)
•————————
•————————
•————————
•————————
•
•
RC

PARAMETRE(2)
•————————
•————————
•————————
•————————
•
•
RC

BLOCS-RESULTATS(2)
•————————
•————————
•————————
•————————

DUREE D'OBSERVATION(2)
•————————
RC

STRUCTURE (N)
•————————
•————————
•————————
•————————
•
•

PARAMETRE (N)
•————————
•————————
•————————
•————————
•
•
RC

BLOCS-RESULTATS (N)
•————————
•————————
•————————
•————————
RC

DUREE D'OBSERVATION (N)
•————————

RC Affichage à l'écran du message :
Mode insteractif ou Automatique (I/A) ? :

TAPEZ :

- A suivi de<RC> Retour au mode commande
- I suivi de<RC> Affichage du message :
  Quel modèle voulez-vous initialiser ?:
  TAPEZ le modèle n° J de votre choix suivi de <RC>

Ceci entraîne l'initialisation du processus J et retour au mode commande.

C. PHASE D'EXECUTION DES TACHES

TAPEZ :

S suivi de <RC> Affichage du message :

QUEL PROCESSUS VOULEZ-VOUS LANCER ?

TAPEZ : n° J tel que :

1=J N) suivi de <RC>

Affichage de 10 pas de calcul et des tableaux-résultats correspondant au processus J suivi du message :

TAPER CARACTERE pour continuer (T,G ou R)

TAPEZ :

- T pour poursuivre
- G pour geler et Retour au moniteur pour procédure de réglage et traitement parallèle
- R pour enregistrer le dernier résultat dans un fichier et poursuivre la simulation.

La commande G permet de geler la tâche en cours d'exécution pour favoriser les tâches suivantes :

* le réglage du processus en cours

* le forçage

* la remise en condition initiale des intégrateurs et le relancement du processus

* le lancement du processus suivant

* le passage des paramètres d'un processus à l'autre

D. PRESENTATION DES MENUS

MS : Modification de structure
MP : Modification de paramètres
MT : Modification de la durée et du pas de calcul
V : Visualisation du contenu d'un bloc résultat
F : Entrer en mode forçage
S : Début de la simulation
P : Poursuite de la simulation
L : Affichage du listing des tâches
H : Affichage des menus
DF : Sauvegarde des N processus dans N fichiers nominatifs
EX : Sortie du simulateur et retour sous le moniteur VAX

BIBLIOGRAPHIE

B I B L I O G R A P H I E

1/ M. MARQUETTE
"Système de simulation hybride
Thèse de Docteur-Ingénieur, Lille 1977

2/ TOMOYUKI, MATSUZAKA
"Continous system Simulation Program Implemented on a personnal computer"
IFAC - Congrès de Vienne, Septembre 1986

3/ R. HERMAN, M. KOHNE
"Design and Application of the Portable Simulation"
IFAC - Congrès de Vienne, Septembre 1986

4/ J-W. NEERMAN
"Continous Dynamic System Simulation for Microcomputer"
Neerman Automatisering, Netherlands, 1983

5/ F. GAUSCH
"Simulation Studies Using the Program DASP"
IFAC - Congrès de Vienne, Septembre 1986

6/ V. MALBASA, D. OBRADOVIC
"Multimicroprocessor Software for Dynamic System Simulation"
IFAC - Congrès de Vienne, Septembre 1986

7/ A. LEMAITRE, M. PERRON
"Modélisation et Simulation du séchage du papier pour la conception de sécheries à hautes performances"
XVIIe EUCEPA Conférence - vienne, Octobre 1977

8/ J.P. BABARY, A. BERGEAUD
"Etude d'un système de réacteur d'épitaxie en phase liquide"
Thèse de Doctorat de Spécialité, Toulouse 1974

9/ J.F. LAFAY
"Modèle en pression et température d'une extrudeuse Bi-Vis"
Thèse de Docteur-Ingénieur, Nantes, Juillet 1978

10/ C. FOULARD, S. GENTIL, J.P. SANDRAY
"Commande et Régulation par calculateur numérique, de la théorie aux applications"
Editions Eyrolles, Paris 1979

11/ G. CORAY
"Technique de programmation modulaire"
Cours de l'Ecole Polytechnique Fédérale de Lausanne, 1983

12/ A. OURMAEV
"Eléments de simulation sur calculateurs analogiques"
Editions - Moscou, 1979

13/ R. MAHL
"Algorithmique et Structures de Données"
Ecole Nationale Supérieure de St Etienne, 1980

14/ R. HERMAN, M. KOHNE
"Design and Application of the portable Simulation"
IFAC, Congrès de Vienne, Septembre 1986

15/ C. FOULARD, J.P. SANDRAZ
"Commande et Régulation par calculateur numérique, de la théorie aux applications"
Editions Eyrolles, Paris 1979

16/ A. OURMAEV
"Eléments de Simulation sur calculateur analogique"
Editions, Moscou, 1979

17/ G.W. BRAMS
"Réseaux de Pétri - Théorie et Pratique"
Tome 1 : Théorie et Analyse, Editions Masson, 1983

18/ R. MAHL
"Algorithmique et Structures de Données
Ecole Supérieure de St Etienne, 1980

19/ P. CHRETIENNE
"Les réseaux de Pétri temporisés"
Thèse d'Etat, Université Pierre et Marie Curie, Paris 1983

20/ G. GUIDOZ
"Le Grafcet : macro-représentation et structuration du traitement des automatismes"
Conférence Automation 85, Paris, Avril 1985

21/ A. OUNSY
"La réalisation et la mise au point d'un logiciel de simulation hybride didactique"
Thèse de Doctorat 3ème cycle, Lille 1985

22/ E. CASTELAIN
"Modélisation et Simulation interactive de cellules de production flexibles dans l'industrie manufacturière"
Thèse de Doctorat de l'Université, Lille 1987

23/ D. CORBEEL, J.C. GENTINA, C. VERCAUTER
"Modélisation homogène du graphe de contrôle d'un système de conduite de processus industriels"
Congrès d'Automatique AFCET, Toulouse, 1985

24/ C. VERCAUTER
"Sur un ensemble d'outils d'aide à la spécification et à la conception

des systèmes industriels"
Thèse de Doctorat 3ème cycle, Lille, 1982

25/ M. MARQUETTE
"Système de Simulation hybride"
Thèse de Docteur-Ingénieur, Lille, 1979

26/ R. LAWSON
"Technique de pilotage automatique et suivi de trajectoire"
DEA Automatique, Laboratoire du Centre d'Automatique, 1981

27/ F. GANSEL
"Simulation Studies Using the program DASP"
IFAC - Congrès de Vienne, Septembre 1986

28/ A.A.B. PRISTKEN
"The GASP IV Simulation language"
451PP, New-York, 1974

29/ M. MARQUETTE
"Système de Simulation hybride"
Thèse de Docteur-Ingénieur, Lille, 1979

RESUME

Nous présentons dans ce mémoire un ensemble de programmes constituant un outil adapté à la description et à l'étude des systèmes dynamiques. L'objectif est la mise en oeuvre et le développement d'un logiciel de simulation hybride multitâche sur la base d'une approche à la fois didactique et orientée vers le traitement parallèle des procédés industriels.

S'inspirant des méthodes de programmation parallèle, un moniteur hybride permet la hiérarchisation et la gestion des différents modèles de processus représentés sous forme de réseaux de blocs à partir du logiciel de base TUTSIM. Le langage de programmation FORTRAN-77 utilisé permet une élaboration facile des algorithmes dans le calcul de l'ensemble des blocs de réseaux, et une implantation aisée du logiciel présenté sur différents calculateurs. La communication et les échanges entre processus sont autorisés par un ensemble de macros-commandes assurant les paramétrages automatiques et le parallélisme dans le traitement.

Trois exemples choisis dans les domaines de l'aéronautique et de la chimie illustrent l'approche proposée.

MOTS-CLES

- CALCUL ANALOGIQUE ET HYBRIDE
- RESEAU DE BLOCS
- MULTITACHES
- MULTIPROCESSEURS
- MODELISATION
- SIMULATION