THESE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le titre de

DOCTEUR DE SPECIALITE
(Mathématiques Appliquées)

par

Pierre—Michel DUCROCQ

AIDE A LA SIMULATION DE PROCESSUS PHYSIOLOGIQUES

Soutenue le 19 Juin 1973 devant la Commission d'Examen

Membres du Jury : MM.	P.	BACCHUS	Président
	B.	DRIEUX	Examinateur
	J.C.	PAGES	Invité
	J.	SWYNGEDAUW	Invité
	C.	CARREZ	Rapporteur

UNIVERSITE DES SCIENCES

ET TECHNIQUES DE LILLE.

DOYENS HONORAIRES

MM. H.LEFEBVRE, PARREAU.

PROFESSEURS HONORAIRES

M. ARNOULT, Mme BEAUJEU, MM. BEGHIN, BROCHARD, CAU, CHAPPELON, CHAUDRON, CORDONNIER, DEHEUVELS, DEHORNE, DEHORS, FAUVEL, FLEURY, P.GERMAIN, HEIM DE BALSAC, HOCQUETTE, KAMPE DE FERIET, KOURGANOFF, LAMOTTE, LELONG, Mme LELONG, MM. LIEBAERT, MARTINOT-LAGARDE, MAZET, MICHEL, NORMANT, PARISELLE, PASCAL, PAUTHENIER, PEREZ, ROIG, ROSEAU, ROUBINE, ROUELLE, WIEMAN, ZAMANSKY.

PROFESSEURS TITULAIRES

M.	ANGRAND Jean Pierre	Géographie et Aménagement Spatial
M.	BACCHUS Pierre	Astronomie et Calcul
M.	BEAUFILS Jean Pierre	Chimie Générale
M.	BECART Maurice	I.U.T. Lille
M.	BLOCH Vincent	Psychophysiologie
M.	BIAYS Pierre	Géographie et Aménagement Spatial
M.	BONNEMAN Pierre	Chimie Industrielle
M.	BONTE Antoine	Géologie Appliquée
M.	BOUGHON Pierre	Mathématiques
M.	BOURIQUET Robert	Biologie Végétale
M.	CAPET Marcel-Francis	Institut de Préparation aux Affaires
M.	CELET Paul	Géologie Générale
M.	CONSTANT Eugène	Electronique
M.	CORSIN Pierre	Paléobotanique
M.	DECUYPER Marcel	Mathématiques
M.	DEDECKER Paul	Mathématiques
M.	DEFRETIN René	Biologie Animale - Directeur de l'Institut de Biologie Maritime de Wimereux
M.	DELATTRE Charles	Géologie Générale
M.	DURCHON Maurice	Biologie Animale
M.	FLATRES Pierre	Géographie et Aménagement Spatial
M.	FOURET René	Physique
M.	GABILLARD Robert	Electronique
M.	GEHU Jean Marie	Institut Agricole
M.	GLACET Charles	Chimie Organique
M.	GONTIER Gérard	Mécanique des Fluides
M.	GUILLAUME Jean	Biologie Végétale
M.	HEUBEL Joseph	Chimie Minérale
Mme	LENOBLE Jacqueline	Physique
M.	MONTREUIL Jean	Chimie Biologique
M.	POUZET Pierre	I.U.T. Lille

Mme SCHWARTZ Marie Hélène	Mathématiques
M. TILLIEU Jacques	Physique
M. TRIDOT Gabriel	Chimie Minérale Appliquée
M. VIDAL Pierre	Automatique
M. VIVIER Emile	Biologie Animale
M. WATERLOT Gérard	Géologie et Minéralogie
M. WERTHEIMER Raymond	Physique

PROFESSEURS A TITRE PERSONNEL

M. BOUISSET Simon	Physiologie Animale
M. DELHAYE Michel	Chimie Physique et Minérale 1er Cycle
M. LEBRUN André	Electronique
M. LINDER Robert	Biologie Végétale
M. LUCQUIN Michel	Chimie Physique
M. PARREAU Michel	Mathématiques
M. PRUDHOMME Rémy	Sciences Economiques et Sociales
M. SAVARD Jean	Chimie Générale
M. SCHALLER François	Biologie Animale
M. SCHILTZ René	Physique

PROFESSEURS SANS CHAIRE

M. BELLET Jean	Physique
M. BODARD Marcel	Biologie Végétale
M. BOILLET Pierre	Physique
M. DERCOURT Jean Michel	Géologie et Minéralogie
M. DEVRAINNE Pierre	Chimie Minérale
M. LOMBARD Jacques	Sciences Economiques et Sociales
Mle MARQUET Simone	Mathématiques
M. MONTARIOL Frédéric	Chimie Minérale Appliquée
M. PROUVOST Jean	Géologie et Minéralogie
M. VAILLANT Jean	Mathématiques

MAITRES DE CONFERENCES (et chargés de fonctions)

M. ADAM Michel	Sciences Economiques et Sociales
M. ANDRE Charles	Sciences Economiques et Sociales
M. AUBIN Thierry	Mathématiques Pures
M. BEGUIN Paul	Mécanique des Fluides
M. BILLARD Jean	Physique
M. BKOUCHE Rudolphe	Mathématiques
M. BOILLY Bénoni	Biologie Animale
M. BONNEMAIN Jean Louis	Biologie Végétale
M. BONNOT Ernest	Biologie Végétale
M. BRIDOUX Michel	I.U.T. Béthune
M. BRUYELLE Pierre	Géographie et Aménagement Spatial

M. CAPURON Alfred	Biologie Animale
M. CARREZ Christian	Calcul
M. CHOQUET Marcel	I.U.T. Lille
M. CORDONNIER Vincent	Calcul
M. CORTOIS Jean	Physique
M. COULON Jean Paul	Electrotechnique
M. DEBRABANT Pierre	Sciences appliquées
M. ESCAIG Bertrand	Physique
Mme EVRARD Micheline	I.U.T. Lille
M. FAIDHERBE Jacques	Biologie Animale
M. FONTAINE Jacques	I.U.T. Lille
M. FROELICH Daniel	Sciences Appliquées
M. GAMBLIN André	Géographie et Aménagement Spatial
M. GOBLOT Rémi	Mathématiques
M. GOSSELIN Gabriel	Sciences Economiques et Sociales
M. GOUDMAND Pierre	Chimie Physique
M. GRANELLE	Sciences Economiques et Sociales
M. GRUSON Laurent	Mathématiques
M. GUIBAULT Pierre	Physiologie Animale
M. HERMAN Maurice	Physique
M. HUARD de la MARRE Pierre	Calcul
M. JOLY Robert	Biologie (Amiens)
M. JOURNEL Gérard	Sciences Appliquées
Mle KOSMANN Yvette	Mathématiques
M. LABLACHE-COMBIER Alain	Chimie Générale
M. LACOSTE Louis	Biologie Végétale
M. LANDAIS Jean	Chimie Organique
M. LAURENT François	Automatique
M. LAVAGNE Pierre	Sciences Economiques et Sociales
Mle LEGRAND Solange	Mathématiques
M. LEHMANN Daniel	Mathématiques
Mme LEHMANN Josiane	Mathématiques
M. LENTACKER Firmin	Géographie et Aménagement Spatial
M. LEROY Jean Marie	ENSCL
M. LEROY Yves	I.U.T. Lille
M. LHENAFF René	Géographie et Aménagement Spatial
M. LOCQUENEUX Robert	Physique
M. LOUAGE Francis	Sciences Appliquées
M. LOUCHEUX Claude	Chimie Physique
M. MAES Serge	Physique
Mme MAILLET Monique	Sciences Economiques et Sociales
M. MAIZIERES Christian	Automatique
M. MALAUSSENA Jean Louis	Sciences Economiques et Sociales
M. MESSELYN Jean	Physique
M. MIGEON Michel	Sciences Appliquées
M. MONTEL Marc	Physique
M. MONTUELLE Bernard	I.U.T. Lille
M. MUSSCHE Guy	Sciences Economiques et Sociales
M. NICOLE Jacques	E.N.S.C.L.
M. OUZIAUX Roger	Sciences Appliquées
M. PANET Marius	Electrotechnique
M. PAQUET Jacques	Sciences Appliquées
M. PARSY Fernand	Mécanique des Fluides
M. PONSOLLE Louis	Chimie (Valenciennes)
M. POVY Jean Claude	Sciences Appliquées
M. RACZY Ladislas	Radioélectricité
Mme RENVERSEZ Françoise	Sciences Economiques et Sociales
M. ROUSSEAU Jean Paul	Physiologie Animale

M. ROYNETTE Bernard	Mathématiques
M. SALMER Georges	Electronique
M. SEGUIER Guy	I.U.T. Béthune
M. SIMON Michel	Sciences Economiques et Sociales
M. SMET Pierre	Physique
M. SOMME Jean	Géographie et Aménagement Spatial
M. THOMAS Daniel	Chimie Minérale Appliquée
M. TOULOTTE Jean Marc	Sciences Appliquées
M. TREANTON Jean René	Sciences Economiques et Sociales
M. VANDORPE Bernard	Sciences Appliquées
M. VILETTE Michel	I.U.T. Béthune
M. WATERLOT Michel	Géologie Générale
Mme ZINN JUSTIN Nicole	Mathématiques.

Monsieur le Professeur P. BACCHUS a bien voulu me faire l'honneur d'accepter la présidence du Jury ; je tiens à lui exprimer ma vive gratitude.

Pour les précieux conseils et suggestions qu'il m'a constamment prodigués au cours de la réalisation de ce travail, Monsieur le Professeur CARREZ est également assuré de ma reconnaissance.

Je veux également remercier Monsieur le Professeur DRIEUX qui a accepter de juger ce travail et Monsieur le Professeur PAGES, Conseiller Scientifique à la compagnie I.B.M-France qui m'a, depuis plusieurs années, encouragé et soutenu dans cette voie de recherche.

Enfin je me dois de rappeler la bienveillante attention que m'a manifestée Monsieur le Professeur SWYNGEDAUW, Professeur à la Faculté de Médecine et Chef du service des isotopes du Centre Oscar Lambret. Sans son autorité et son expérience une bonne part de ce travail n'aurait pu être assurée.

Pour terminer, je n'aurai garde d'oublier tous ceux dont le travail et la compétence ont permis la réalisation de cette Thèse ; je pense en particulier à Madame CANTEGRIT et à Monsieur et Madame DEBOCK. Je les remercie.

A mes Parents,

A ma Femme.

TABLE DES MATIERES

INTRODUCTION

Les modèles et les techniques de simulation ont toujours été utilisés avec profit dans les domaines où la complexité des phénomènes, leurs diversités, ainsi que les lacunes des sciences et des techniques, rendent l'expérimentation difficile ou impossible. Ces modèles, de nature mathématique, mécanique, électrique ou autre furent et sont utilisés en astronomie, en électricité, en physique nucléaire etc... La physiologie a elle aussi depuis quelques temps tiré partie de cet outil. Il lui apporte des possibilités nouvelles, par exemple pour l'évaluation de paramètres difficilement accessibles par l'expérimentation directe, ou encore l'élaboration, la mise au point et le contrôle d'organes artificiels.

Le domaine pédagogique est également concerné, où l'étudiant peut, sur un système simulé, comprendre plus facilement certains mécanismes ou parfaire sa technique du diagnostic ou de la thérapeutique.

En outre l'élaboration du modèle se révèle souvent très fructueuse pour le physiologiste, qui est ainsi astreint à faire le point précis de ses connaissances et à les formaliser assez complètement. Un dialogue s'établit entre le système réel et le système simulé, par comparaison des résultats expérimentaux et théoriques. De nouvelles expériences peuvent alors être entreprises pour vérifier certains aspects et être à l'origine de découvertes ou de changements de conception.

Après avoir montré, dans le chapitre suivant, différents types de modèles rencontrés en physiologie et les différentes méthodologies en présence, nous proposerons un langage de description et de simulation de certains de ces processus.

Chapitre I

MODELES PHYSIOLOGIQUES

Les phénomènes physiologiques sont très divers. Les modèles qui les décrivent le sont tout autant. Voici, rapidement décrits, quelques uns des nombreux modèles publiés à ce jour. En général seules la structure du système et les fonctions caractéristiques seront données. Les valeurs numériques des coefficients, paramètres, seuils etc... ne seront pas explicitées.

1. Régulation Thermique

La chaleur diffuse dans le corps humain dans toutes les directions selon l'équation fondamentale :

$$\rho C \frac{\partial T}{\partial t} = K \frac{\partial^2 T}{\partial x^2}$$

ρ : densité des tissus — T : température

C : chaleur spécifique — K : conductivité thermique spécifique

x : abscisse dans la direction de diffusion

Il faut en outre tenir compte de la production métabolique M, et des pertes par radiation et convection R et V. On a alors :

$$\rho C \frac{\partial T}{\partial t} = K \frac{\partial^2 T}{\partial x^2} + M - R - V$$

La température est maintenue par trois actions :

- La conductivité thermique K est fonction, par l'intermédiaire de la vasomotricité, d'une part de l'écart entre la température réelle et la température idéale, T_o, et d'autre part de la vitesse de variation de cet écart.

$$K = f_1(T - T_o , \frac{\partial T}{\partial t})$$

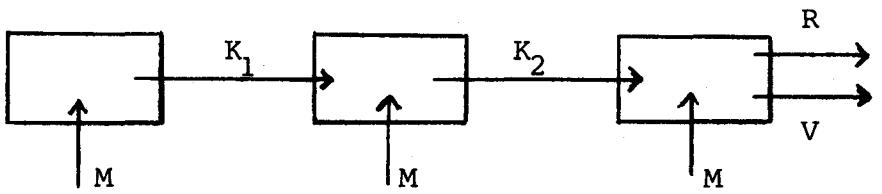
- Le terme de convection est une fonction linéaire croissante de l'écart, au dessus d'une certaine température T_1

$$V = k_v(T - T_o) \text{ pour } T \geq T_1 > T_o$$
$$= o \qquad T < T_1$$

- Enfin, quelle que soit la température, le métabolisme varie proportionnellement à l'écart :

$$\frac{\partial M}{\partial t}(t) = k_M(T(t) - T_o) \qquad k_M < o$$

Dans une simulation de ce système régulé, effectuée sur calculateur analogique, CROSBIE, HARDY & FESSENDEN utilisent une discrétisation du corps humain, en le représentant par trois espaces homogènes, interne, externe et intermédiaire, entre lesquels la chaleur circule. Le terme géométrique de la diffusion est alors représenté par les échanges thermiques entre ces trois zones.



Ils obtiennent ainsi trois équations différentielles du premier ordre, à coefficients et à second membre variables.

Les résultats de la simulation permettent d'étudier les variations de température en différentes régions du corps dans des conditions d'environnement définies ainsi que les réactions à des variations de température ambiante ou au cours d'exercices.

2. Le système musculaire

Certains réflexes musculaires sont des exemples typiques de boucles de régulation. Par exemple un sujet tenant une barre à la main est capable de la maintenir dans une certaine position, malgré diverses perturbations tendant à l'en écarter.

Ce contrôle est physiologiquement expliqué de la façon suivante :

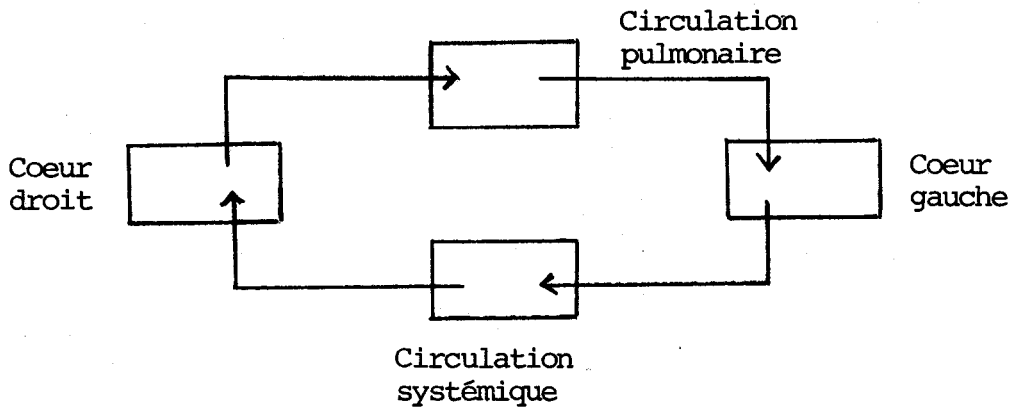
Certains fuseaux spécialisés du muscle ont une rétraction qui permet de maintenir la position constante. Ces fuseaux émettent des fibres nerveuses, venant de capteurs sensibles à l'étirement et à la vitesse d'étirement. Ils reçoivent en outre par les fibres gamma une commande agissant sur la sensibilité des capteurs.

On a pu, en écrivant les équations mathématiques qui lient positions, vitesses, coefficients de viscosité et forces, élaborer un modèle de ce type de régulation très fine. De nombreux travaux ont paru sur ce sujet (STARK, RICHALET etc...).

3. Système Cardio-vasculaire

De nombreux modèles de ce système ont été proposés. Ils adoptent tous le même schéma de base :

Deux pompes sont insérées sur un circuit à sens unique. Elles représentent respectivement les coeurs droit et gauche. Les portions de circuit qui les relient sont assimilables aux circulations pulmonaire et systémique et sont caractérisées par leur capacité et leur résistance. Les pompes elles, sont définies par leurs fréquences de mise en route et leur débit.



Ce système est doué de propriétés d'auto-régulations puisque le débit de sortie des pompes dépend des pressions dans le circuit et que ces dernières dépendent elles-mêmes du débit des pompes.

Des systèmes extrinsèques de contrôle peuvent être greffés sur ce schéma.

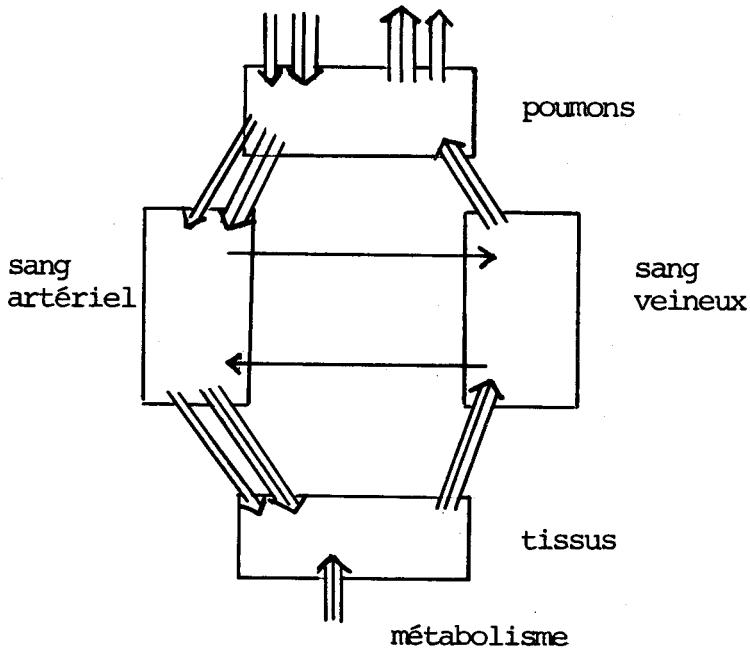
En agissant sur le type des relations entre les quatre composants principaux de ces modèles, de nombreux auteurs (GRODINS, DEFARES, BENEKEN etc...) ont étudié la variation des volumes ou des pressions en fonction d'autres caractéristiques physiologiques. Une régulation à long terme, par l'intermédiaire des variations liquidiennes totales a également pu être simulée.

Un aspect particulier de ces problèmes est l'étude de la forme et de la propagation des ondes de pression dans les différentes portions du circuit.

4. Le système Respiratoire

Ce système peut être schématiquement représenté de la façon suivante :

Les flèches simples représentent la circulation sanguine.
Les flèches doubles indiquent le trajet de l'oxygène et
les flèches triples, celui du gaz carbonique.



Les échanges entre les quatre blocs se font proportiennellement aux quantités contenues dans les blocs. Le retard circulatoire est souvent négligé.

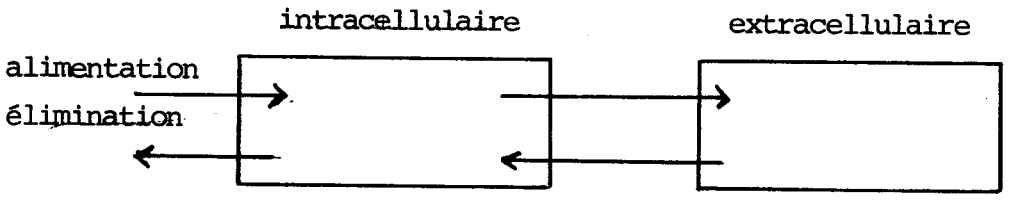
La ventilation, c'est-à-dire les échanges entre l'air extérieur et les poumons, est fonction de la différence entre la concentration veineuse en gaz carbonique et la valeur normale de cette concentration.

Le métabolisme produit, dans l'espace tissulaire, une certaine quantité de CO_2 qui pourra varier avec les conditions d'expérience.

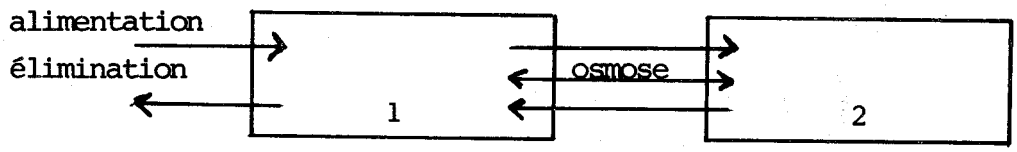
Ce modèle a pu être perfectionné ou étendu par l'introduction par exemple de la régulation cérébrale ou de l'aspect cyclique de la ventilation. C'est souvent le transport du CO_2 ou de O_2 qui est étudié. On pourra de même s'intéresser au comportement d'autres gaz, tels que l'oxyde de carbone CO par exemple.

5. Mouvements de l'eau et des électrolytes

Les mouvements de l'eau et des électrolytes constituent un système très complexe et très important, régulé de façon complète et rigoureuse par différents moyens. Dans les conceptions les plus simples, on peut considérer deux espaces, intra et extra cellulaire, entre lesquels circulent constamment l'eau et ces électrolytes dissous, essentiellement le sodium.



Mouvement des électrolytes



Mouvement de l'eau

En ce qui concerne les électrolytes les transferts de bloc à bloc se font proportionnellement à la concentration dans le bloc. L'apport extérieur est alimentaire. L'élimination est une fonction non linéaire de la concentration.

Pour l'eau les processus sont semblables. Il faut y ajouter l'aspect osmotique : un mouvement d'électrolyte dans un sens entraîne un mouvement d'eau dans le même sens.

On obtient donc un système de quatre équations différentielles du premier ordre :

$$\frac{dQ_{e2}}{dt} = k_{12}\, C_{e1} - k_{21}\, C_{e2}$$

$$\frac{dQ_{o2}}{dt} = r_{12}\, Q_{o1} - r_{21}\, Q_{o2} - w(k_{21}\, C_{e2} - k_{12}\, C_{e1})$$

$$\frac{dQ_{e1}}{dt} = A_e - f_e(C_{e1}) + k_{21}\, C_{e2} - k_{12}\, C_{e1}$$

$$\frac{dQ_{o1}}{dt} = A_o - f_o\,(C_{e1}, Q_{o1}) + r_{21}\, Q_{o2} - r_{12}\, Q_{o1} + w(k_{21}\, C_{e2} - k_{12}\, C_{e1})$$

où Q_{e1} = contenu du bloc 1 en électrolyte
Q_{o1} = " " en eau
Q_{e2} = " " 2 en électrolyte
Q_{o2} = " " en eau
C_{e1}, C_{e2}, C_{o1}, C_{o2} : concentration en eau et en électrolyte dans les deux blocs
k_{12}, k_{21}, r_{12}, r_{21} : taux de transferts
w, A_o : constantes
f_e, f_o : fonctions non linéaires, par exemple

$$f_e(C_1) = k_e\, C_1 \text{ pour } C_1 \leq C_{ref}$$
$$= k_e\, C_1 + k'_e(C_1 - C_{ref}) \text{ pour } C_1 \geq C_{ref}$$

$$f_o = f_o^1 + f_o^2 + f_o^3$$

$$f_o^1 (Q_{ol}) = r_o Q_{ol}$$

$$f_o^2 (Q_{ol}) = r'_o(Q_{ol} - Q_{ref}) \text{ pour } Q_{ol} \geq Q_{ref}$$

$$= 0 \qquad \text{sinon}$$

$$f_o^3 (C_{el}) = r''_o(C_{ref} - C_{el}) \text{ pour } C_{el} \leq C_{ref}$$

$$= 0 \qquad \text{sinon}$$

Ce modèle peut apparaître trop simple. En outre les valeurs numériques des divers coefficients sont difficiles à obtenir.

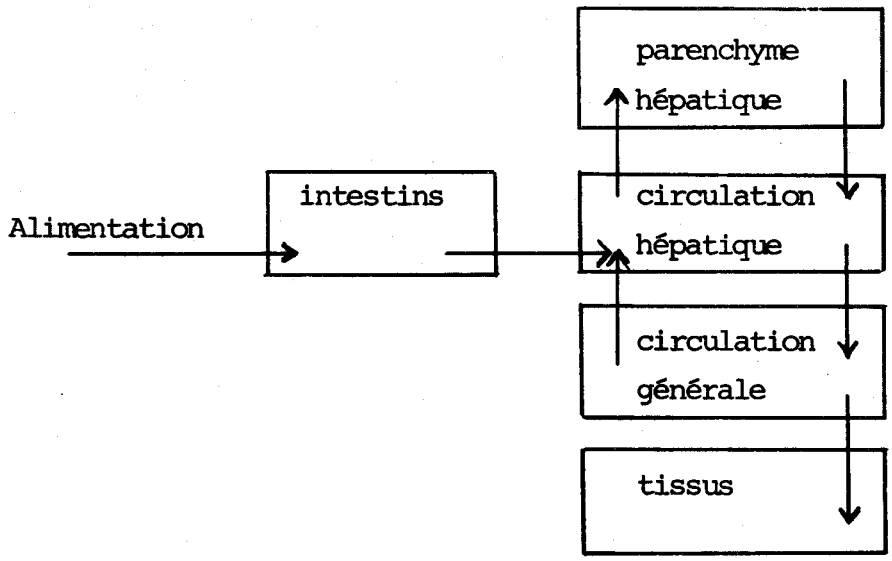
On peut améliorer le schéma précédent en faisant intervenir l'action d'hormones, telles que l'A-D-H. (hormone antidiurétique). On peut obtenir une expression mathématique de la quantité de cette hormone secrétée, en fonction de la quantité totale d'eau présente dans l'organisme. On connait également l'expression de l'influence de cette hormone sur l'élimination de l'eau. De même l'action de l'aldostérone sur les mouvements du sodium est connue et peut être simulée. On pourra également représenter le mécanisme de la soif, en déclenchant l'ingestion d'eau lorsque le contenu en eau sera compris entre deux valeurs de référence. Ce modèle, réalisé en particulier par REEVE & Coll., donne des résultats interessants qui ont permis, par exemple, de formuler certaines hypothèses sur l'origine de certains types de diabètes.

6. Régulations hormonales

Il existe de très nombreux exemples dans l'organisme de contrôles effectués par l'intermédiaire d'hormones. Nous venons d'en voir un exemple ci-dessus. Bien d'autres ont été étudiés parmi lesquels on peut citer la régulation de l'hormone A.C.T.H. par d'autres hormones comme le cortisol dont la production est elle-même fonction de la concentration en A.C.T.H.

Un autre exemple intéressant a été développé par BEAUGAS, COMYN et PARMENTIER. Il concerne la régulation de la glycémie (taux de sucre dans le sang). Bien qu'assez incomplet du point de vue physiologique il rend compte de phénomènes relativement complexes. Le trajet du glucose dans l'organisme est représenté sur le schéma de la page suivante.

Le foie joue le rôle d'un réservoir régulateur : si la concentration sanguine en glucose est trop élevée, le foie stocke, sous forme de glycogène, le sucre en excédent et le libère si la concentration devient insuffisante.



Cette action est en outre controlée par deux hormones, agissant en sens inverse, l'insuline et l'adrénaline.

La première stimule l'effet de stockage du foie, alors que la seconde augmente au contraire la libération du glucose. En outre l'insuline augmente la consommation des tissus en glucose.

Insuline et Adrenaline sont produites respectivement par le pancréas et par la glande medullo-surrénale. Les quantités secrétées dépendent de la concentration de l'hormone considérée dans le sang et de la glycémie : une hypoconcentration en glucose augmentera la production d'adrenaline, alors que celle de l'insuline sera une fonction croissante de la glycémie. Il y a donc un effet de contre-réaction. En outre une injection intramusculaire d'insuline peut être simulée.

Ces considérations se traduisent par un système d'équations différentielles du premier ordre exprimant les variations de concentration en sucre et en hormones dans les différentes régions.

Il faut d'ailleurs noter que BEAUGAS et COMYN n'ont pas utilisé pour présenter leur modèle une correspondance aussi nette entre les compartiments qui le constituent et les différentes entités physiologiques réelles. Ils estiment en effet qu'agir autrement serait prendre le risque d'assimilations excessives ou même absurdes. Ils s'en tiennent donc à une modélisation aussi abstraite que possible. Malheureusement cette méthodologie ne peut guère être appréciée des utilisateurs, médecins, biologistes, etc... non mathématiciens pour lesquels le support d'une représentation concrête est souvent indispensable.

Quoiqu'il en soit ce modèle a permis la simulation de différents cas, normaux et pathologiques et les résultats obtenus se sont souvent montrés compatibles avec la réalité.

D'autres modèles plus élaborés de cette régulation ont d'ailleurs été conçus, qui donnent des résultats très corrects.

7. Migration et cinétique d'éléments marqués

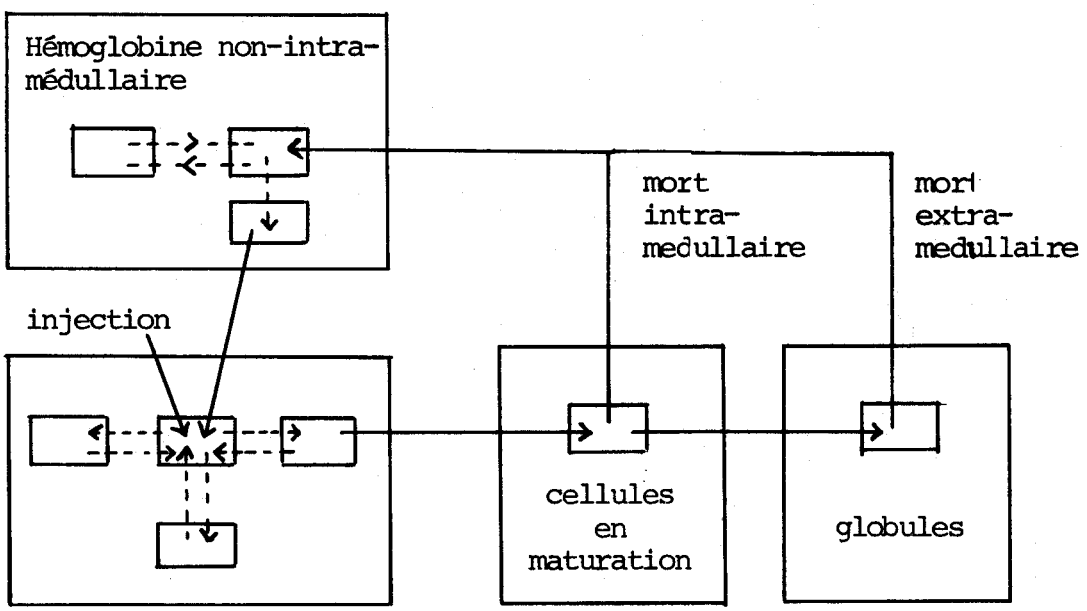
La technique consistant à introduire dans l'organisme des substances radioactives afin d'étudier leurs migrations, leur fixation, ou leur élimination est répandue et utilisée pour de nombreux domaines. Elle permet en effet d'obtenir par des mesures purement externes un grand nombre de valeurs qui seraient sinon très difficiles à atteindre.

La modélisation des phénomènes ainsi mis en évidence est souvent effectuée car la relative facilité avec laquelle les paramètres numériques expérimentaux peuvent être obtenus élimine un des principaux obstacles à une simulation efficace. Nous allons en donner quelques exemples :

a) *Cinétique du fer radioactif dans les cas normaux et pathologiques*
MONOT, MARTIN & COLL ont présenté deux modèles complémentaires de cette cinétique, le second étant plus particulièrement adapté à la simulation des cas pathologiques.
De façon générale ces deux modèles, ainsi que les variantes qui peuvent en être tirées, mettent en présence trois systèmes :
1) Le premier système représente la phase antérieure à l'absorption par la moëlle du fer marqué, injecté par voie intraveineuse. Il est constitué de compartiments dont les équivalents physiologiques ne sont pas totalement élucidés. Le nombre de ces compartiments (4 ou 5) et les différentes liaisons que l'on peut établir entre eux peuvent donner lieu à des variantes.
2) Le deuxième système correspond aux cellules erythropoïetiques en cours de maturation dans la moëlle.
3) Enfin le dernier représente l'ensemble des hématies (globules rouges).

Dans les cas pathologiques, il faut tenir compte de la possibilité d'une destruction anormale et prématurée soit des cellules erythropoïétiques soit des hématies. Ce phénomène se traduit par un retour du fer vers le premier système par l'intermédiaire de l'hémoglobine non intra-médullaire, qui constitue alors un quatrième système. Le modèle peut alors être schématisé de la façon

suivante :



Les auteurs démontrent, à partir de considérations physiologiques, que la quantité de fer marqué parvenant depuis la moëlle aux globules à l'instant t peut se mettre sous la forme :

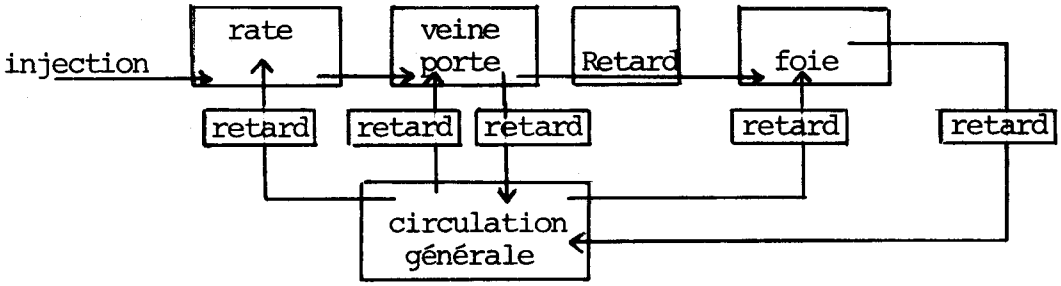
$$\int_0^{\theta} G(t,\theta,\tau)\, d\tau$$

où G est une fonction assez complexe déterminée expérimentalement et θ une constante obtenue également par l'expérience (durée du cycle de maturation). De même les phénomènes de mort extra et intra-medullaire conduisent à des formules du même type.

La simulation revient donc à la résolution d'un système integro-différentiel. Les résultats obtenus ont montré, outre l'influence des méthodes numériques utilisées, le rôle joué par certaines grandeurs physiologiques (avidité cellulaire, fonction de division cellulaire etc...). Certains modèles, proposés par des hématologues, se sont avérés incompatibles avec les données expérimentales.

b) *La Splenoportographie Isotopique*

Cet examen consiste à injecter dans la rate un produit radioactif (albumine marquée à l'I^{131}) et à mesurer ensuite l'évolution de la radioactivité dans la zone d'injection, dans la zone hépatique et dans la région sous-clavière COMYN propose plusieurs modèles de plus en plus complets du processus de la dilution du produit marqué. Le plus élaboré est représenté par le schéma :



Les valeurs numériques des retards et des différents coefficients de passage, obtenus en première approximation par des considérations physiologiques, ont été ensuite affinés par ajustements successifs.
Ce modèle fournit des résultats ayant une très bonne ressemblance avec les résultats expérimentaux et permet en outre l'étude des conséquences de certaines anomalies hépatiques.
Certaines hypothèses, concernant par exemple le processus de fixation de la substance sur le foie, peuvent également être testées.

c) *Le Nephrogramme Isotopique*

Un exemple détaillé d'un modèle des processus étudiés par cet examen est donné dans un chapitre ultérieur avec sa programmation en SIPHALGOL.
De nombreux autres modèles ont été proposés, parmi lesquels il faut citer ceux de PIRCHER, MARTIN et MONOT, COMYN et PARMENTIER.
Dans tous les cas il s'agit ici encore d'étudier le passage et la répartition d'un produit marqué entre plusieurs territoires suivant des lois connues ou hypothétiques dans l'espoir de retrouver des résultats expérimentaux.

8. Les processus à caractère aléatoire

Les processus entrant dans cette catégorie sont très variés. Ils concernent, par exemple, le mouvement brownien des molécules, ou la croissance de populations de bactéries ou de cellules, ou encore la cinétique de divers enzymes. La plupart des phénomènes décrits dans les exemples précédents, s'ils sont considérés d'un point de vue microscopique et non plus macroscopique, peuvent être interprêtés de façon probabiliste.

Les techniques de simulation mises en oeuvre dans ces cas (Monte-carlo etc...) ne sont pas fondamentalement différentes de celles que l'on rencontre dans les domaines industriels ou économiques. On y rencontre les mêmes problèmes de

de génération de nombres aléatoires, de gestion de file d'attente, d'estimation de lois ou de grandeurs statistiques.

Les références [1], [4], [5], [6], [9], [12], [13], [17], [18], [21], [27], [28], [32], [33], [34], [38], [40], [47] de la bibliographie fournissent d'autres exemples de modèles et de simulation. Les pages qui précèdent ne constituent certes pas une liste exhaustive des différents types de systèmes physiologiques que l'on peut être amené à simuler. Elles en donnent toutefois un aperçu assez significatif.

Il faut noter d'abord que, quelle que soit la nature du système simulé, différentes façons de concevoir et d'aborder le problème se présentent. On peut en distinguer deux classes. En premier lieu, celle des modèles qui prétendent à une justification physiologique immédiate. Ils décrivent directement une version plus ou moins simplifiée du système réel. Ils impliquent obligatoirement la formulation d'hypothèses physiologiques, parfois discutables.

L'avantage de cette méthode est évidemment de permettre une interprétation facile des paramètres du modèle et des phénomènes observés lors de l'expérience de simulation. Par contre le caractère souvent implicite des hypothèses faites peut être à l'origine d'assimilations abusives ou absurdes entre les entités constituant le modèle et ces composantes du système réel. C'est pourtant dans cette optique qu'ont été présentés, plus ou moins nettement, la plupart des exemples cités dans ce chapitre.

L'autre méthode consiste à adopter un modèle de nature strictement mathématique, constitué d'équations choisies de façon à déterminer, pour certaines variables privilégiées du modèle, une évolution au cours du temps dont la forme mathématique soit semblable à celle de l'évolution des grandeurs expérimentalement mesurables.

Par exemple, si une étude préliminaire a semblé montrer que la courbe de l'évolution dans le temps de la concentration sanguine d'un produit est une combinaison linéaire de deux exponentielles décroissantes, le modèle correspondant sera un système de deux équations différentielles du premier ordre. On peut considérer que ces équations sont la traduction d'un processus d'échange entre deux compartiments, mais aucune hypothèse n'est faite concernant la nature physiologique de ces compartiments. Dans ce cas le risque d'absurdité est certes réduit, mais l'information apportée sur le système simulé est uniquement descriptive et en aucune façon explicative.

Nous prendrons d'autant moins partie pour l'une ou l'autre méthode que l'optimum, qui est d'ailleurs fonction du système à simuler, aussi bien que des connaissances et des préjugés de l'auteur du modèle, cet optimum se situera bien souvent dans une solution hybride.

La suite et en particulier le système de programmation que nous proposons, devrait être indépendant du choix méthodologique. Quoiqu'il en soit on peut opérer parmi les modèles présentés les discriminations suivantes :

En premier lieu les modèles de processus à caractère aléatoire forment une classe très différente des autres tant en ce qui concerne leur forme que les méthodes numériques qui seront utilisées. Ils sont eux mêmes très variés et il parait assez difficile d'élaborer une formulation générale de ces problèmes. Parmi les modèles restant, de type déterministe, on peut constater qu'un certain nombre d'entre eux admettent facilement une description formalisée en termes de circulation, d'échange entre un nombre fini d'entités.

Les modèles du système cardio-vasculaire décrivent bien la circulation du sang entre quatre entités assimilées aux coeurs droit et gauche et aux circulations pulmonaire et systémique. De même les échanges de CO_2 entre les poumons, le sang et les tissus font bien l'objet des modèles du système respiratoire.

Par contre d'autres modèles échappent totalement à ce type de description. On voit mal comment, par exemple, un modèle d'un phénomène musculaire pourrait être mis sous cette forme. Enfin certains systèmes n'admettent cette formulation qu'au prix d'hypothèses simplificatrices ou d'artifices de présentation. C'est le cas par exemple du modèle présenté de la régulation thermique où la circulation de la chaleur entre les différentes parties du corps n'a pu être représentée qu'au prix d'une discrétisation tout à fait arbitraire de l'espace corporel.

La classe de modèles ainsi définie ne recouvre donc pas toute l'étendue des processus physiologiques simulables. Elle en constitue pourtant une part importante puisqu'on pourra y inclure en général la circulation sanguine, les échanges gazeux, les phénomènes pharmacologiques, la diffusion et l'élimination des produits etc... Elle recouvre en fait les phénomènes dont la description mathématique ne ferait intervenir que des équations différentielles du premier ordre par rapport au temps.

C'est uniquement à cette classe de processus dynamiques, déterministes et continus que nous nous intéresserons dans la suite.

Chapitre II

S I M U L A T I O N

La description de modèles effectuée dans le chapitre précédent a permis de voir quelles sont les premières étapes de la démarche des auteurs de modèle et de simulation.

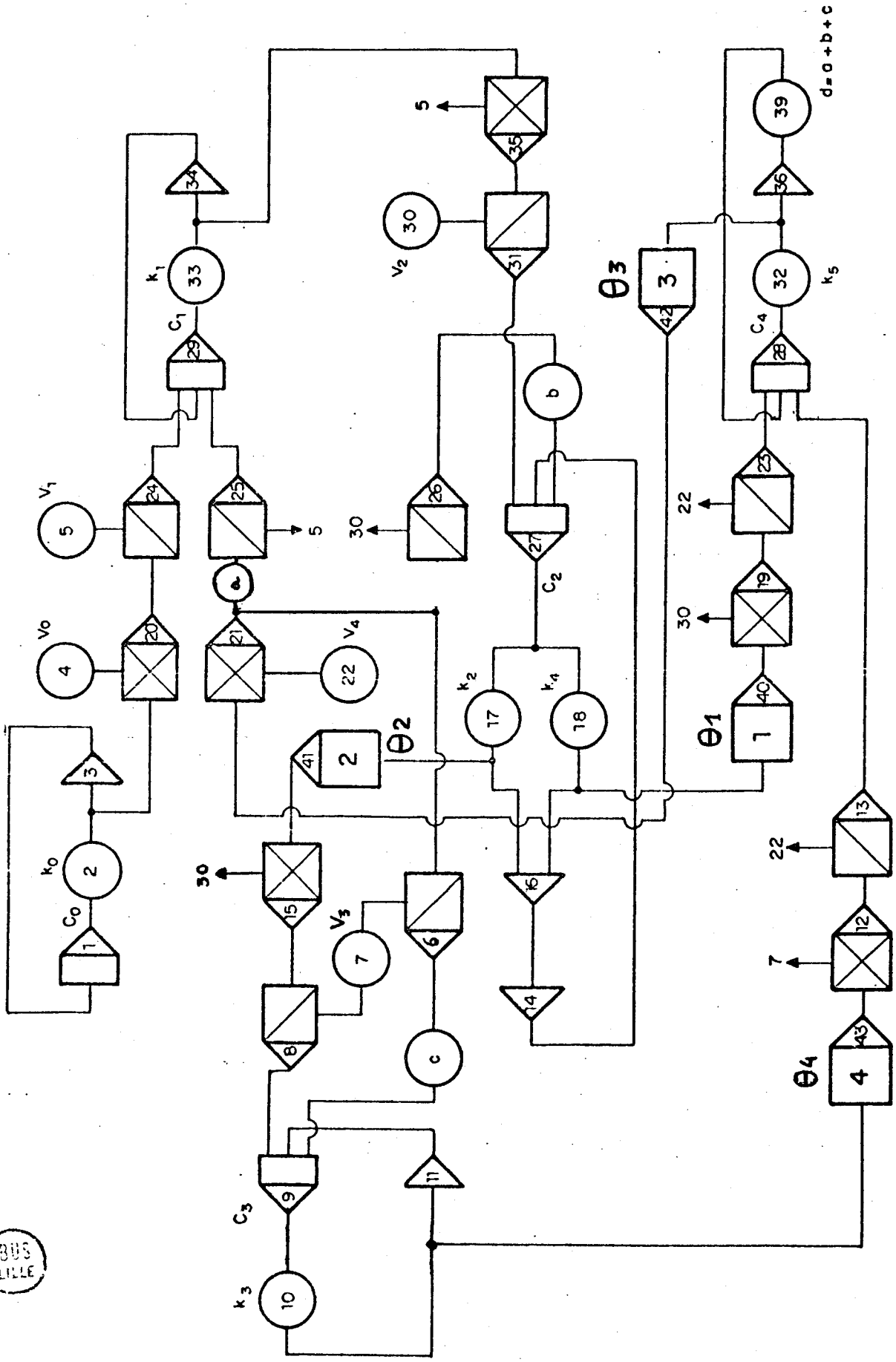
II.1. Les étapes de la simulation

Il est d'abord,bien entendu, nécessaire d'avoir une connaissance aussi précise que possible du phénomène à simuler. Une étude préalable, détaillée et critique, est donc la première préoccupation. Elle ne concerne que l'aspect purement physiologique du système et n'est donc pas notre propos ici.

Elle amène à préciser sans ambiguïté les connaissances ou les incertitudes sur le comportement du système, ainsi que les hypothèses qui seront faites. L'image, éventuellement simplifiée, du système qui s'en dégage est ensuite formalisée, par exemple sous forme d'un schéma. Pour la classe de processus qui nous intéresse ce schéma sera très souvent, on en a vu des exemples, constitué d'un réseau de flèches reliant des blocs fonctionnels. Ce symbolisme exprime que la ou les entités considérées circulent, dans la direction et le sens des flèches, d'un bloc à l'autre. Des indications supplémentaires seront en général portées sur le schéma, concernant la nature particulière des échanges entre chaque bloc et lorsqu'elles sont connues, les valeurs numériques des taux, coefficients, seuil, retard etc ... qui interviennent dans ces échanges.

En règle générale ce schéma qui est le reflet immédiat des hypothèses initiales sert ensuite de support à l'élaboration des équations mathématiques dont la résolution constituera en fait la véritable expérience de simulation. Cette résolution étant rarement possible analytiquement elle ne peut être effectuée que par des méthodes analogiques ou numériques.

Dans l'un et l'autre cas les équations devront donc être à nouveau transposées, soit sous forme de circuit analogique, soit sous forme d'un programme destiné à un calculateur digital. L'utilisation de circuits analogiques présente certains avantages grâce à la relative facilité de représentation des équations et à la possibilité de modifier aisément les valeurs numériques des paramètres.



(d'après COMYN)

Les problèmes d'ajustement sont alors plus commodément abordés. En outre le coût moins élevé de cette technique la fait souvent préférer. Par contre les calculateurs digitaux ne connaissent pas les limitations de l'analogique en ce qui concerne la précision des résultats, les facteurs d'échelles, la diversité des types de systèmes envisagés etc ...

En outre la formalisation souvent assez poussée des programmes les rend plus abordables à des non-spécialistes et en tout cas plus facilement modifiables que les circuits analogiques dont la mise en oeuvre est souvent longue. La nécessité d'effectuer à nouveau des câblages complexes rend difficiles les changements de modèle. Ce dernier inconvénient est d'ailleurs supprimé dans les calculateurs hybrides. Dans l'hypothèse d'un calculateur digital on utilisera parfois un langage assembleur, plus souvent un langage évolué à vocation générale. Ce pourra également être un langage plus particulièrement orienté vers ce genre de problème. Le langage C.S.M.P par exemple, ou SIAL 70, utilise une formulation par blocs semblable à celle des circuits analogiques pour programmer la résolution d'équations différentielles.

A titre d'exemple la figure ci-contre représente le diagramme C.S.M.P du modèle de la spleno-portographie isotopique de COMYN. Il correspond à un système de cinq équations différentielles du premier ordre, avec retards. Ce diagramme sera transmis à l'ordinateur soit directement par un écran cathodique, soit après une transcription sur cartes perforées selon un code adapté.

Quelque soit le mode de résolution adopté la suite des opérations est ensuite identique. Les résultats fournis par l'expérience de simulation sont confrontés avec les résultats expérimentaux. On cherchera bien entendu à obtenir la meilleure ressemblance entre ces deux types de résultats. Le critère de ressemblance est d'ailleurs bien difficile à définir. Les différences constatées peuvent être dues :

- à l'inadéquation du modèle,
- aux simplifications et aux phénomènes secondaires négligés lors de l'élaboration du modèle,
- aux erreurs accidentelles et aléatoires qui entachent les résultats expérimentaux.

Aussi plusieurs modèles, même très différents, peuvent-ils donner des résultats qui paraissent satisfaisants, sans qu'il soit possible à ce stade de déterminer lequel est le meilleur. Une tentative de validation pourra toutefois

être faite en comparant les réactions du modèle dans un nombre maximum de circonstances différentes avec les réactions du système réel dans les situations analogues (pour autant qu'on puisse les déterminer). En tout état de cause on n'obtiendra jamais de preuves rigoureuses de la validité des hypothèses, tout au plus on réunira un faisceau de présomptions convergentes.

Par contre si les résultats simulés et expérimentaux sont totalement dissemblables on pourra conclure au rejet des hypothèses faites. Il faudra donc modifier ces dernières. Les modifications pourront porter sur la structure même du modèle ou, et c'est le cas le plus fréquent, sur les valeurs numériques des paramètres utilisés.

Les nouvelles hypothèses conduiront à une nouvelle expérience de simulation dont les résultats seront à nouveau interprétés et donneront éventuellement lieu à d'autres modifications. Ainsi, par approximations successives, on pourra espérer améliorer le modèle et cerner au plus près la réalité. On peut imaginer de faire effectuer ce travail d'ajustement des paramètres automatiquement par l'ordinateur.

Aucune solution générale n'a encore été proposée à ce jour mais dans un certain nombre de cas particuliers des réalisations partielles ont pu être mises en oeuvre. Nous en parlerons plus en détail dans le Chapitre V.

II.2. Formalisation des modèles

On ne peut manquer d'être frappé par le fait qu'au travers des étapes décrites ci-dessus, le même modèle se trouve formulé d'au moins quatre façons différentes :

- Forme initiale, telle qu'elle découle immédiatement de l'étude préliminaire du système. Utilisant le langage naturel elle ne saurait être, dans l'état actuel des choses, directement exploitée.
- Forme schématique : un schéma, éventuellement éclairé de quelques indications annexes souvent numériques, représente à l'aide de symboles simples et peu nombreux les différents phénomènes et intéractions intervenant dans le système.
- Forme mathématique : l'introduction de symboles mathématiques et d'identificateurs littéraux pour les variables en jeu, permet de représenter les équations qui permettront de calculer les résultats cherchés.
- Forme programmée : directement, ou par appel à des sous programmes, les algorithmes de résolution des équations sont présentés et enchainés dans un ordre convenable.

De ces quatre formes, la première est propre au physiologiste, les deux dernières à l'informaticien ou à l'automaticien qui opèrera la simulation. C'est par la seconde forme que la communication entre eux s'établira. Elle est assez simple et assez souple pour satisfaire le premier, assez rigoureuse pour être acceptée du second.

L'écriture de ce schéma constitue donc la phase essentielle de l'opération. Il est très tentant de limiter autant que possible le travail à cette phase, en y concentrant les étapes suivantes de formalisation mathématique et programmée. Ceci n'a rien d'impossible dans le cas d'une résolution des équations par voie numérique sur un calculateur digital puisque, le modèle étant parfaitement défini par le schéma, l'ordinateur doit pouvoir en déterminer le programme de simulation.

II.3. Programmation schématique

Il s'agit donc d'établir un système de programmation qui, utilisant un mode de description des modèles directement inspiré de la représentation schématique, suffirait à la mise en oeuvre des expériences de simulation et permettrait au physiologiste, auteur du modèle, une approche plus facile et plus souple du niveau informatique. Le premier travail est d'établir une liste aussi complète que possible des notions et des termes qui interviendront dans ces descriptions.

II.3.1. Blocs, produits, liaisons

La classe des modèles décrits est, on l'a vu, celle des modèles d'échange, donnant lieu à des équations différentielles par rapport au temps du premier ordre. Il faudra donc introduire en premier lieu la notion d'entité circulante. Dans la plupart des cas il s'agira de molécules, ou de cellules, qui, à leur échelle suivent leur voie propre mais dont l'ensemble se comporte suivant une loi définie par la résultante statistique des comportements particuliers. Cette entité passive circulante sera appelée "produit". Il faut bien voir que ce terme recouvre les cas les plus fréquents mais que dans le cas, par exemple, de la régulation thermique, le "produit" sera la chaleur qui n'a pourtant pas de structure moléculaire.

Comme dans la plupart des processus, industriels ou autres, l'entité passive s'oppose aux entités actives. Ce sont les constituants du modèle qui en assurent l'évolution, qui "traitent" le produit. Dans le cas de nos modèles le traitement constituera la plupart du temps simplement à recevoir le produit et à le renvoyer, parfois avec retard, vers d'autres points d'échange, avec un débit défini.

Si une méthodologie explicative a été adoptée ces entités actives correspondront de façon plus ou moins rigoureuse à des entités réelles, organes ou parties d'organes. Nous appellerons ces entités des "blocs". Le modèle est alors défini comme un ensemble de liaisons entre blocs. Chaque liaison est définie par son bloc origine et son bloc extrémité. Le modèle sera complètement décrit si on sait à chaque instant calculer le débit du produit dans chaque liaison. On supposera, ce qui ne restreint pas le domaine d'application, que la loi définissant cette quantité dépend uniquement du bloc origine de la liaison.

A chaque bloc doit donc être attaché un "type" définissant l'expression générale du débit de sortie de ce bloc. Par exemple on conviendra qu'un bloc de type "compartiment" a un débit de sortie proportionnel à son contenu.

Un catalogue de types devra être établi de façon à pouvoir décrire un maximum de situations.

II.3.2. Paramètres

Il faudra préciser, pour chaque liaison, un certain nombre de valeurs numériques précisant la loi générale définie par le type du bloc origine. Dans le cas de l'exemple ci-dessus cette valeur numérique serait celle du coefficient de proportionnalité à utiliser et qui peut être différent, non seulement pour chaque bloc compartiment, mais encore pour chaque liaison issue de ce bloc.

Ce sont les paramètres de la liaison. Selon les cas il peut s'agir de valeurs numériques (paramètres constants) ou d'expression arithmétique permettant de calculer à chaque instant une valeur, fonction de l'état du système (paramètres variables).

II.3.3. Evolution du système - Relations

L'existence de paramètres variables indique que les "fonctions" du système ne seront pas figées. Il doit en être de même pour la structure. Des liaisons pourront être supprimées au cours de l'évolution du système ou au contraire rétablies.

D'autres fonctions que le simple échange peuvent intervenir : un apport extérieur ou une fuite peuvent modifier le pool total d'un produit. Ou encore des transformations, synthèses ou utilisations, peuvent se superposer à la circulation des produits.

Plusieurs produits peuvent d'ailleurs intervenir dans un même système. Les blocs n'ont alors pas forcément les mêmes fonctions pour les différents produits.

L'état du système vis à vis d'un produit peut intervenir sur l'évolution vis à vis d'un autre produit. C'est ce qui se passe en particulier dans le cas des régulations hormonales où l'élévation en certains points de la concentration en hormones inhibe ou au contraire accélère d'autres processus, eux-mêmes influant sur la production d'hormone.

Cette interdépendance n'est pas nécessairement instantanée et en fait ce pourra être l'état du système au temps $t-\tau$ qui influera sur l'évolution au temps t. Il faudra donc doter le modèle d'une mémoire, enregistrant les informations utiles dans la suite et introduire des instructions utilisant ces informations.

Enfin les conditions d'expérience - durée, conditions initiales etc ... - devront être précisées, ainsi que les résultats désirés et la forme sous laquelle ils doivent être présentés.

II.3.4. Technique de programmation

La nécessité de pouvoir utiliser ce système de programmation sur un matériel quelconque exclue la création de toute pièce d'un nouveau langage. Il faudrait en effet écrire un compilateur ou un interpréteur pour chaque langage-machine, ce qui réduirait considérablement les avantages d'un langage destiné à servir occasionnellement à des physiologistes sans connaissance informatique. Il semble plus indiqué d'utiliser un langage général dont les compilateurs existent sur de nombreux matériels. L'accroissement du temps de traitement ou de l'encombrement de la mémoire qui peuvent en résulter sont de peu de poids en regard des avantages apportés par cette technique.

Le système de programmation sera alors constitué d'un ensemble de sous-programmes en langage général. La description du modèle et de l'expérience de simulation sera faite par des appels à ces sous-programmes dont les identificateurs constitueront ainsi des "phrases" ressemblant autant que possible à celles du langage naturel qui décriraient le schéma du même modèle. Les paramètres de ces sous-programmes interviendront également dans l'énoncé de la phrase. Ils seront aussi peu nombreux que possible et limités aux informations fournies par le programmeur. Les autres valeurs, élaborées par un sous-programme, seront transmises aux sous-programmes suivant sous forme d'arguments muets, par l'emploi d'identificateurs

globaux ou d'allocation commune de la mémoire.

Nous avons choisi d'écrire ces sous-programmes en ALGOL 60, mais le même travail aurait pu tout aussi bien être effectué en FORTRAN ou autres langages évolués.

Nous avons baptisé le système de SImulation de processus PHysiologiques en ALGOL : SIPHALGOL.

Chapitre III

S I P H A L G O L

Un programme SIPHALGOL sera toujours constitué de deux sections distinctes. La première fournira la description du modèle proprement dit et la seconde précisera les conditions de l'expérience de simulation.

III.1. Description du modèle

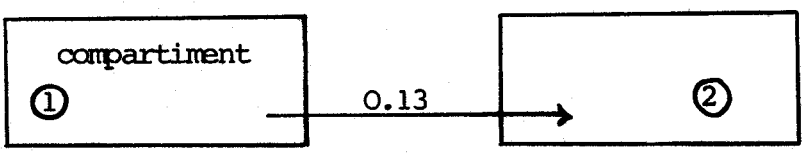
Le schéma descriptif du modèle étant conçu comme un ensemble de liaisons, le programme SIPHALGOL reprend ces liaisons une par une, produit par produit.

III.1.1. Informations préliminaires

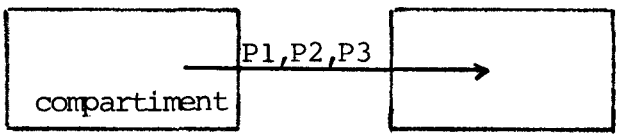
Avant tout, afin de permettre une allocation économique de la mémoire (voir chapitre suivant) il faudra donner le nombre de produits et de blocs en présence. De même la valeur du pas d'intégration devra être fournie. En effet la résolution numérique des équations différentielles étant obtenue par une méthode d'intégration par pas, la précision des résultats, mais aussi la durée des calculs, dépendent de la valeur du pas. Le choix de cette valeur est laissé au programmeur. Il doit l'indiquer dès le début du programme car en certaines circonstances elle peut avoir elle aussi une influence sur l'allocation de la mémoire.

III.1.2. Etablissement des liaisons

Imaginons le schéma partiel suivant :

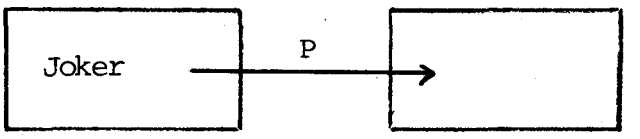


indiquant que le produit considéré passe du premier au second bloc, proportionnellement à la quantité de produit contenue dans le bloc de gauche. On convient classiquement d'appeler compartimental ce type de transfert. Le coefficient de proportionnalité qui précise la relation est 0.13.



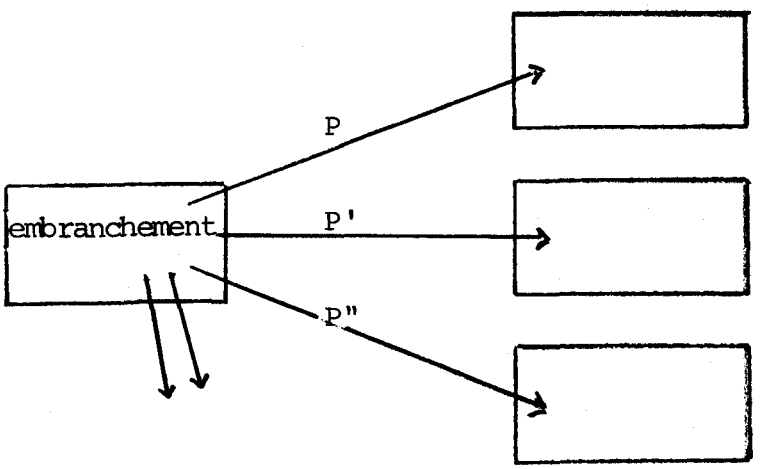
$S = P1 * (Q-P2)^{+} + P3$

P2, P3 } non donné ⟹ = 0



S=P

l'emploi d'un paramètre P variable permet d'obtenir une fonction quelconque comme débit de sortie.



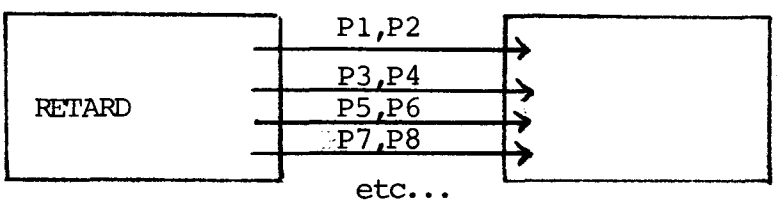
$S = P * E(T)$

$S' = P' * E(T)$

$S'' = P'' * E(T)$

etc ...

$P + P' + P'' + \ldots = 1$



$S = P2 * E(T-P1) + P4 * E(T-P3) + P6 * E(T-P5) + P8 * E(T-P7) + \ldots$

$P2 + P4 + P6 + P8 + \ldots = 1$

S=O

S = débit de sortie, Q = contenu, E(T) = débit d'entrée en temps T, P, P'..., P1, P2 = paramètres

L'existence d'une liaison entre les blocs 1 et 2 pourra en SIPHALGOL s'exprimer de la façon suivante :

BLOC(1); ALIMENTE(2);

Il faut également préciser le type de la liaison, c'est-à-dire le type du bloc origine. Ici on dira :

COMPARTIMENT(1);

Les deux indications - existence et type de la liaison - peuvent être fournies de façon plus groupée par :

COMPARTIMENT(1); ALIMENTE(2);

D'autres types de blocs sont également disponibles. La figure ci-contre en donne une description sommaire. On en trouvera les définitions complètes dans l'annexe B.

III.1.1. Paramètres de la liaison

Pour que la liaison soit complètement décrite il faudra donner en outre le paramètre à utiliser. La "phrase" complète sera alors :

COMPARTIMENT(1); ALIMENTE(2); PARAMETRE(0.13);

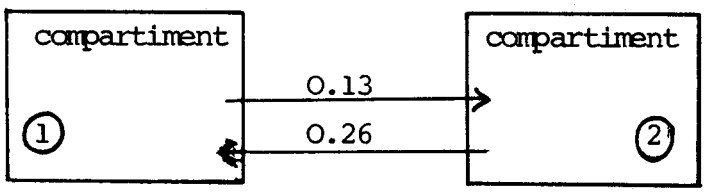
Dans certains cas (voir Annexe I) une liaison nécessite plusieurs paramètres. Ce sera le cas par exemple des liaisons dont le bloc origine est un compartiment. Le débit du produit dans ces liaisons est donné par la formule :

$$D = P1(Q - P2)^{+} + P3$$

exprimant que le débit est égal à la valeur P3 augmentée d'une quantité proportionnelle à l'écart positif du contenu Q du bloc avec une valeur de référence P2. Il y aura donc trois paramètres P1, P2, P3 à fournir (les paramètres nuls peuvent être omis, on retrouve alors les exemples précédents). On répètera autant de fois qu'il le faut le mot PARAMETRE, qui peut être remplacé par le mot ET.
Par exemple :

COMPARTIMENT(1); ALIMENTE(2); PARAMETRE(0.13); ET(50); ET(3);

Chaque liaison étant ainsi décrite, l'ensemble constituera une première partie du programme SIPHALGOL, appelée partie "MODELE".
Par exemple si le modèle est réduit à deux blocs de type compartimental entre lesquels circule le produit, le schéma sera : (voir page suivante)



Les instructions SIPHALGOL seront :

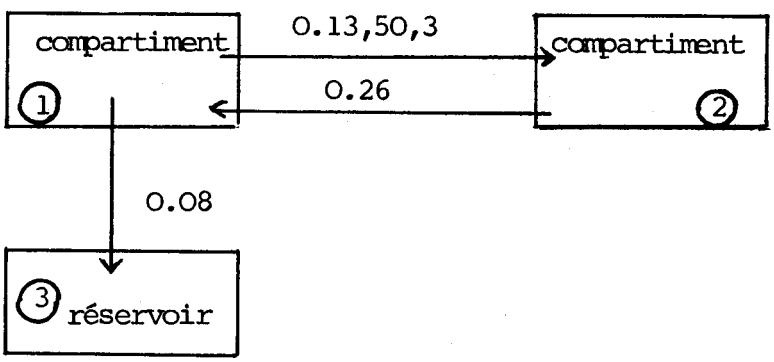
COMPARTIMENT(1); ALIMENTE(2); PARAMETRE(0.13);
COMPARTIMENT(2); ALIMENTE(1); PARAMETRE(0.26);

En fait les informations décrivant des liaisons et communes à plusieurs liaisons pourront être exprimées en une seule fois. Les liaisons peuvent différer par le produit concerné, le bloc origine, le bloc extrémité. Les instructions qui donnent ces informations peuvent être considérées comme positionnant des pointeurs qui ne seront pas modifiés tant qu'une instruction contraire ne sera pas prise en compte.

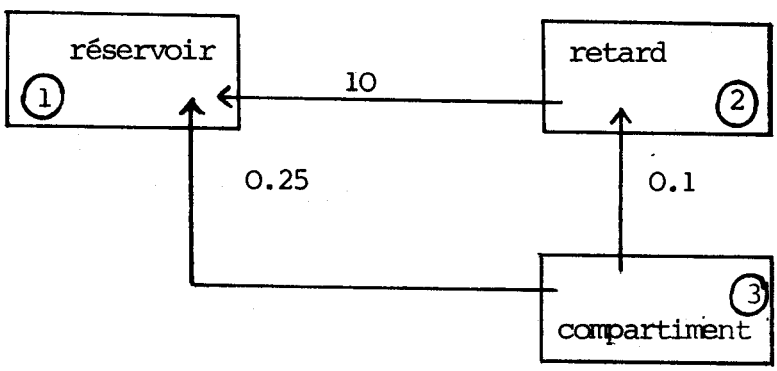
Soit par exemple le modèle suivant : trois blocs et deux produits sont en cause. Pour le premier produit l'échange a lieu comme dans l'exemple précédent entre les blocs 1 et 2. Il existe en outre une liaison du bloc 1 vers le bloc 3 qui accumule tout le produit qu'il reçoit. Le second produit passe du bloc 3 au bloc 1 et au bloc 2 suivant un mode compartimental, puis transite du bloc 2 au bloc 1 après avoir subi un retard de 10 unités de temps correspondant à une accumulation temporaire dans le bloc 2.

Les schémas seront :

produit 1



produit 2



Le programme sera :

```
PRODUIT(1); COMPARTIMENT(1); ALIMENTE(2); PARAMETRE(0.13);
ET(50); ET(3); AINSIQUE(3); PARAMETRE(0.08);
COMPARTIMENT(2); ALIMENTE(1); PARAMETRE(0.26);
RESERVOIR(3); SANSISSUE;
PRODUIT(2); COMPARTIMENT(3); ALIMENTE(2); PARAMETRE(0.1);
AINSIQUE(1); PARAMETRE(0.25);
RETARD(2); ALIMENTE(1); PARAMETRE(10);
RESERVOIR(1); SANSISSUE;
```

Les liaisons sont décrites dans un ordre indifférent.

III.1.4. Paramètres variables

On l'a vu, les paramètres affectés aux liaisons ne gardent pas toujours une valeur constante. Par exemple lorsque le bloc origine est un compartiment, le taux de transfert peut varier en étant influencé par des facteurs tels que la présence en plus ou moins grande quantité d'un autre produit dans le bloc. C'est le cas des régulations hormonales. Dans cette éventualité la valeur numérique du paramètres doit être remplacée en SIPHALGOL par la mention VARIABLE.
Exemple :

```
COMPARTIMENT(1); ALIMENTE(2); PARAMETRE(VARIABLE);
```

Une expression permettant de calculer à chaque instant la valeur effective devra bien sûr être fournie dans une autre partie du programme.

III.1.5. Mémorisation temporaire

Il peut arriver que ces influences mutuelles entre différentes grandeurs du système s'exercent avec retard et que ce soit l'état du système au temps $t-\tau$ qui intervienne au temps t. Les valeurs utiles devront alors être gardées en mémoire le temps nécessaire. Ceci n'est pas fait systématiquement et il faut que le programmeur le demande explicitement dans cette première partie. Par exemple si la quantité de produit n° 2 contenue dans le bloc 1 doit être enregistrée pour être utilisée 15 unités de temps plus tard, on écrira :

```
GARDER(CONTENU(1,2)); PENDANT(15);
```

Outre CONTENU(I,P) représentant le contenu du bloc I en produit P on peut

on peut également garder :

ENTREE(I,P) représentant le débit d'entrée dans le bloc I en produit P
SORTIE(I,P) " " de sortie du " " "
APPORT(I,J,P) " " dans la liaison entre les blocs I et J en produit P.

III.2. Description de l'expérience

La deuxième partie du programme, appelée partie "EXPERIENCE" va contenir toutes les informations qui n'ont pas trouvé place dans la première section parce qu'elles ne présentaient pas un caractère constant. Ce seront en particulier les informations sur les conditions de l'expérience de simulation. Il y a en fait sept catégories d'informations éventuelles.

III.2.1. Etat Pré-initial

Lorsque la mise en mémoire de certaines quantités a été demandée pour une période T1, ces quantités ne sont disponibles que lorsque le temps T est supérieur à T_o + T1 (T_o valeur initiale du temps). En effet dans le cas contraire - on dira alors que T est "petit" - l'expérience de simulation n'ayant pas encore commencé au temps T-T1, aucune valeur correspondant à ce temps n'a pu être calculée. Si une expression fait référence à une de ces quantités elle sera supposée de valeur nulle. Toutefois des indications contraires peuvent être données au moyen d'instructions telles que :

```
TPETIT;
FAIRE(EXCONTENU(1,2,TO-T1)); EGALE(0.5*T1+2);
FAIRE(EXENTREE(2,2,TO-T1)); EGALE(0.03*T1*T1);
FINTPETIT;
```

qui affecteront, pour toutes les valeurs de T1 permises, la valeur de l'expression argument de EGALE à la quantité argument du FAIRE précédent.

Les valeurs anciennes de CONTENU, ENTREE, SORTIE et APPORT sont respectivement appelées EXCONTENU, EXENTREE, EXSORTIE et EXAPPORT. Dans chaque cas un paramètre supplémentaire indique la valeur du temps pour laquelle la valeur est désirée.

III.2.2. Etat initial

L'état initial du modèle doit être indiqué.

On trouvera des instructions telles que :

```
INITIALEMENT;
PRODUIT(1); BLOC(1); CONTIENT(100); DEBITE(0.1);VERS(2);
DEBITE(0.15); VERS(3);
BLOC(2); CONTIENT(50);
PRODUIT(2); BLOC(1); CONTIENT(25);
FININIT;
```

Lorsque les valeurs initiales ne seront pas données elles seront supposées nulles. La valeur initiale du temps doit être donnée. La valeur finale, fixant la durée de l'expérience, peut également être fournie. L'expérience peut toutefois être interrompue d'autres façons (en particulier par une instruction STOP) :

```
DEPARTA(0); STOPA(50);
```

III.2.3. Modification du pool de produit

Outre les échanges entre blocs, les différents produits peuvent être impliqués dans d'autres types de phénomènes : ils peuvent être introduits de l'extérieur, ou élaborés à l'intérieur de certains blocs ou encore disparaître pour des raisons variables. On pourra décrire ces phénomènes par des "phrases" de ce type :

```
PRODUIT(1); BLOC(2); SYNTHETISE(0.72);
AUTEMPS(10); PRODUIT(2); BLOC(1); INJECTER(100); FINQUAND;
BLOC(2); UTILISE(0.25*CONTENU(2,1));
PRODUIT(3); BLOC(5); PERFUSER(0.50);
```

Dans le cas de l'injection l'apparition de la quantité de produit en argument est instantanée. Dans les autres cas les arguments représentent les débits d'apparition ou de disparition du produit.

III.2.4. Expression des paramètres variables

On devra aussi donner l'expression des paramètres qui ont été annoncés "VARIABLE" dans la première partie. Il faudra définir la liaison concernée. On écrira par exemple :

```
BLOC(1); VERS(2); PARAMETRE(CONTENU(1,2)-25);
```

III.2.5. Modification de structure

La structure du modèle peut être modifiée en cours d'expérience par l'ouverture ou la fermeture de certaines liaisons. Il suffira d'écrire, après avoir spécifié le produit concerné :

```
COUPER(I,J);
RETABLIR(I,J);
```

pour supprimer ou rétablir la liaison entre les blocs I et J. De même des blocs peuvent être totalement exclus du modèle, ou réinclus par les instructions :

```
EXCLURE(I); REINCLURE(J);
```

III.2.6. Instructions conditionnelles

La plupart des instructions qui viennent d'être décrites peuvent être astreintes à ne prendre effet que sous certaines conditions. Nous distinguerons entre instructions conditionnelles proprement dites et instructions périodiques.

1 - La condition peut être exprimée comme argument du mot QUAND :
QUAND (expression booléenne ALGOL); instructions SIPHALGOL ; AUTREMENT ; instructions SIPHALGOL ; FINQUAND ;
La deuxième partie peut éventuellement être supprimée.
On a alors :

```
QUAND(expression booléenne) ; Instructions SIPHALGOL ; FINQUAND ;
```

2 - Lorsque la condition porte exclusivement sur les valeurs du temps on pourra utiliser :

```
DEPUIS(T1); JUSQUA(T2); TOUSLES(HT); Instruction SIPHALGOL ;
```

Seule la première instruction suivant l'expression des conditions périodiques est affectée par cette condition. Un ou deux quelconques des trois mots DEPUIS, JUSQUA, TOUSLES, peuvent être omis.

III.2.7. Présentation des résultats

Enfin les résultats demandés et leur présentation doivent être exprimés. Les possibilités du matériel utilisé interviennent à ce niveau de façon importante. Le programme réalisé avec la machine Bull M40 du Laboratoire de Calcul permet d'écrire par exemple :

```
TOUSLES(15); AFFICHER(CONTENU(1,2)/CONTENU(1,TOTAL)); INTITULE("FRACTION?PARTIELLE\);
```

```
TRACER(CONTENU(2,2)); ENFONCTIONDE(T);
ETAT(60);
```

Ces instructions déterminent :

- toutes les quinze unités de temps l'impression de la valeur de l'expression en argument de AFFICHER, avec le titre indiqué,
- le tracé d'une courbe,
- l'impression sous forme d'un tableau de l'état général du système à la 60ème unité de temps.

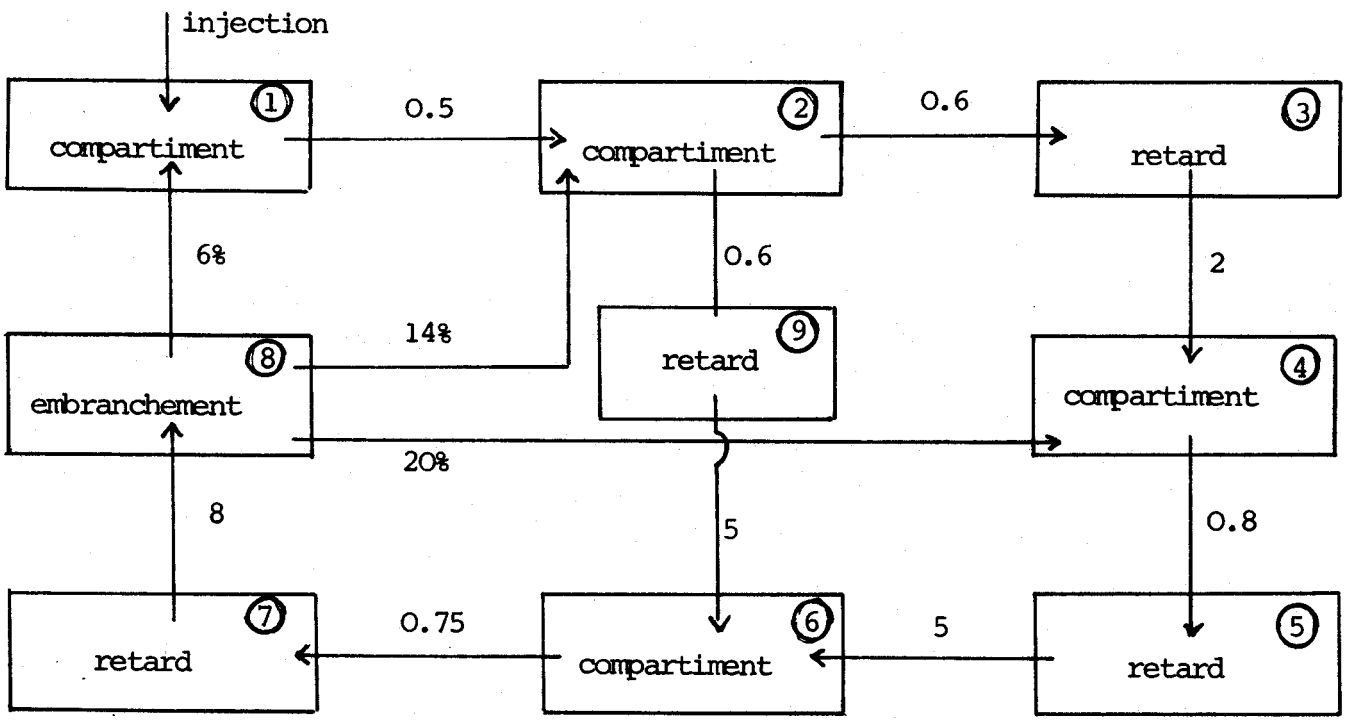
III.3. Exemples

A l'aide de ces instructions nous pouvons maintenant traiter des exemples plus élaborés que les précédents et écrire des programmes complets.

Nous tenterons de justifier ces exemples par des cas concrets mais il est bien évident que les modèles seront alors très simplifiés ce qui leur ôtera une grande partie de leur valeur d'un point de vue physiologique. Nous voulons simplement montrer comment SIPHALGOL permet de rendre compte de certains types de phénomènes.

III.3.1. La splenoportographie isotopique

Reprenons d'abord, de façon sommaire, le modèle de la splenoportographie isotopique décrit dans le ler chapitre. Un seul produit intervient ici : la sérum albumine marquée à l'Iode 131. Le schéma du modèle est le suivant :



Les compartiments 1, 2, 4 et 6 ont des équivalents physiologiques plus ou moins justifiés : respectivement rate, veine porte, foie et circulation générale. Les autres blocs sont des blocs utilitaires représentant les retards introduits sur la circulation ou la séparation du produit en plusieurs fractions. (L'unité de temps est la seconde, l'unité de mesure du produit radioactif est la microcurie). Les liaisons entre les blocs 2, 9 et 6 sont destinées à permettre la simulation d'anastomose porto-cave.

Le programme SIPHALGOL complet s'écrira :

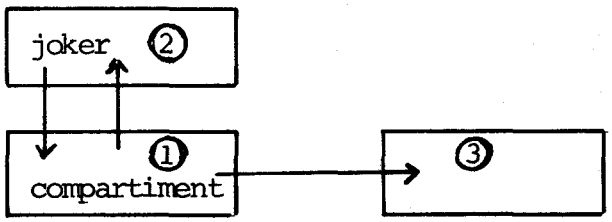
```
NBBLOCS(9); PAS(0.5);
COMPARTIMENT(1); ALIMENTE(2); PARAMETRE(0.5);
COMPARTIMENT(2); ALIMENTE(3); PARAMETRE(0.6);
AINSIQUE(9); PARAMETRE(0.6);
RETARD(9); ALIMENTE(6); PARAMETRE(5);
RETARD(3); ALIMENTE(4); PARAMETRE(2);
COMPARTIMENT(4); ALIMENTE(5); PARAMETRE(0.8);
RETARD(5); ALIMENTE(6); PARAMETRE(5);
COMPARTIMENT(6); ALIMENTE(7); PARAMETRE(0.75);
RETARD(7); ALIMENTE(8); PARAMETRE(8);
EMBRANCHEMENT(8); ALIMENTE(1); PARAMETRE(0.06);
AINSIQUE(4); PARAMETRE(0.2); AINSIQUE(2); PARAMETRE(0.14);
EXPERIENCE:DEPARTA(0); STOPA(100);
AUTEMPS(0); BLOC(1); INJECTER(150); FINQUAND;
SIMULATION(1); EXCLURE(9); SIMULATION(2); EXCLURE(3);
SIMULATION(TOUTES);
TRACER(CONTENU(6,1)); ENFONCTIONDE(T);
EXECUTION;
```

Ce programme, après avoir décrit le modèle, décrit deux expériences de simulation différente : dans la seconde un shunt du foie est réalisé par communication directe entre le compartiment "veine porte" et la circulation générale. La liaison entre veine porte et foie est alors totalement supprimée. On aurait pu s'approcher mieux de la réalité en la rendant simplement plus "résistante" en diminuant le taux de transfert entre les blocs 2 et 3.

III.3.2. Régulation glycémique

De même s'inspirant du modèle de la régulation glycémique décrit dans le chapitre I on pourrait écrire le programme SIPHALGOL correspondant. Nous n'en donnerons que quelques extraits mettant en lumière les aspects intéressants.

Par exemple une représentation partielle du système est donnée par le schéma :



Où : ① représente la circulation hépatique
② le parenchyme hépatique
③ la circulation générale

Le stockage ou la libération du glucose par le foie est fonction de l'écart de concentration par rapport à la normale et des concentrations d'insuline et d'adrénaline.

On aura donc dans la première partie du programme :

```
PRODUIT(1); COMPARTIMENT(1); ALIMENTE(2);
PARAMETRE(0.2); ET(0.06); ET(VARIABLE);
AINSIQUE(3); PARAMETRE(0.5);
JOKER(2); ALIMENTE(1); PARAMETRE(VARIABLE);
```

L'expression des paramètres variables est donnée dans la seconde partie :

```
PRODUIT(1); BLOC(1); VERS(2); ILYA(2); PARAMETRE(CONSTANT);
ET(0.02*CONCENTRATION(1,2));
BLOC(2); VERS(1);
PARAMETRE(0.2*MAX(0.06-CONTENU(1,1),0)+0.03*CONCENTRATION(1,3));
```

Ces instructions expriment que :

- le glucose rejoint la circulation générale en quantité proportionnelle à la quantité présente dans la circulation hépatique.
- La quantité captée par le foie comprend deux parts, l'une proportionnelle à l'écart en excès de la quantité de glucose par rapport à la normale, l'autre

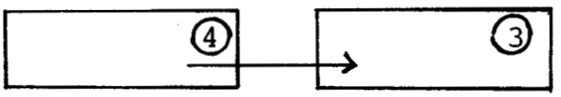
proportionnelle à la concentration en insuline (produit 2).

- La quantité libérée par le foie est, elle, constituée d'une part proportionnelle à l'écart en défaut de la quantité de glucose par rapport à la normale et d'une autre part proportionnelle à la concentration en adrénaline (produit 3).

On aurait pu éviter l'emploi assez lourd de la fonction MAX pour prendre la partie positive de l'écart dans l'expression du dernier paramètre de l'exemple. Il suffisait d'écrire :

```
PARAMETRE(0.2*(0.06-CONTENU(1,1))+0.03 CONCENTRATION(1,3));
QUAND(0.06-CONTENU(1,1)<0); COUPER(2,1); FINQUAND;
```

De même la secrétion de l'insuline par le pancréas, qui s'accroit lorsque la glycémie dépasse la normale et décroit lorsque la concentration sanguine de l'insuline augmente, aurait pu s'écrire :




```
JOKER(4); ALIMENTE(3); PARAMETRE(VARIABLE);

BLOC(4); VERS(3);
PARAMETRE(0.1*(CONTENU(1,1)-0.06)*(0.001-CONCENTRATION(3,2)));
QUAND(CONTENU(1,1)<0.06 or CONCENTRATION(3,2)>0.001);
COUPER(4,3); FINQUAND;
```

Bien entendu toutes les valeurs numériques qui viennent d'être utilisées sont purement imaginaires et n'ont aucun rapport avec les valeurs réelles qui devraient être établies soit par mesures expérimentales soit par ajustements successifs.

III.3.3. Expansion d'un embol

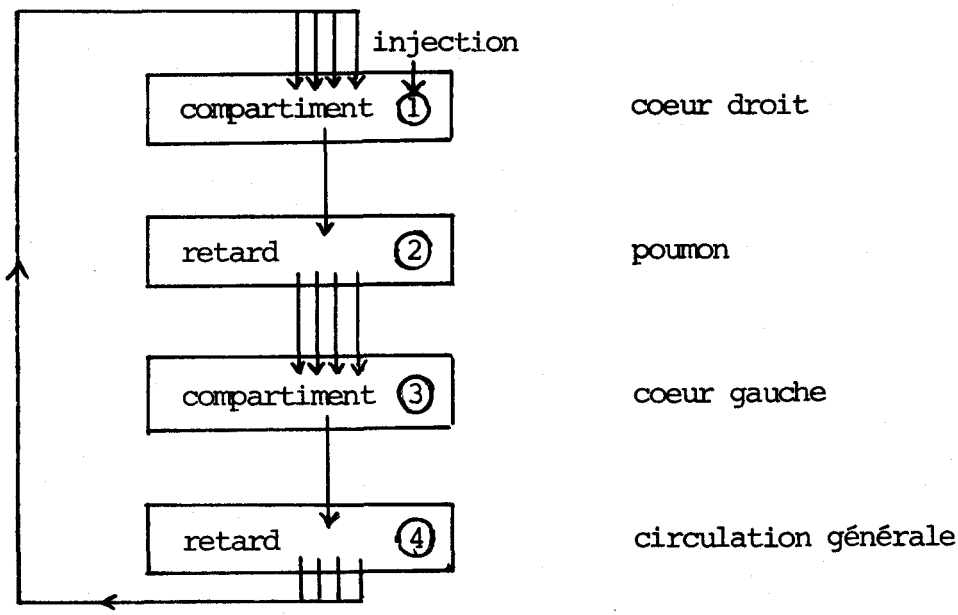
L'exemple suivant n'a pas été décrit dans le premier chapitre. Le processus étudié est celui de l'expansion d'un embol, en général radioactif, introduit dans la circulation sanguine par injection intraveineuse. Il traversera successivement le coeur droit, les poumons, le coeur gauche, la circulation générale avant de revenir dans le coeur droit et le poursuivre le cycle. Les coeurs droit et gauche sont des cavités de mélange où l'homogénéisation entre le sang et le produit radioactif est pratiquement instantanée.

Par contre la circulation pulmonaire semble pouvoir être décrite comme une série de canaux parallèles dont les temps de transit sont différents, distribués

autour d'une valeur moyenne.

La circulation générale peut être considérée d'une façon analogue.

On a alors le schéma suivant :



Les différents temps de traversée des poumons et de recirculation seront traduits en utilisant la possibilité de fractionner le produit entrant dans un bloc "retard" en plusieurs parts subissant des retards différents.

On écrira en SIPHALGOL :

```
NBBLOC(4);
COMPARTIMENT(1); ALIMENTE(2); PARAMETRE(0.5);
RETARD(2); ALIMENTE(3); PARAMETRE(10); ET(0.01);
ET(7); ET(0.2); ET(5); ET(0.3); ET(3); ET(0.4);
COMPARTIMENT(3); ALIMENTE(4); PARAMETRE(0.35);
RETARD(4); ALIMENTE(1); PARAMETRE(30); ET(1/3);
ET(10); ET(1/3); ET(5); ET(1/3);
EXPERIENCE:
DEPARTA(0); STOPA(50);
BLOC(1); DEPUIS(0); JUSQUA(1.25); PERFUSER(80);
TRACER(CONTENU(1,1)+CONTENU(3,1));
PUIS(CONTENU(2,1)); PUIS(CONTENU(4,1));
ENFONCTIONDE(T);
EXECUTION;
```

Ainsi le retard 2 réalise une réparation du produit en quatre fractions de 10, 20, 30 et 40 % subissant respectivement des retards de 10, 7, 5 et 3 unités de temps.

De même le retard 4 fait subir des retards de 30, 10 et 5 unités de temps a des fractions égales au tiers du produit arrivant dans ce bloc.

Chapitre IV

L'INTERPRETATION

IV.1 - Principes

L'interprétation d'un programme tel qu'il a été défini au chapitre précédent consiste principalement en l'élaboration et la résolution des équations différentielles associées au modèle décrit. L'essentiel de ce travail sera effectué par la procédure EXECUTION qui clot obligatoirement le programme. Elle reprendra toutes les informations transmises par les instructions précédentes et qui auront été temporairement enregistrées en mémoire. Par une méthode d'intégration numérique la valeur du vecteur d'état du modèle à chaque instant sera calculée. La méthode utilisée, on le verra, est une méthode par pas de Runge-Kutta d'ordre 2. Un pas de calcul nécessite alors deux phases. Lors de chacune de ces phases, donc deux fois par pas, certaines informations telles que la valeur des paramètres variables ou les quantités de produit apparaissant dans le modèle par exemple, doivent être réévaluées. Il est donc nécessaire d'exécuter les instructions correspondantes deux fois par pas. Par contre d'autres informations, paramètres constants, structure du modèle etc ..., sont fixes et n'ont pas à être réévaluées.

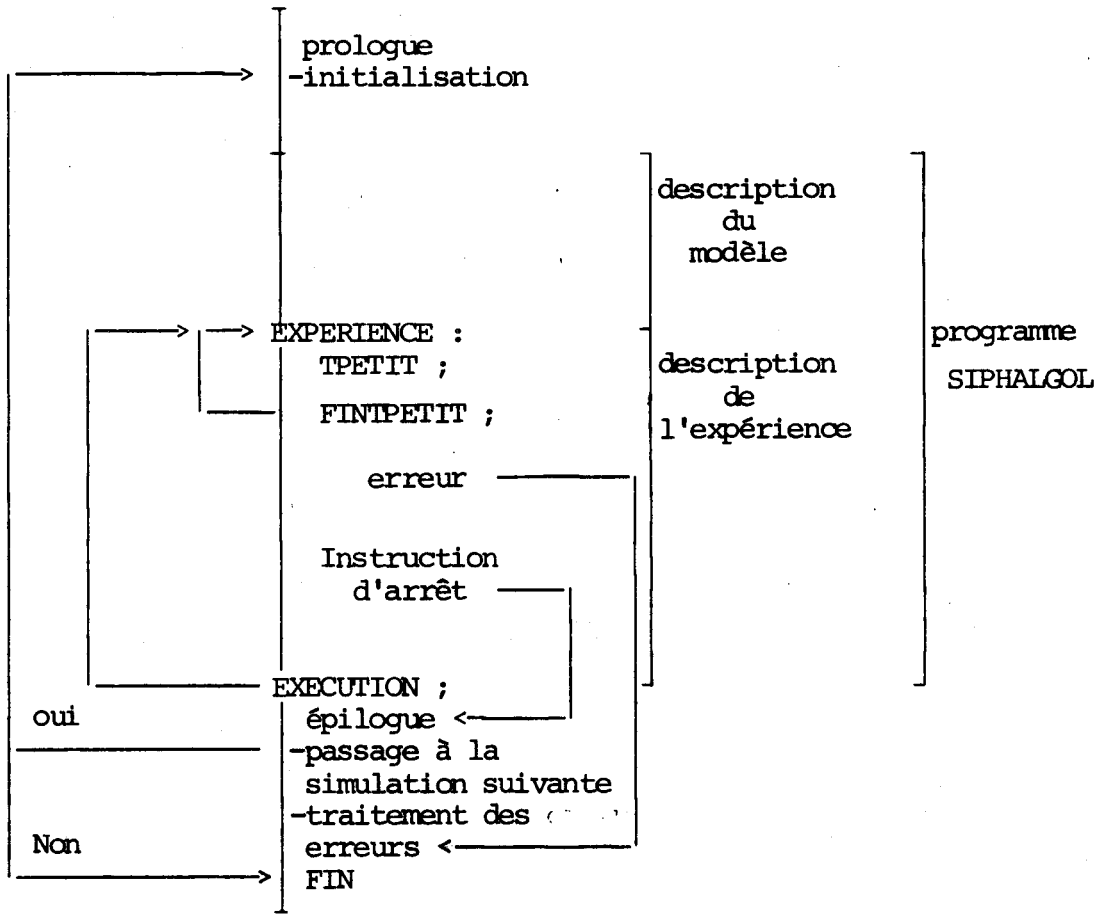
C'est pourquoi le programme est divisé en deux parties distinctes groupant, pour la première, les informations constantes et, pour l'autre, les informations variables. Ces deux parties sont séparées par l'étiquette EXPERIENCE. La procédure EXECUTION se termine par un renvoi vers cette étiquette. Ainsi les instructions de la seconde partie sont bien réexécutées à chaque phase.

De même, à l'intérieur de la seconde partie, le groupe d'instructions débutant par TPETIT et s'achevant par FINTPETIT, situé s'il existe immédiatement après EXPERIENCE, doit être exécuté plusieurs fois (voir page IV-14). La technique utilisée sera la même : l'instruction FINTPETIT comporte un renvoi vers l'étiquette EXPERIENCE qui est effectué autant de fois que nécessaire.

L'arrêt de l'expérience peut survenir soit parce qu'il est explicitement prévu par une instruction STOP ou STOPA soit à la suite d'un diagnostic d'erreur. Il est obtenu par un saut vers une étiquette désignant les instructions ALGOL qui constituent l'épilogue de chaque programme. Outre l'édition d'informations diverses ces

instructions déterminent si une nouvelle simulation doit être entreprise. Si tel est le cas il est nécessaire de reprendre l'exécution avant même la première partie car des variantes peuvent y être mentionnées et certaines variables doivent être réinitialisées. Il y a alors débranchement vers le prologue d'initialisation.

Le schéma de principe de l'exécution d'un programme SIPHALGOL s'établit donc comme suit :



Dans certaines circonstances des instructions ne doivent pas être exécutées : c'est le cas si elles se rapportent à une simulation différente de celle qui est en cours ou si leur prise en compte est subordonnée à la réalisation des conditions formulées par des instructions QUAND ou DEPUIS, par exemple. C'est pourquoi un certain nombre d'indicateurs booléens sont utilisés : leurs valeurs sont fixées par les instructions posant les conditions et sont testées par toutes les autres instructions. Les conditions portant sur le numéro de simulation peuvent intervenir n'importe où dans le programme, les autres ne peuvent apparaître que dans la seconde partie.

Le programme d'interprétation proposé et décrit dans ce chapitre ne réalise qu'un nombre assez réduit de diagnostics d'erreur. On pourrait envisager l'écriture d'un autre programme assurant exclusivement ce travail d'analyse et auquel les programmes SIPHALGOL seraient soumis préalablement. Tout programme soumis à l'interprétation est supposé satisfaire aux règles d'écriture de SIPHALGOL. Seules des erreurs apparaissant à l'exécution (contenu négatif, référence à des valeurs non stockées, etc ...) sont signalées. En ce cas le type d'erreur est imprimé ainsi que le contenu de la mémoire au moment de l'erreur. Pour permettre la localisation de l'erreur un numéro est affecté à chaque instruction. Cette numérotation est à distinguer de trois autres numérotations indépendantes dont la justification sera donnée dans les paragraphes suivants et qui affectent respectivement les instructions TOUSLES, les instructions SYNTHETISE, INJECTER etc ..., et les instructions qui peuvent être suivies de l'instruction PUIS.

IV.2 - Organisation de la mémoire

Les information qui devront être rangées en mémoire seront nombreuses et diverses. On cherchera à organiser ce rangement pour d'une part occuper le moins de place possible et d'autre part pour permettre une recherche facile et une localisation rapide de ces informations. Ces deux impératifs ne sont d'ailleurs pas toujours conciliables et il faudra souvent se contenter d'un compromis.

Tant que le modèle n'a pas été défini on est dans l'ignorance du volume global d'informations ainsi que des tailles relatives des diverses classes d'informations qu'il faudra enregistrer. Ceci, joint au souci d'économiser la place disponible en mémoire conduit à envisager une organisation aussi dynamique que possible.

Par contre les programmes SIPHALGOL seront en général écrits par des utilisateurs ignorants des règles de l'ALGOL et dont il faut simplifier le travail au maximum. En particulier ils ne devront pas avoir de déclarations à effectuer. La meilleure solution semble donc être d'utiliser un seul tableau à une dimension de taille maximum et dans lequel seront rangées le plupart des informations. Ce tableau sera, par programme, divisé en plusieurs segments correspondant aux divers types de données. Un certain nombre de pointeurs, d'indicateurs et de mémoires de manoeuvre vont permettre l'utilisation de ces informations.

Celles-ci concernent :

- la structure du modèle (liaisons)
- les fonctions des blocs
- les paramètres attachés aux liaisons
- l'état du modèle à chaque instant

- Dans certains cas, tout ou partie de l'état passé du modèle, tant en ce qui concerne la structure que les fonctions.
- Les phénomènes autres que l'échange des produits (synthèse etc ...)

Plus précisément, elles devront permettre de répondre à chaque instant aux questions suivantes :

- Pour un produit donné, existe-t-il une liaison entre deux blocs donnés ?
- Quel est le type d'un bloc ?
- Quelles sont les valeurs des paramètres attachés à une liaison ?
- Comment, et au moyen de quelles valeurs, calculer l'évolution du système ?

Certaines de ces informations telles que par exemple le nombre et le type des blocs sont fixes, en valeur et en taille, pendant toute l'expérience, d'autres sont éssentiellement variables. Certaines sont relatives aux blocs, d'autres aux liaisons.

Les règles de SIPHALGOL sont telles que dans la première partie du programme les informations fournies ont toutes un caractère constant. On y apprend en effet :

- le nombre de produits intervenant dans le modèle
- le nombre de blocs
- le type de chaque bloc
- les liaisons existantes. Même si par la suite elles sont "coupées", elles restent virtuellement existantes et peuvent d'ailleurs être rétablies.
- le nombre de paramètres attachés à ces liaisons et la valeur constante de certains d'entre eux
- le cas échéant, le volume constant des blocs
- la durée de stockage des quantités qu'il faut mémoriser
- le pas d'intégration.

On peut classer ces informations en trois types :

- Les informations relatives à chaque bloc : volume, type ainsi que les blocs auxquels il est relié. Des informations qui seront élaborées par la suite telles que le contenu et les débits d'entrée et de sortie pourront également être rattachées à cette classe.

- Les informations relatives aux liaisons : essentiellement nombre et valeur des paramètres. Ici encore des valeurs obtenues lors de l'expérience de simulation (débit dans la liaison, paramètre variable) seront du même type.

- Enfin les quantités mémorisées et les temps de stockage.

La taille globale du premier groupe peut être facilement déterminée : en effet des les premières instructions on connait le nombre de blocs et le nombre de produits. En convenant de ranger les informations relatives à chaque couple produit-bloc sous un format fixe, la taille de mémoire nécessaire peut être calculée.

Par contre le volume des deux autres groupes ne peut être connu tant que la première partie du programme n'a pas été entièrement exécutée. En effet les éléments de description du modèle sont fournis dans un ordre absolument quelconque et rien ne permet de déterminer s'ils ont ou non été tous enregistrés. Ceci conduit à séparer la mémoire disponible en trois segments, correspondants aux trois types décrits ci-dessus. Les segments correspondants aux deuxième et troisième type auraient pu être réduits en un seul mais leur séparation rendra plus aisée la mise à jour du segment de troisième type.

IV.2.1. Le premier segment

Le segment 1 sera formé d'enregistrements de longueur fixe, associés à chaque couple produit-bloc, ainsi constitués :

- un nombre d'emplacements de mémoire égal au nombre total de blocs, qui indiquent par leur contenu si une liaison a été définie entre le bloc considéré et celui dont le numéro correspond au numéro de l'emplacement. L'absence de liaison est marquée par une valeur négative ou nulle (voir page IV.15). Une valeur positive, au contraire, indique l'existence de la liaison et constitue un index, renvoyant à l'enregistrement des informations sur la liaison (2ème segment). Par exemple, si cinq blocs ont été annoncés et que le troisième enregistrement est ainsi constitué :

0	2000	0	0	1973	...

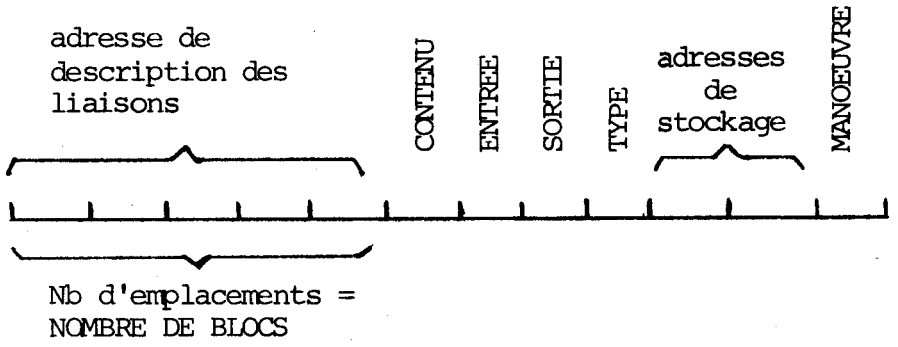
On pourra en conclure que le troisième bloc est l'origine d'une liaison vers le deuxième bloc ainsi que vers le cinquième. L'adresse de rangement des paramètres de chaque liaison étant respectivement 2000 et 1973.

- Ensuite quatre emplacements sont destinés à recevoir respectivement le contenu, le débit d'entrée, le débit de sortie et le type du bloc pour le produit considéré.

- Le contenu et les débits d'entrée et de sortie peuvent faire l'objet d'une demande de mémorisation. Deux emplacements sont réservés pour enregistrer les adresses éventuelles de ces stockages (3ème segment). Si aucune demande n'est faite la valeur enregistrée est nulle. L'exploitation de ces informations est décrite de façon plus détaillée page IV.13.

- Enfin une position supplémentaire est destinée à servir de mémoire de manoeuvre et à contenir des valeurs intermédiaires.

Chaque enregistrement est donc composé de N emplacements avec N = NOMBRE DE BLOCS+7



Pour chaque produit, tous les blocs sont ainsi décrits et les emplacements nécessaires sont au nombre de

NOMBRE DE BLOCS*(NOMBRE DE BLOCS + 7).

Le **volume de chaque bloc**, indépendant du produit, doit également être enregistré et NOMBRE DE BLOCS emplacements supplémentaires sont réservés à cet effet à la fin du premier segment. En fin de compte la longueur de ce segment est :

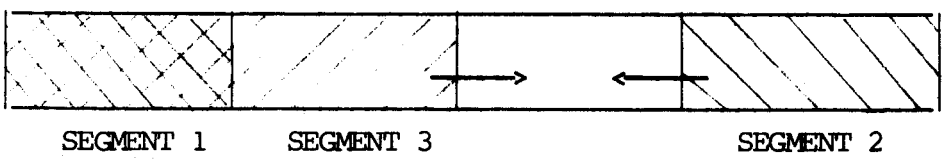
NOMBRE DE BLOCS*(NOMBRE DE PRODUITS*(NOMBRE DE BLOCS + 7) + 1)

Cette longueur ne dépend que du nombre de blocs et du nombre de produits, indications qui sont fournies en tête du programme. La place nécessaire peut donc être réservée et l'implantation des autres segments pourra utiliser l'espace restant.

Cette méthode présente deux inconvénients : en premier lieu l'absence de liaison entre deux blocs est représenté par l'enregistrement d'un zéro. Dans le cas de modèles "creux" c'est-à-dire ou les liaisons sont peu nombreuses, une place importante pourrait être gagnée en adoptant un format variable pour ranger ces informations et en supprimant les trous. D'autre part les nombres rangés sont aussi bien de type réel que de type entier, ce qui amène à utiliser un tableau de type réel pour ranger une majorité de nombres entiers. Là encore d'autres méthodes auraient permis de réduire cette perte de place. Mais ces méthodes diminueraient à coup sûr la souplesse des règles d'écriture du programme, en particulier en ce qui concerne l'ordre des instructions, et accroîtraient notablement le temps des calculs d'adresse nécessaires. Les modèles habituellement rencontrés en physiologie ne nécessitant pas une taille de mémoire très importante, le souci de minimiser le temps d'exécution nous a paru le plus important.

IV.2.2. Utilisation de la place disponible en mémoire

Les segments 2 et 3 vont être implantés dans la partie libre du tableau. Les instructions de stockage pouvant être mélées aux instructions de description des liaisons, les deux segments vont se développer simultanément et il faudra veiller à ce qu'ils ne se recouvrent pas. La taille du premier segment ayant été déterminée, on pourra implanter l'un des deux autres, par exemple le troisième, à la suite. A chaque instruction de mise en mémoire, qui peut intervenir à tout endroit de la première partie, la taille du segment 3 s'accroît et la place disponible pour le segment 2 diminue. Ce dernier sera donc implanté dans le sens des adresses décroissantes à partir de l'extrémité de la mémoire.



Les segments 2 et 3 vont à la rencontre l'un de l'autre et, si la taille du modèle n'est pas excessive, un espace libre subsiste entre eux à la fin de la première partie du programme. Ces emplacements inoccupés constitueront par la suite un quatrième segment où seront rangés les informations fournies dans la seconde partie du programme.

IV.2.3. Le deuxième segment

Le segment 2 contiendra donc les informations relatives aux liaisons. Pour chacune on trouvera un enregistrement de longueur variable ainsi formé :

1ère position : nombre de paramètres affectés à la liaison
2ème position : adresse éventuelle de stockage prolongé du débit dans la liaison
3ème position : valeur courante du débit dans la liaison (APPORT)
4ème position et suivantes : valeurs des paramètres.

. . . . .	P3	P2	P1	apport	adr. de stock.	nb. de param.

La longueur de chacune de ces zones est donc
NOMBRE DE PARAMETRES + 3.

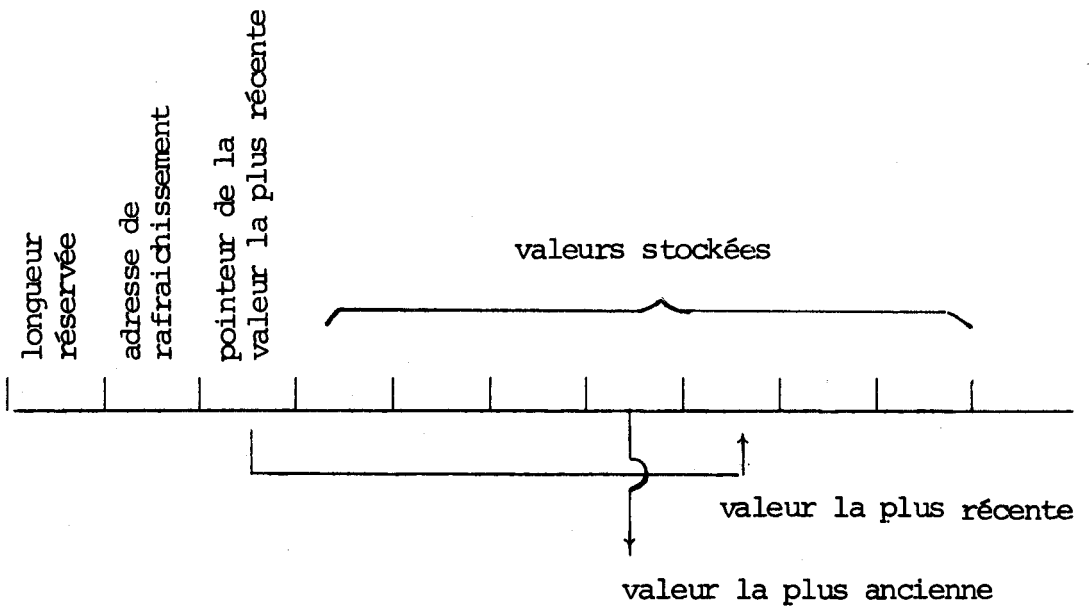
C'est l'adresse droite de la zone qui est enregistrée dans la position correspondante du 1er segment et qui permet de retrouver les paramètres de la liaison.

IV.2.4. Le troisième segment

Dans le 3ème segment, chaque fois qu'une instruction de stockage sera prise en compte il faudra réserver une place suffisante. Cette place ne peut être calculée que si on connait le pas d'intégration. En effet la mise en mémoire d'une quantité pendant X unités de temps signifie la mise en mémoire des X÷H valeurs calculées pendant ce temps. C'est pourquoi l'instruction PAS doit toujours être présente avant les instructions GARDER et PENDANT.

Pour la même raison on voit que le choix d'un pas d'intégration très petit, s'il améliore la précision des résultats, aura non seulement comme effet d'accroître le temps de calcul, mais également d'augmenter l'encombrement de la mémoire. Ce dernier inconvénient, qui apparait aussi lors de l'utilisation de blocs de type retard, n'est pas le moindre.

Une fois la place nécessaire calculée la réservation se fera simplement en déplaçant un index pointant le début de la zone libre. Trois positions supplémentaires sont réservées : dans la première on écrit la longueur de la zone utilisée, dans la seconde l'adresse (dans les segments 1 ou 2) où est disponible à chaque instant la valeur courante de la quantité stockée (adresse de rafraichissement) et dans la dernière un pointeur indiquant la position dans la liste de la valeur la plus récente.



La mise à jour consiste en un simple déplacement du pointeur de valeur récente. La valeur la plus ancienne est remplacée par la valeur trouvée à l'adresse de rafraichissement et devient la plus récente. En bout de liste le pointeur est réinitialisé en tête.

IV.2.5. Le quatrième segment

Lorsque la fin de la première partie du programme survient, les longueurs des trois premiers segments sont définies et ne seront plus modifiées. Seuls leurs contenus pourront varier. Par contre, dans la seconde partie, des informations seront apportées qui ne trouvent pas leur place dans la structure décrite jusqu'ici. Elles concernent les phénomènes autres que l'échange de produits entre les blocs, c'est-à-dire ceux dont rendent compte les instructions INJECTER, PERFUSER, SYNTHETISE et UTILISE.

Ces informations seront enregistrées dans la partie de la mémoire restée libre, constituant ainsi un quatrième segment. Pour chaque instruction de ce type, il faudra enregistrer plusieurs valeurs :

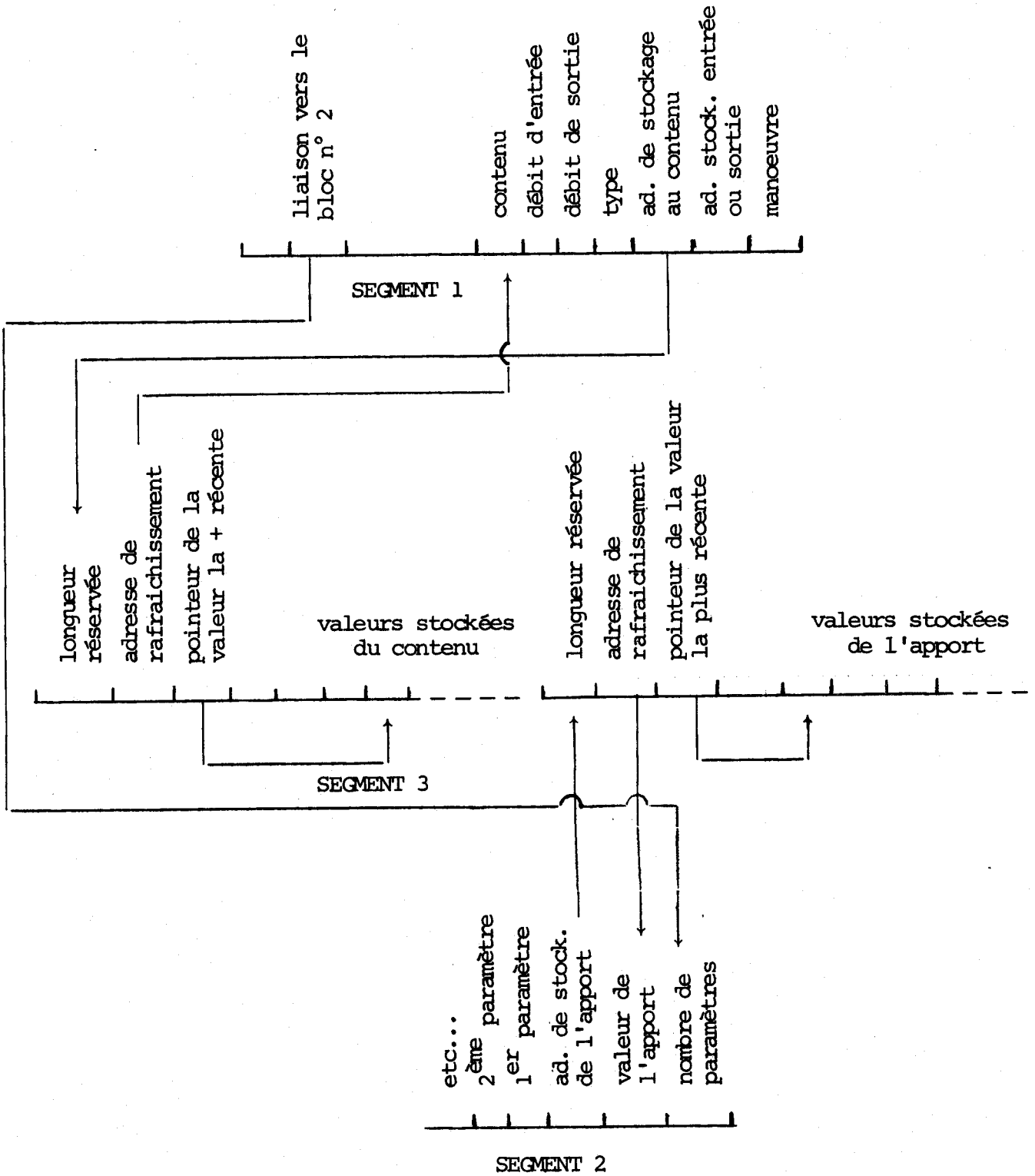
- le numéro du produit concerné
- le numéro du bloc concerné
- la quantité de produit en cause : en fait cette valeur est enregistrée deux fois car la méthode d'intégration étant d'ordre deux, chaque pas de l'intégration nécessite deux phases. A chaque phase la quantité est réévaluée et les deux valeurs doivent être enregistrées. C'est dans la procédure EXECUTION que toutes ces informations seront rassemblées et traitées.
- Enfin, pour assurer la compatibilité avec les méthodes d'intégration (voir description des procédures SYNTHETISE et INJECTER), un quatrième nombre permettant de repérer l'instruction qui est à l'origine de chaque enregistrement devra également être rangé en mémoire.

Au total les enregistrements sont donc formés de cinq nombres. A chaque évaluation de l'état du modèle, un balayage du segment 4 permettra de prendre en compte les renseignements qui y ont été rangés et le segment sera ensuite libéré pour permettre l'enregistrement de nouvelles valeurs.

La mémoire se présente donc en fin de compte de la façon suivante :

Le segment 4 est de longueur variable au cours de l'expérience de simulation et laisse une place vide plus ou moins importante.

Le schéma suivant rappelle les relations existant entre les différents enregistrements relatifs à un même bloc. Il décrit l'aspect des segments 1, 2 et 3 pour un bloc dont le contenu est maintenu en mémoire de façon prolongée ainsi que l'apport d'une liaison avec le bloc 2.



Les flèches représentent les chaînages reliant les enregistrements.

IV.3 - Les procédures

La mémoire étant ainsi organisée, il reste à la garnir. C'est le rôle des différentes procédures, qui constituent les mots de base de SIPHALGOL. Chacune range à la place voulue les informations qu'elle élabore. La dernière, EXECUTION, examine et tire partie de ces informations pour évaluer l'état du système au temps T. Un retour en arrière par débranchement vers l'étiquette EXPERIENCE, et de nouveaux appels aux procédures, modifient le contenu de la mémoire et permettent de calculer un nouvel état du système, et ainsi de suite.

Les pages qui suivent décrivent les procédures les plus importantes.

IV.3.1. Procédures de description statique

NBBLOCS, NBPRODUITS

Ces deux procédures qui fournissent le nombre de blocs et le nombre de produits intervenant dans le modèle, permettent de calculer la longueur du premier segment de mémoire et de déterminer l'adresse gauche du troisième segment.

PAS

La valeur du pas d'intégration ayant été fournie par cette procédure, les instructions de mise en mémoire prolongée qui apparaîtront ensuite pourront donner lieu à la réservation de place voulue. En effet la taille de la zone à réserver dépend de la valeur du pas.

PRODUIT, BLOC

Ces procédures positionnent un index INDEX 1 sur le début de la zone du premier segment, correspondant au produit et au bloc mentionnés. Les numéros de bloc et de produit sont en outre mémorisés.

COMPARTIMENT, RETARD, JOKER, EMBRANCHEMENT, RESERVOIR

Ces procédures ont le même effet que BLOC. En outre un indicateur de type est rangé dans l'emplacement prévu à cet effet.

ALIMENTE, AINSIQUE

Sont deux procédures équivalentes qui rangent dans l'emplacement du segment 1, correspondant à la liaison définie, l'adresse INDEX PARAMETRE de la zone du segment 2 où seront rangés les paramètres de la liaison.

PARAMETRE, ET

Les paramètres fournis par ces procédures sont enregistrés en séquence dans la zone d'adresse droite INDEX_PARAMETRE. Le nombre de paramètres de la liaison, enregistré dans la même zone, est incrémenté de 1.

En outre, un indicateur booléen RETARD 1, éventuellement positionné par la procédure ALIMENTE précédente, indique que le paramètre est le premier d'une liaison dont l'origine est un bloc de type retard. Dans ce cas la réservation dans le troisième segment d'une zone suffisante pour l'enregistrement des valeurs à mémoriser est effectuée. L'adresse de cette zone est enregistrée sous forme d'un paramètre caché, inaccessible au programmeur.
RETARD 1, reprend ensuite la valeur FAUX.

Cette procédure peut également apparaître dans la seconde partie lorsque des paramètres ont été spécifiés variables.

GARDER, PENDANT

Le corps de la procédure GARDER est inexistant. Son seul rôle est d'appeler la fonction, CONTENU, ENTREE, SORTIE ou APPORT, qui est son argument.

Ces fonctions ne prennent pas alors de valeur significative puisque l'expérience de simulation n'a pas encore commencé, mais cet appel a toutefois deux effets :

- l'adresse ADST où sera rangée l'adresse de stockage est mémorisée. Cette adresse est soit dans le segment 1 (CONTENU, ENTREE, SORTIE), soit dans le segment 2 (APPORT). Pour un même bloc et un même produit, elle est identique pour les fonctions ENTREE et SORTIE, car on ne stocke jamais que l'une ou l'autre.
- Un indicateur de fonction TYPE_FONCTION prend les valeurs : 0 pour CONTENU
  -1 pour SORTIE
  +1 pour ENTREE
  2 pour APPORT

  La procédure PENDANT utilise ces deux renseignements. La longueur de la zone nécessaire au stockage est calculée en fonction du pas d'intégration. L'adresse de stockage est rangée en ADST. Si la fonction à stocker est SORTIE l'adresse est stockée sous forme négative.
  En outre lorsque le stockage est demandé pour deux quelconques des trois fonctions CONTENU, ENTREE, SORTIE, et pour les mêmes bloc et produit l'un des stockages

effectivement réalisé concerne toujours le CONTENU.

Par exemple si les instructions :

GARDER (ENTREE (1,1)) ; PENDANT (15) ;

GARDER (SORTIE (1,1)) ; PENDANT (10) ;

ont été écrites, ENTREE sera stockée pendant 15 unités de temps mais le stockage de SORTIE sera systématiquement abandonné pour être remplacé par celui de CONTENU, pendant 10 + h unités de temps. Ainsi, grâce à la relation :

sortie $(t-t_1)$ = contenu $(t-t_1-h)$ + entrée $(t-t_1)$ - contenu $(t-t_1)$

les trois quantités seront accessibles dans la suite du programme.

CAPACITE

La valeur fournie en argument est rangée dans la position prévue en fin de premier segment pour recevoir le volume du bloc. Si l'argument est VARIABLE, cette instruction réapparaît également en 2ème partie.

IV.3.2. Les procédures de description dynamique

TPETIT, FAIRE, EGALE, FINTPETIT

L'objet de ces procédures est de définir l'état pré-initial du système en affectant des valeurs non nulles aux quantités dont la mise en mémoire prolongée a été demandée. Cet effet est obtenu en exécutant plusieurs fois les procédures FAIRE et EGALE situées entre TPETIT et FINTPETIT. A chaque exécution l'argument de FAIRE, qui est une des fonctions EXCONTENU, EXENTREE, EXSORTIE ou EXAPPORT, élabore l' adresse de la zone de stockage correspondante. La procédure EGALE détermine l'adresse exacte du rangement et y enregistre la valeur de son argument. Cet argument est une fonction d'un identificateur T1 qui est incrémenté du pas d'intégration à chaque passage. L'adresse de rangement est celle qui correspond à TINITIAL - T1. Tant que toutes les valeurs voulues n'ont pas été enregistrées la procédure FINTPETIT renvoie vers l'étiquette EXPERIENCE pour un nouveau passage.

Bien entendu ces instructions ne doivent être prises en compte que lors du premier passage dans la partie EXPERIENCE. A cet effet un indicateur booléen dont la valeur est testée par ces instructions, sera positionné à FAUX lors des passages suivants.

INITIALEMENT, CONTIENT, DEBITE, VERS, FININIT

L'état initial du modèle doit en général être précisé, tant en ce qui concerne

les contenus des différents blocs qu'en ce qui concerne les transits dont la valeur peut intervenir dans certaines expressions. C'est le rôle des procédures CONTIENT, DEBITE et VERS qui affecte soit le contenu des blocs soit l'apport des liaisons ainsi que, dans ce cas, l'entrée et la sortie des blocs correspondants.

Les instructions ne sont effectuées qu'au premier passage dans la partie EXPERIENCE. Un indicateur booléen les inhibe aux passages suivants.

CONTENU, ENTREE, SORTIE, APPORT

Les valeurs voulues sont extraites des positions correspondantes de la mémoire. Si l'argument donnant le produit est TOTAL (=0) une sommation est effectuée. Enfin les débits sont obtenus par division par le pas.

VOLUME, CONCENTRATION

La fonction VOLUME prend en général la valeur qui a été fournie précédemment par l'instruction CAPACITE. Toutefois, si cette valeur est nulle, en particulier quand elle n'a pas été fournie, elle est remplacée par le contenu total du bloc. La concentration en un produit est égale au rapport du contenu en ce produit et du volume du bloc.

EXCONTENU, EXENTREE, EXSORTIE, EXAPPORT

Lorsqu'une de ces quantités est appelée, un test préalable vérifie la compatibilité avec les stockages effectivement réalisés. En cas d'erreur, un message est émis et le programme interrompu. Sinon les valeurs correspondantes sont lues dans le troisième segment. Lorsque l'argument indiquant le produit est TOTAL, les valeurs correspondant à chaque produit sont ajoutées.

COUPER, EXCLURE, RETABLIR, REINCLURE

L'existence d'une liaison étant simplement indiquée par une valeur positive dans la position correspondante du premier segment, il suffira pour la supprimer de remplacer cette valeur par son opposée, négative. Le rétablissement de la liaison se fera de la même façon. EXCLURE (resp. REINCLURE) un bloc consiste simplement à supprimer (resp. rétablir) toutes les liaisons qui y parviennent ou en sont issues.

En outre les blocs exclus sont repérés par une valeur négative dans la mémoire de manoeuvre qui leur est affectée dans le premier segment. Ainsi la procédure EXECUTION ne tiendra plus compte de ces blocs pour certains traitements. Enfin, on verra dans la description d'EXECUTION que chaque évaluation de l'état du modèle se fait en deux phases. La structure ne devant pas changer entre les deux phases ces

instructions ne sont prises en compte que lors de la première.

SYNTHETISE, PERFUSER, UTILISE, INJECTER

Les rôles de ces quatre procédures sont pratiquement identiques. Ils consistent à enregistrer dans le quatrième segment les informations qui seront ensuite reprises par EXECUTION. D'autre part, ainsi qu'il a déjà été dit l'évaluation de l'état du modèle à chaque instant se fait en deux phases. Aucun phénomène ne doit débuter ni se terminer entre temps, sous peine d'obtenir des résultats absurdes. Or dans certains cas l'emploi d'instructions conditionnelles pourrait aller à l'encontre ce cette règle. C'est pourquoi dans la deuxième phase les instructions SYNTHETISE etc ... sont prises en compte, indépendamment de toutes autres conditions, si et seulement si elles avaient été prises en compte à la première phase. A cet effet, un numéro d'instruction est élaboré et enregistré avec les autres informations. Il permet de vérifier que l'instruction a déjà été effectuée. A chaque phase la quantité de produit en argument est enregistrée dans la position correspondante. Dans le cas particulier de l'injection, qui présente un caractère instantané et dont l'argument ne doit pas varier entre les deux phases, la quantité de produit injectée est systématiquement égale à celle fournie lors de la première phase.

QUAND, AUTEMPS, AUTREMENT, FINQUAND

Un nouvel indicateur booléen, VALIDE1, est utilisé pour indiquer si les conditions imposées par les instructions QUAND, AUTEMPS et AUTREMENT sont ou non vérifiées. Toutes les autres instructions, sauf SIMULATION (voir plus loin) testent sa valeur. Plusieurs groupes QUAND ... FINQUAND peuvent être imbriqués (10 au maximum). On utilise une pile booléenne, PILES_DES_QUAND, dans laquelle sont enregistrées pour chaque QUAND ou AUTEMPS la valeur primitive de VALIDE1 et la valeur de la condition imposée. La nouvelle valeur de VALIDE1 est obtenue par intersection logique. AUTREMENT remplace la tête de pile par sa négation. FINQUAND dépile deux positions et restitue la valeur précédente de VALIDE1.

DEPUIS, TOUSLES, JUSQUA, COMPRIS

Certaines instructions (instructions controlées) peuvent être assujetties à ne prendre effet que périodiquement, entre des valeurs extrêmes du temps. Dans ce but leur exécution est subordonnée à la valeur d'un autre indicateur booléen, VALIDE2. Sa valeur est déterminée, en fonction du temps, par les procédures DEPUIS, TOUSLES, JUSQUA et COMPRIS. En outre on utilise un tableau annexe, PROCH_EXEC, dont chaque position correspond à une instruction TOUSLES. A chaque appel de cette

instruction le tableau reçoit la prochaine valeur du temps pour laquelle l'instruction controlée devra être exécutée.

SIMULATION, A, PUIS

Plusieurs variantes du modèle ou de l'expérience peuvent être programmées et donner lieu à des exécutions successives. Un troisième indicateur booléen, VALIDE3, indique la validité ou l'invalidité des instructions, c'est-à-dire la coïncidence ou la non coïncidence entre le numéro de la simulation en cours et celui de la simulation demandée. Toutes les procédures, sauf QUAND, AUTREMENT et FINQUAND testent la valeur de VALIDE3. Chacune établit la valeur de cet indicateur pour l'instruction qui suit. La procédure RES est appelée systématiquement en tête de toutes les instructions qui peuvent être suivies d'un appel à la procédure PUIS. Chacune de ces instructions est numérotée et le nombre d'instructions PUIS qui la suivent immédiatement est enregistré dès le premier passage dans la position correspondante d'un tableau auxiliaire. Grâce à ce tableau et aux informations élaborées et transmises par SIMULATION, RES établit le numéro de la simulation correspondant à l'instruction et détermine si elle doit être exécutée. Un code entier de type de procédure est transmis à l'éventuelle instruction PUIS qui suit. Si cette instruction doit être effectivement prise en compte, un aiguillage renvoie, en fonction du type de procédure, vers l'appel de l'instruction voulue.

EXECUTION

Les processus auxquels s'applique ce système de simulation se traduisent mathématiquement par un système d'équations différentielles du premier ordre. Evaluer l'état du modèle et son évolution, c'est résoudre ces équations. Cette résolution est obtenue par une méthode numérique d'intégration. Dans le programme décrit ici, c'est une méthode de Runge-Kutta d'ordre 2, dite d'Euler-Cauchy, qui est utilisée. Une autre méthode, d'un autre rang, aurait pu être choisie. Il suffirait de modifier les instructions correspondantes de la procédure EXECUTION ainsi que la taille des réservations de mémoire du quatrième segment. On pourrait facilement imaginer que ces modifications soient déterminées par l'appel d'une procédure supplémentaire indiquant la méthode choisie.

La méthode d'Euler-Cauchy s'énonce de la façon suivante :

Soit $\frac{dZ}{dt}(t) = f(t, Z(t))$

le système différentiel à résoudre,
(u, v) les conditions initiales (Z(u) = v)

On calcule successivement :

$$Z_1 = v + h\ f(u, v)$$

$$Z_2 = v + \frac{h}{2}\ (f\ (u, v) + f(u + h, Z_1))$$

Z_2 est l'approximation retenue pour Z(u + h)

(h est le pas d'intégration, constant).

En réitérant le procédé on obtiendra des approximations successives de Z(u + 2h), Z(u + 3h) etc ...
Dans le cas des processus simulés, Z est en quelque sorte la fonction d'état du modèle. La méthode consiste alors, connaissant l'état Z du système au temps t, à calculer d'abord, avec une hypothèse de linéarité, une première approximation Z_1 de l'état au temps t + h. Avec la même hypothèse on obtient à partir de Z_1 une nouvelle évaluation de la variation de Z. La moyenne arithmétique de ces deux évaluations successives, appliquée à Z, fournit Z_2. On voit donc que le calcul se fait à chaque pas en deux phases : calcul de Z_1, puis de Z_2. Entre ces deux phases Z doit être mémorisée. Dans le cas de ce programme d'interprétation, chaque phase correspond à un appel de toutes les procédures constituant la deuxième partie du programme SIPHALGOL, suivi d'un appel à la procédure EXECUTION. Les tâches effectuées par cette procédure sont les suivantes :
Les entrées et sorties des différents blocs sont mises à zéro. Lors de la première phase les quantités transitant par chaque liaison (apport) sont également remises à zéro et les zones de stockage du segment 3 sont mises à jour.
Les mouvements de produit, tels que synthèse, injection etc... sont retrouvés dans le segment 4 et attribués aux entrées ou aux sorties correspondantes. Suivant la phase on utilise soit la première valeur fournie, soit la moyenne arithmétique des deux.

Débute ensuite le traitement des liaisons ordinaires, c'est-à-dire n'ayant pas comme origine un bloc embranchement et n'étant pas définies comme liaison de vidange. Pour chacune la quantité qui l'emprunte est calculée selon la formule associée au type du bloc origine.

Lors de la deuxième phase, la valeur définitivement utilisée est la moyenne arithmétique des valeurs calculées au cours des deux phases. Pour les blocs retard cette valeur est simplement lue dans la zone de stockage associée à ces blocs.

On y ajoute éventuellement la quantité de produit empruntant une voie à paramètre nul (type joker - voir la description des blocs retards). La valeur calculée est enregistrée et ajoutée aux entiers et sorties correspondantes.

Au cours de la première phase, le contenu est rangé dans la mémoire de manoeuvre associée à chaque bloc dans le premier segment. Au contraire il est rappelé lors de la seconde phase.

S'il existe des blocs retards ayant une voie joker, la quantité de produit qui l'emprunte est prélevée équitablement sur tous les enregistrements du segment 3 qui constituent les sorties futures du bloc (voir plus loin). Quand toutes les liaisons ordinaires ont été traitées, on peut s'intéresser aux liaisons de vidange des blocs joker. On les reconnait à ce qu'aucun paramètre n'y est associé. La quantité de produit qui les emprunte est égale à la différence entre le contenu du bloc et les quantités transitant par les autres liaisons issues du bloc. Tous les autres types de bloc ayant été traités, on peut s'attacher aux embranchements. (Il faut noter que par syntaxe, deux embranchements ne peuvent être reliés). Les quantités de produit transitant par les liaisons issues de ces blocs sont égales à la quantité entrant dans le bloc, multipliée par le paramètre de la liaison. Si celui-ci est absent on lui affecte le complément à 1 de la somme des autres.

D'autre part, lors de la deuxième phase, alors qu'on a calculé la valeur définitive des apports aux blocs retards, on répartit, selon les paramètres enregistrés dans le deuxième segment, cette valeur entre les différentes positions de la zone de stockage associée à chaque bloc dans le segment 3. On constitue ainsi pas à pas la valeur des sorties futures du bloc.

Enfin le nouveau contenu de chaque bloc est calculé en fonction du contenu précédent et des quantités ENTREE et SORTIE.

Certains problèmes peuvent surgir à cette étape. Le plus important a son origine dans l'évaluation séquentielle de l'effet de processus qui en réalité se déroulent simultanément. En particulier les quantités de produit en transit entre ces blocs dépendent en général de l'état du modèle au temps T c'est-à-dire essentiellement du contenu des blocs au temps T. C'est pourquoi il est nécessaire de ne modifier ces contenus qu'une fois tous les apports évalués. Mais on pourra alors obtenir des contenus négatifs. Ceci peut être dû à une programmation abhérante (paramètres incohérents par exemple) mais aussi à un choix d'un pas d'intégration trop grand donnant une approximation trop grossière d'une valeur qui devrait être en réalité positive mais proche de zéro. Une correction rétrospective

des calculs pourrait être envisagée mais conduirait à un traitement si laborieux qu'elle ne présente guère d'intérêt. On a préféré émettre un message signalant simplement l'anomalie. Si un tel message est émis plus de dix fois au cours d'une même expérience, elle est interrompue. Cette méthode permet dans une certaine mesure de ne pas tenir compte des erreurs dues à la grossièreté des approximations et qui tendent souvent à se corriger au cours de l'évolution de l'expérience. En outre les blocs exclus ne sont pas pris en compte à ce niveau.

Avant de retourner vers l'étiquette EXPERIENCE et de reprendre l'appel des autres procédures pour une itération supplémentaire, on procède à la fin de la seconde phase à une réinitialisation du quatrième segment.

D'autre part à la fin de la première phase le temps est incrémenté de la valeur du pas. En effet c'est bien $f(u + h, Z_1)$ qui apparait dans la formule de RUNGE-KUTTA. L'état du modèle calculé à ce moment n'est cependant pas celui qui correspondra définitivement à la valeur du temps ainsi fixée. Mais ce n'est déjà plus celui de la valeur précédente. Il serait illogique de continuer à faire figurer cette dernière dans les expressions où elle peut intervenir.

Chapitre V

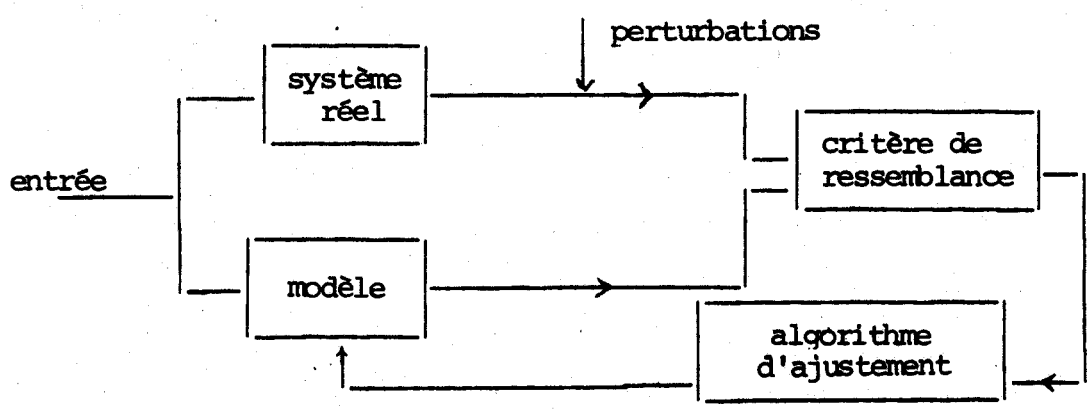
L'AJUSTEMENT DES PARAMETRES

La programmation d'un modèle en SIPHALGOL, telle qu'elle vient d'être décrite nécessite la connaissance de tous les aspects du modèle, ou au moins le choix d'hypothèses suppléant au manque d'information. En effet la structure du modèle doit être totalement explicitée et les fonctions qu'elle supporte doivent également être fournies et précisées par tous les paramètres nécessaires. Le problème alors résolu par les méthodes d'intégration numérique est le problème direct de déterminer la réponse du système simulé à une certaine entrée.

Mais bien souvent certains aspects du modèle sont peu ou mal connus et le problème, inverse, est alors de déterminer ces aspects inconnus de façon que la réponse du modèle soit aussi proche que possible de la réponse du système réel.

Ceci est particulièrement vrai dans le cas des systèmes physiologiques où l'expérimentation et la mesure, difficiles, permettent rarement d'obtenir des informations très complètes. Si la structure est en général assez bien connue, les valeurs des paramètres numériques sont souvent fournies de façon très imprécise. On cherchera donc les valeurs de ces paramètres qui déterminent la meilleure concordance entre les observations que l'on peut faire sur le système réel et les valeurs obtenues pour les grandeurs correspondantes du modèle.

Il convient d'ailleurs de remarquer que les observations faites sur le système réel sont, surtout dans le domaine physiologique, entachées de perturbations dues soit à l'imprécision des mesures, soit à des phénomènes secondaires. Le schéma suivant représente alors la démarche suivie :



Le grand intérêt de ces problèmes nous a conduit à tenter d'intégrer à SIPHALGOL des procédures permettant l'ajustement des paramètres.

Formulation du problème :

Le système à simuler peut être décrit par l'équation différentielle :

$$y' = f(y, a, u(t))$$

où y est un vecteur de n variables d'état, u un vecteur de r entrées, c'est-à-dire des valeurs connues, et a un vecteur de m paramètres inconnus.

Les observations sur le système réel sont données par un vecteur $z = Dy + Eu$ (E et D sont des matrices de dimension $q \times n$ et $q \times r$). Si les perturbations ne sont pas négligeables les observations seront $z' = z + d$.

De même le modèle est décrit par : $\boxed{x' = g(x, \alpha, u(t))}$

α est le vecteur-paramètre du modèle et x le vecteur d'état du modèle.

Le critère d'erreur pourra être de la forme :

$$J(T, \alpha) = \int_0^T (x - y)\, W(x - y)\, dt$$

où W est une matrice de pondération appropriée.

On cherchera alors la valeur α^* de α qui minimise J.

V.1 - Quelques algorithmes

Nous ne signalerons que pour mémoire les cas où une expression analytique de α^* peut être obtenue. Parmi les autres techniques on peut distinguer :

V.1.1. Les techniques de recherche

Des valeurs successives de α, notées α^1, α^2 ... etc sont calculées soit au hasard, soit à l'aide de certains critères plus ou moins fins. Des tests de comparaisons permettent de déterminer la valeur minimum de J et le paramètre correspondant. La technique de recherche peut être soit "brutale", c'est-à-dire procédant par énumération exhaustive de toutes les combinaisons de paramètres, soit procéder par élimination. Par exemple, si la fonction J est supposée ne présenter qu'un minimum, les valeurs obtenues pour $J(\alpha^{n-k})$, $J(\alpha^{n-k+1})$, ..., $J(\alpha^{n-1})$ peuvent montrer par leur disposition que certaines régions de l'espace des paramètres peuvent être éliminées pour la recherche de α^n.

Différentes techniques (Bolzano, Fibonacci etc ...) ont été proposées. Des méthodes dites "d'escalade" peuvent également être utilisées :

- méthodes de relaxation où un sous ensemble de paramètres étant fixe on cherche une approximation du minimum local pour les paramètres restants. Le procédé est réitéré en faisant varier le sous ensemble fixé.

- Méthodes de recherche au voisinage, où le minimum local est recherché pour quelques points au voisinage de α^n.

Toutes ces méthodes nécessitent de nombreuses évaluations de l'erreur donc de nombreuses résolutions des équations. Elles sont donc coûteuses sur calculateur digital et plus appropriées aux calculateurs hybrides. En outre elles présentent souvent de graves défauts de convergence pour certaines configurations de la fonction $J(\alpha)$.

V.1.2. Les méthodes utilisant le gradient

On y recherche la valeur de α pour laquelle le vecteur gradient

$$\nabla_\alpha J = [\frac{\partial J}{\partial \alpha_1} , \frac{\partial J}{\partial \alpha_2} , \ldots \frac{\partial J}{\partial \alpha_m}] \text{ est nul.}$$

L'algorithme de détermination des approximations successives de α^* est alors :

$$\alpha^{n+1} = \alpha^n - K \nabla_\alpha J(\alpha^n)$$

où K est une matrice.

C'est dans le choix de K que se distinguent les différentes méthodes de gradient :

Plus grande pente : $K = kI$

Newton-Raphson : $K = \dfrac{J(\alpha)}{||\nabla_\alpha J(\alpha)||^2}$

Newton : $K = [\dfrac{\partial^2 J}{\partial \alpha_j \partial \alpha_k}]^{-1} = (\dfrac{\partial \nabla J(\alpha^n)}{\partial \alpha})^{-1}$

Gauss-Newton : $K = [\int_0^T 2\nabla x \nabla x^T dt]^{-1}$ (si le critère est celui des moindres carrés).

etc ...

Ces méthodes peuvent d'ailleurs être modifiées et accélérées (PARTAN, FLETCHER-POWELL).

V.1.3. Les méthodes de recherches au hasard

On peut également envisager l'algorithme :

$$\alpha^{n+1} = \alpha^n + \Delta\alpha^n$$

où $\Delta\alpha^n$ est un vecteur dont les composantes ont été tirées au hasard (méthode de Monte Carlo). Le critère d'erreur est calculé et si on constate une amélioration le nouveau paramètre est accepté et ainsi de suite.

Dans certaines variantes l'incrément $\Delta\alpha^n$ peut être corrigé en fonction des échecs ou des réussites précédentes. On a pu montrer que lorsque J vérifiait certaines conditions et que le nombre de paramètres était important (> 4) la convergence des méthodes du hasard était meilleure que celle des méthodes de gradient et que les temps de calcul étaient également plus courts.

V.2 - L'identification stochastique

Les méthodes décrites précédemment étaient en fait valables pour les systèmes où les perturbations sur les mesures expérimentales sont négligeables. Ce ne sera en général pas le cas pour les processus physiologiques où au contraire de nombreuses circonstances, liées soit aux conditions d'expérience soit à la nature même des phénomènes, peuvent venir perturber les mesures.

On ne pourra plus alors utiliser ces méthodes puisque le vecteur z n'est plus disponible mais que seul $z' = z + d$ pourra être mesuré.

On peut tenter de remédier à cet inconvénient en calculant une estimation de z par la moyenne de nombreux z', résultats d'expériences identiques. Mais cette répétition des expériences est souvent difficile ou impossible en physiologie et il faut se résoudre à travailler sur le vecteur perturbé z'. C'est l'objet de l'approximation stochastique.

Dans un premier temps nous nous limiterons au cas scalaire, c'est-à-dire où un seul paramètre est à évaluer. Soit $J(\alpha)$ le critère d'erreur (par exemple $J(\alpha) = \int_{T_1}^{T_2} e(\alpha, t)^2 \, dt$). α étant fixé, $J(\alpha)$ est une variable aléatoire qui possède une certaine densité de probabilité $p(r)$:

$$Pr\,[r \leq J(\alpha) \leq r + dr] = p(r)\,dr$$

L'espérance mathématique de $J(\alpha)$ est la fonction de régression :

$$M(\alpha) = \int_{-\infty}^{+\infty} rp(r)\,dr, \text{ qu'on peut encore écrire } M(\alpha) = \int_{-\infty}^{+\infty} r\,dH(r/\alpha)$$

avec $H(r/\alpha) = Pr\,[J(\alpha) \leq r]$ pour α fixé.

Ni la nature exacte de $H(r/\alpha)$ ni celle de $M(\alpha)$ ne sont connues.

Kiefer et Wolfowitz proposent un algorithme permettant la résolution du problème :

Trouver α^* tel que $M(\alpha^*) < M(\alpha)$, $\forall \alpha \neq \alpha^*$

Cet algorithme est assez voisin de celui proposé par Robbins et Monro pour résoudre $M(\alpha^*) = 0$ et qui est le suivant :

α^1 est choisie arbitrairement, éventuellement au hasard. On calcule ensuite :

$$\alpha^{n+1} = \alpha^n - d_n J_n$$

où $\{d_n\}$ est une suite de nombres réels et $\{J_n\}$ une suite de variables aléatoires dont la distribution pour α^n donné est la même que celle de $J(\alpha^n)$. Si $M(\alpha)$ vérifie certaines conditions de régularité (cf. [43] et [54]) et si $\{d_n\}$ satisfait à

$$d_n \geq 0 \qquad \sum d_n^2 < \infty \qquad \sum d_n = \infty$$

alors on peut démontrer que α^n converge vers α^* en probabilité c'est-à-dire :

$$\forall \varepsilon, \lim_{n\to\infty} \Pr [|\alpha^n - \alpha^*| \geq \varepsilon] = 0$$

Il converge aussi vers α^* avec la probabilité 1 ;

$$\Pr [\lim_{n\to\infty} \alpha^n = \alpha^*] = 1$$

Enfin on prouve également une convergence au sens des moindres carrés :

$$\lim_{n\to\infty} E[(\alpha^n - \alpha^*)^2] = 0$$

L'algorithme présenté par Kiefer et Wolfowitz pour résoudre $M(\alpha^*) = \min_{\alpha} M(\alpha)$, et qui sera utilisé pour l'ajustement des paramètres, est dérivé du précédent. α^1 étant choisi arbitrairement on calcule ensuite

$$\alpha^{n+1} = \alpha^n + a_n \frac{(r_{2n} - r_{2n-1})}{c_n}$$

où $\{\alpha_n\}$ et $\{c_n\}$ sont deux suites de nombres réels positifs telles que :

$$\lim_{n\to\infty} c_n = 0 \qquad \sum_{n=1}^{\infty} a_n = \infty$$

$$\sum_{n=1}^{\infty} a_n c_n < \infty \qquad \sum_{n=1}^{\infty} \left(\frac{a_n}{c_n}\right)^2 < \infty$$

et r_{2n-1} et r_{2n} sont deux variables aléatoires indépendantes ayant respectivement

des distributions de probabilité $H(r \mid \alpha^n - c_n)$ et $H(r \mid \alpha^n + c_n)$, c'est-à-dire celles de $J(\alpha^n - c_n)$ et $J(\alpha^n + c_n)$

Afin d'assurer la convergence de la méthode les auteurs font les remarques suivantes sur la fonction M :

- Si elle existe la dérivée de M en α^* est nulle. Aussi peut on supposer qu'elle reste suffisamment faible au voisinage de α^*.
- Par contre hors d'un voisinage de α^* la valeur absolue de cette dérivée doit être bornée inférieurement. En effet une fonction trop "plate" conduirait à une convergence très lente vers l'optimum.
- Enfin une élévation trop brutale par endroit de la fonction pourrait amener des valeurs de α^n très éloignées de α^* et même, α^n pourrait tendre vers l'infini.

Ces considérations amènent à formuler pour M des conditions de régularité, qui d'ailleurs n'imposent pas que M soit dérivable ni même continue. Ce sont les suivantes :

1 - $\exists\ \beta > 0$ et $B > 0$ tels que
$|\alpha'-\alpha^*| + |\alpha''-\alpha^*| < \beta \Longrightarrow |M(\alpha')-M(\alpha'')| < B\,|\alpha'-\alpha''|$

2 - $\exists\ \rho > 0$ et $R > 0$ tels que
$|\alpha'-\alpha''| < \rho \Longrightarrow |M(\alpha')-M(\alpha'')| < R$

3 - $\forall\ \delta > 0\ \exists\ \Pi(\delta)$ tel que
$$|\alpha-\alpha^*| > \delta \Longrightarrow \inf_{\frac{\delta}{2} > \varepsilon > 0} \frac{|M(\alpha+\varepsilon)-M(\alpha-\varepsilon)|}{\varepsilon} > \Pi(\delta)$$

Ces conditions peuvent sembler sévères mais sont en fait souvent vérifiées, au moins dans un voisinage de α^*, dans la plupart des problèmes réels.
L'algorithme proposé présente incontestablement une forte ressemblance avec une méthode de gradient.
Le gain K serait alors $2a_n$ et $\frac{r_{2n}-r_{2n-1}}{2c_n}$ serait une estimation du gradient. A ceci près que le gain et le paramètre de perturbation décroissent au cours des itérations successives, l'analogie est complète en l'absence de bruit.

En choisissant par exemple $\begin{cases} r_{2n} = J(\alpha^n + c_n) \\ r_{2n-1} = J(\alpha^n - c_n) \end{cases}$

Cet algorithme pourra s'appliquer à l'estimation d'un paramètre inconnu d'un modèle.
Le choix de suites a_n et c_n judicieuses pourront influencer favorablement la convergence. On pourra prendre par exemple

$a_n = \frac{A}{n}$ et $C_n = \frac{C}{n^p}$, où A et C sont des constantes et p un nombre positif inférieur à 1/2.

A et C peuvent alors être adaptés par essais successifs pour obtenir une convergence satisfaisante.

Le même algorithme peut être étendu au cas de m > 1 paramètres à ajuster. α est alors un vecteur dont les composantes sont les paramètres inconnus et α^{n+1} est obtenu encore par la formule

$$\alpha^{n+1} = \alpha^n - a_n \frac{r_{2n} - r_{2n-1}}{c_n}$$

où cette fois r_{2n} et r_{2n-1} sont des vecteurs dont les composantes sont :

$$r^i_{2n} = J(\alpha^n + c_n e_i)$$
$$r^i_{2n-1} = J(\alpha^n - c_n e_i) \qquad (e_i = [0,0,\ldots o,1,0\ldots 0])$$

Les suites a_n et c_n doivent satisfaire aux mêmes conditions. Toutefois le choix en est plus délicat si l'on veut assurer une convergence assez rapide pour tous les paramètres.

V.3 - Application à SIPHALGOL

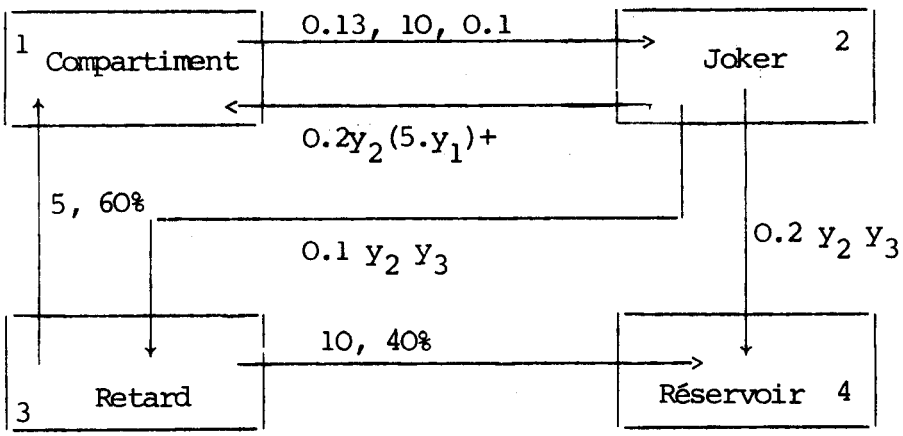
Il semble bien que, dans le cas des processus concernés par SIPHALGOL, ce soit la méthode d'approximation stochastique qui convienne le mieux. Aussi avons nous tenté de compléter SIPHALGOL par des procédures permettant de ne fournir, pour certains paramètres, qu'une valeur approchée. Des valeurs expérimentales d'une variable du modèle sont alors également fournies et le système, utilisant la méthode décrite au paragraphe précédent détermine une valeur ajustée du ou des paramètres.

Cet algorithme, comme d'ailleurs tous ceux qui ont été présentés jusqu'ici, nécessite plusieurs résolutions du système d'équations différentielles. Le nombre de ces résolutions dépend de la rapidité de la convergence et donc, en particulier, des suites a_n et c_n.

Lorsqu'il y a plus d'un paramètre à ajuster, on a vu qu'il était difficile d'optimiser ces suites et la convergence est souvent lente. D'autre part il ne semble pas qu'un critère d'arrêt raisonnable ait été proposé pour cette méthode et on est réduit à effectuer un nombre maximum d'itérations. Toutes ces raisons font que l'ajustement est souvent long.

Or la souplesse de présentation au modèle en SIPHALGOL se paie par la longueur de l'interprétation.

Par exemple nous avons étudié le modèle représenté par le schéma suivant :



où y_i représente le contenu du bloc i.

Ce modèle n'a aucune signification réelle et n'a été conçu que pour tester SIPHALGOL dans des relations entre blocs relativement complexes.

Le programme SIPHALGOL s'établit ainsi :

```
NBBLOCS(4); PAS(0.33);
COMPARTIMENT(1); ALIMENTE(2); PARAMETRE(0.13);
ET(10); ET(0.1);
JOKER(2); ALIMENTE(1); PARAMETRE(VARIABLE);
AINSIQUE(3); PARAMETRE(VARIABLE);
AINSIQUE(4); PARAMETRE(VARIABLE);
RETARD(3); ALIMENTE(1); PARAMETRE(5); ET(0.6);
AINSIQUE(4); PARAMETRE(10); ET(0.4);
RESERVOIR(4); SANSISSUE;
EXPERIENCE:
INITIALEMENT; BLOC(1); CONTIENT(15);
```

```
BLOC(3); CONTIENT(5);
FININIT;
DEPARTA(0); STOPA(40);
QUAND(CONTENU(1,1)>5); COUPER(2,1);
AUTREMENT; RETABLIR(2,1);
FINQUAND;
BLOC(2); VERS(1); PARAMETRE(0.2*CONTENU(2,1)*(5-CONTENU(1,1)));
VERS(3); PARAMETRE(0.1*CONTENU(2,1)*CONTENU(3,1));
VERS(4); PARAMETRE(0.2*CONTENU(2,1)*CONTENU(3,1));
TRACER(CONTENU(2,1) + CONTENU(3,1));
ENFONCTIONDE(T);
EXECUTION;
```

L'exécution de ce programme sur la Bull M40 demande 20 secondes non compris la compilation ni le tracé de courbe. Puisque l'expérience dure 40 unités de temps et que le pas choisi est de 0.33, cette durée correspond donc à 1/6 sec. par pas.

Nous avons traité le même problème par la méthode traditionnelle consistant à établir dans un premier temps le système d'équations puis à établir le programme ALGOL.

Les équations différentielles sont les suivantes :

$$\frac{dy_1}{dt}(t) = -0.13(y_1(t)-10)^+ - 0.1 + 0.2y_2(t)(5-y_1(t))^+ + 0.06y_2(t-5)y_3(t-5)$$

$$\frac{dy_2}{dt}(t) = -0.2y_2(t)(5-y_1(t))^+ - 0.3y_2(t)y_3(t) + 0.13(y_1(t)-10)^+ + 0.1$$

$$\frac{dy_3}{dt}(t) = 0.1y_2(t)y_3(t) - 0.06y_2(t-5)y_3(t-5) - 0.04y_2(t-10)y_3(t-10)$$

$$\frac{dy_4}{dt}(t) = 0.2y_2(t)y_3(t) + 0.04y_2(t-10)y_3(t-10)$$

Le programme résultant comporte environ 30 instructions ALGOL et son exécution a demandé 8.5 secondes, soit 2.3 fois moins que le programme SIPHALGOL. Il faut noter que ce modèle comporte peu de blocs mais que le réseau des liaisons est assez dense. Dans des modèles plus creux c'est-à-dire comportant un grand nombre de blocs pour une proportion moins grande de liaisons ce rapport peut s'élever jusqu'à 5. Lorsqu'une seule résolution du système est nécessaire le bénéfice de temps réalisé

en évitant d'expliciter les équations et par la simplicité de la programmation est encore appréciable.

Ceci n'est plus aussi manifeste dans le cas de l'ajustement des paramètres où il semble plutôt que l'écriture d'un programme particulier à chaque problème, en permettant le choix de la méthode la plus adaptée et l'utilisation totale des éventuelles particularités du système, soit plus économique et plus justifiée.

Peut-être la mise au point d'autres types de méthodes d'ajustement pourra-t-elle dans l'avenir modifier cet état de chose. On pourrait par exemple songer à modifier dynamiquement les paramètres à ajuster au cours de chaque expérience de simulation et ainsi réduire le nombre d'itérations nécessaires. Nous avons fait quelques tentatives en ce sens qui ont montré d'une part que la justification théorique de telles méthodes est difficile et d'autre part que leur éventuelle convergence dépend fortement du type de système simulé.

En outre il faut signaler, bien que ce soit assez secondaire qu'à l'échelon de la programmation et de l'intégration de l'ajustement à SIPHALGOL quelques difficultés apparaissent. Les valeurs numériques de la variable de référence doivent être lues sur un support externe et il est nécessaire d'introduire de nouvelles instructions précisant la présentation et la nature de ces informations. Elles peuvent d'autre part être nombreuses et nécessiter un enregistrement sur un fichier externe (bande ou disque). Il en résulte des procédures parfois lourdes à manipuler et dont l'interprétation pourra varier selon le compilateur ALGOL et le matériel utilisés.

Il semble donc bien qu'en l'état actuel des choses on atteigne là une limite des systèmes tels que SIPHALGOL qui seraient plutôt destinés à une utilisation épisodique et courte par des expérimentateurs non informaticiens en vue de la vérification d'hypothèses, où pour une représentation commode, en vue de l'enseignement par exemple, de phénomènes connus.

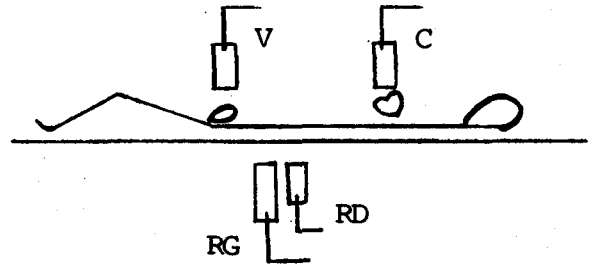
C'est pourquoi il est important que la programmation d'un tel système dans un langage évolué suffisamment répandu, permette sa mise en oeuvre quasi-immédiate sur un grand nombre de matériels.

Chapitre VI

UNE APPLICATION : SIMULATION DU NEPHROGRAMME ISOTOPIQUE

VI.1 - Principe et description physiologique

Le Nephrogramme Isotopique consiste, en son principe, en un enregistrement de la radioactivité de chaque rein durant les minutes qui suivent l'injection intraveineuse d'une petite quantité d'un produit radioactif, en l'occurrence l'Hippuran marqué à l'Iode 131. On enregistre souvent simultanément les radioactivités cardiaques et vésicales.



Ces mesures se traduisent par des courbes dont la forme permet d'obtenir certains renseignements sur l'activité rénale.

Les nephrogrammes présentent dans les cas normaux un aspect caractéristique (figure 1) :

- Deux premiers segments (I et II), tous deux ascendants, mais dont le premier est de courte durée et de forte pente. Ils sont séparés par une inflexion plus ou moins marquée de la courbe.
- Un troisième tronçon (III), descendant celui-là, séparé du deuxième par un sommet plus ou moins aigu.

La courbe cardiaque est décroissante et on peut régulièrement avec de bons résultats la décomposer en une somme de deux exponentielles.

La courbe vésicale présente dès le début de l'enregistrement une faible élongation qui reste pratiquement constante jusqu'aux instants suivant le sommet du nephrogramme. Puis elle croît, assez brutalement d'abord, plus lentement ensuite jusqu'à la fin de l'épreuve.

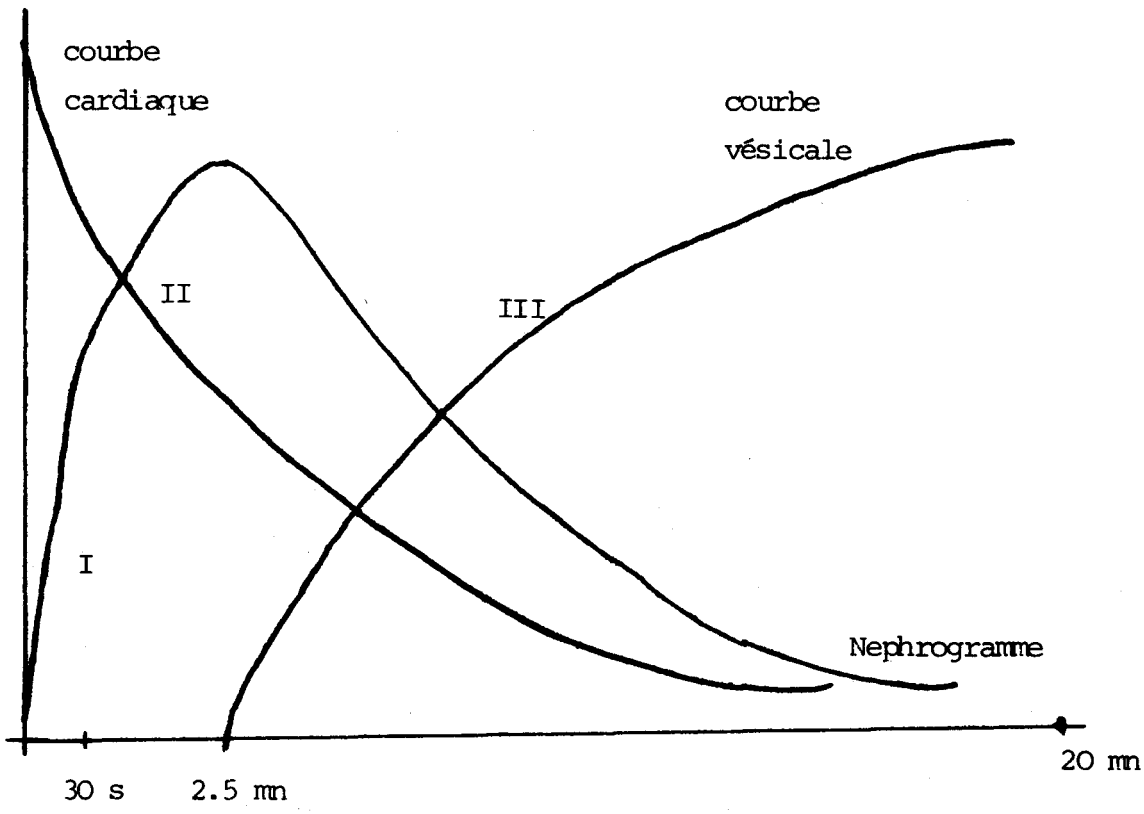


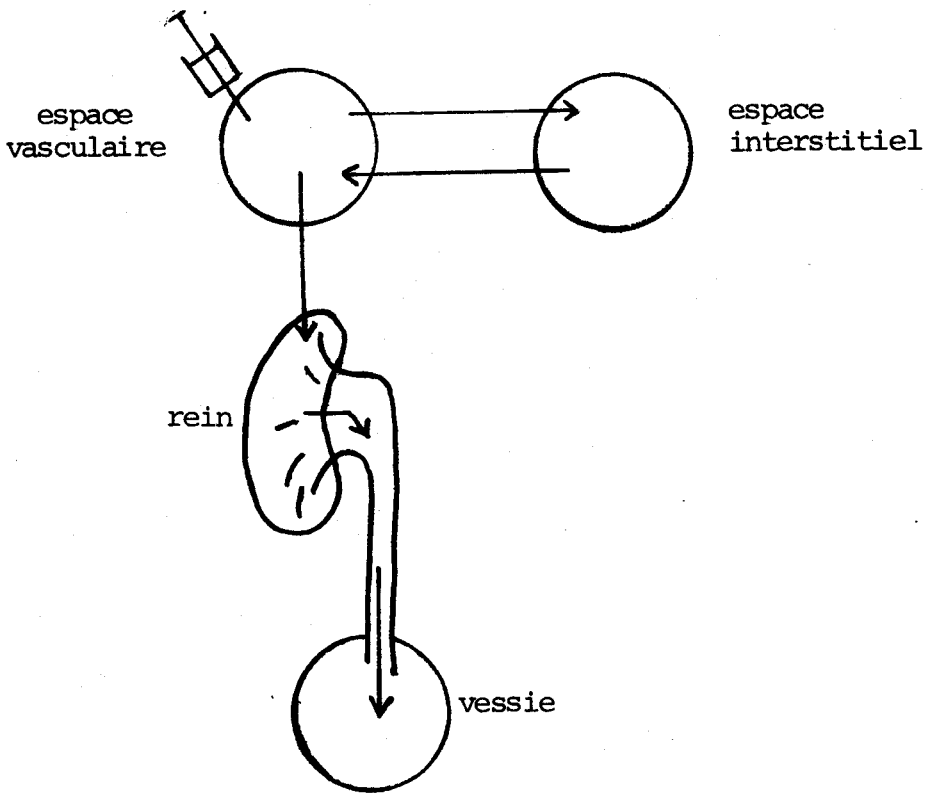
FIGURE I - Nephrogramme normal

Cette décomposition des courbes est classiquement interprétée comme le reflet de phénomènes successifs mais qui peuvent se superposer légèrement :

- le produit radioactif injecté envahit rapidement les territoires vascularisés observés par la sonde rénale et détermine le tronçon I, de forte pente.
- L'élimination rénale est la cause d'une accumulation du produit dans le parenchyme et les cavités pyélocalicielles. La courbe continue donc de croître, bien que moins rapidement (segment II).
- Le segment descendant III, normalement concave vers les temps croissants, traduit l'évacuation vers la vessie et le cheminement dans le rein et le bassinet d'une urine de moins en moins radioactive. Il coïncide avec l'élévation de la courbe vésicale.

Dans certains cas pathologiques on observe des aspects tout différents : diminution de la pente du segment II, rupture de pente, retard d'apparition du sommet, décroissance ralentie du troisième segment, inversion de sa concavité, etc ...

La physiologie de l'appareil rénal suggère le mécanisme d'épuration suivant :



Dès les premiers instants le produit injecté se répand en quelques 15 à 20 secondes, c'est-à-dire pratiquement instantanément, dans l'espace vasculaire. A partir de celui-ci, il diffuse vers les espaces interstitiels, selon un mode bi-compartimental de type mamillaire, à l'origine des deux exponentielles de la courbe cardiaque. Ceci explique également le palier initial de la courbe de vessie, alors que celle-ci ne contient pas encore d'urine radioactive.

Il est simultanément éliminé vers le rein, en quantité proportionnelle à la concentration sanguine. La secrétion rénale correspond schématiquement à trois phénomènes qui se déroulent simultanément dans chaque nephron. Ceux-ci, au nombre d'un million environ pour chaque rein, sont constitués d'un glomérule prolongé par un tubule qui débouche dans le bassinet.

Il se produit :

a - une filtration du plasma sanguin au niveau des glomérules,
b - une réabsorption de l'eau et une concentration de l'urine dans le tubule,
c - une captation, par les cellules tubulaires, de l'Hippuran qui est ensuite excrété dans la lumière du tubule.

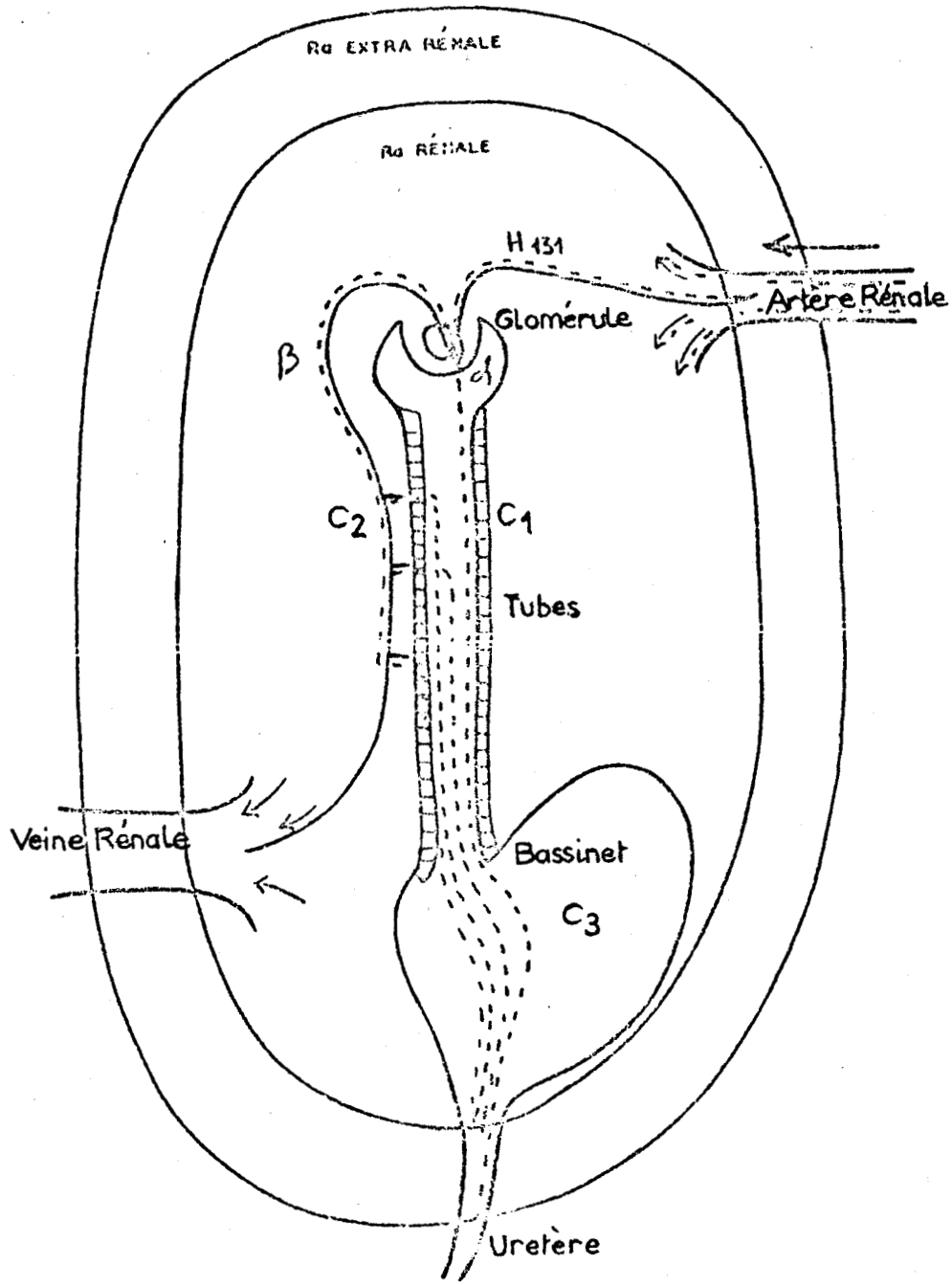


FIGURE II - Schéma mnémotechnique du champ d'exploration d'un collimateur rénal

Le schéma mnémotechnique de la figure II résume l'évolution du nephrogramme : l'hipurran radioactif parvient au rein, par l'artère rénale, environ 10 secondes après l'injection intraveineuse du produit. L'artère rénale se divise en un système capillaire à l'intérieur de chaque glomérule, où a lieu le phénomène de filtration. Le filtrat chemine le long du tubule et s'enrichit en Hippuran du fait de la sécrétion et de la réabsorption de l'eau. Le produit s'accumule ensuite dans

le bassinet avant d'être déversé dans la vessie. Le segment II correspond donc à ces phénomènes qui s'opèrent dans le rein normal simultanément dans tous les tubules. Le sommet survient au moment où l'urine radioactive commence à être déversée dans la vessie et coïncide avec le début de l'ascencion du diagramme vésical.

En outre les collimateurs débordent plus ou moins largement des reins et la radioactivité enregistrée provient de l'Hippuran qui se trouve

a - dans les vaisseaux rénaux et extra-rénaux,
b - dans les espaces interstitiels,
c - dans les voies d'excrétion : tubules et bassinets.

Le produit qui emprunte la voie glomérulaire (20 % environ dans les cas normaux) parviendra au bassinet, constitué par l'ensemble des cavités pyélocalicielles, un certain temps τ_1 après sa captation. En effet la résorbtion d'eau détermine un ralentissement du flux et une accumulation du produit radioactif. Ce temps τ_1 est d'environ 1,2 mn.

La durée τ_2 du phénomène sécrétoire est, normalement, très voisine de τ_1, c'est-à-dire 1,2 mn. Pourtant dans les cas pathologiques ce temps peut être considérablement allongé, et même dépasser la durée de l'épreuve (20 mn).

Enfin le produit chemine dans les cavités pyélocalicielles suivant un mode approximativement laminaire et parvient dans l'uretère et à la vessie après un temps θ.

La valeur de θ dépend de la diurèse et du volume des cavités, grandeurs l'une et l'autre mesurables. Elle est normalement d'environ 1,5 mn.

Pathologiquement trois types principaux d'anomalies peuvent être relevées :

a - la diminution d'affinité du rein pour l'Hippuran : la pente du deuxième segment est alors nettement diminuée.
b - le ralentissement de la sécrétion : le temps de transit rénal d'une fraction du produit, empruntant la voie sécrétoire, est allongé. Ceci se traduit sur les courbes par une décroissance moins marquée dans les premiers instants du segment III. Plus la part de nephrons anormaux est importante, plus cette décroissance est faible. Le sommet peut apparaître étalé ou retardé. Lorsque ce phénomène est isolé, la radioactivité du parenchyme rénal en fin d'épreuve est la même que chez un sujet normal.

c - La stase parenchymateuse : il s'agit en fait du cas limite du précédent : la sécrétion est totalement inexistante, ou au moins insensible, pendant la durée de l'épreuve pour une part plus ou moins importante des nephrons. Lorsqu'elle affecte la totalité des nephrons la stase se caractérise par une courbe constamment ascendante, alors que la sécrétion du rein est abondante et qu'il n'y a pas de dilatation pyélocalicielle.
Lorsqu'elle affecte une partie seulement des nephrons, elle se traduit, en fin d'épreuve, par une radioactivité parenchymateuse trop élevée. Il est important de noter qu'un nephrogramme n'est correctement interprétable que si le temps θ, calculé d'après les données expérimentales (débit d'urine, volume des cavités pyélocalicielles d'après l'urographie) est suffisamment court. De même lorsque la diurèse est insuffisante le débit trop lent masque le phénomène de ralentissement de sécrétion ainsi que la stase parenchymateuse.

VI.2 - Formalisation du modèle

On a vu que la courbe cardiaque suggérait la représentation des espaces vasculaires et intercellulaires par deux compartiments entre lesquels le produit radioactif circule continuellement, passant de l'un à l'autre en quantité proportionnelle à la concentration du compartiment d'origine.

Le produit est injecté instantanément dans le compartiment vasculaire d'où il est ensuite éliminé vers le rein. Les taux de transferts correspondants ont été calculés comme moyenne des valeurs obtenues à partir de courbes expérimentales normales. Ces courbes étaient décomposées en deux exponentielles, $A_1 e^{-\mu t}$ et $B_1 e^{-\nu t}$, à l'aide d'un programme de régression linéaire et les coefficients obtenus par les formules :

k_{21} (taux de transfert du compartiment interstitiel vers le vasculaire) $= \nu A_1 - \mu B_1$

k_e (coefficient d'élimination rénale) $= \dfrac{\mu\nu}{k_{21}}$

k_{12} (taux de transfert du compartiment vasculaire vers l'interstitiel) $= \mu A_1 - k_e$

Les valeurs numériques retenues ont été les suivantes :

$k_{12} \# 0.13\ mn^{-1}$ $\quad k_{21} \# 0.26\ mn^{-1}$ $\quad k_e = 0.08\ mn^{-1}$ à $0.1\ mn^{-1}$

Seule la valeur de k_e est vraiment significative, les deux autres appartenant à une distribution fort large.

Trois voies s'offrent ensuite au produit : l'une sur laquelle son cheminement a une durée $\tau_1 \neq 1,2$ mn. (voie normale). Une autre sur laquelle le temps de transit est $\tau_2 > \tau_1$ (voie pathologique de retard de sécrétion). La troisième enfin, qui correspond à la stase parenchymateuse et où le retard est infini. La modification du paramètre τ_2, ainsi que des proportions de produit empruntant une voie ou l'autre, permettront de simuler différents types d'anomalies. Les deux premières voies se rassemblent ensuite et le produit qui en est issu subit un nouveau retard θ, traduisant l'accumulation pyélocalicielle.

Enfin le réservoir vésical reçoit la totalité du produit sortant du rein

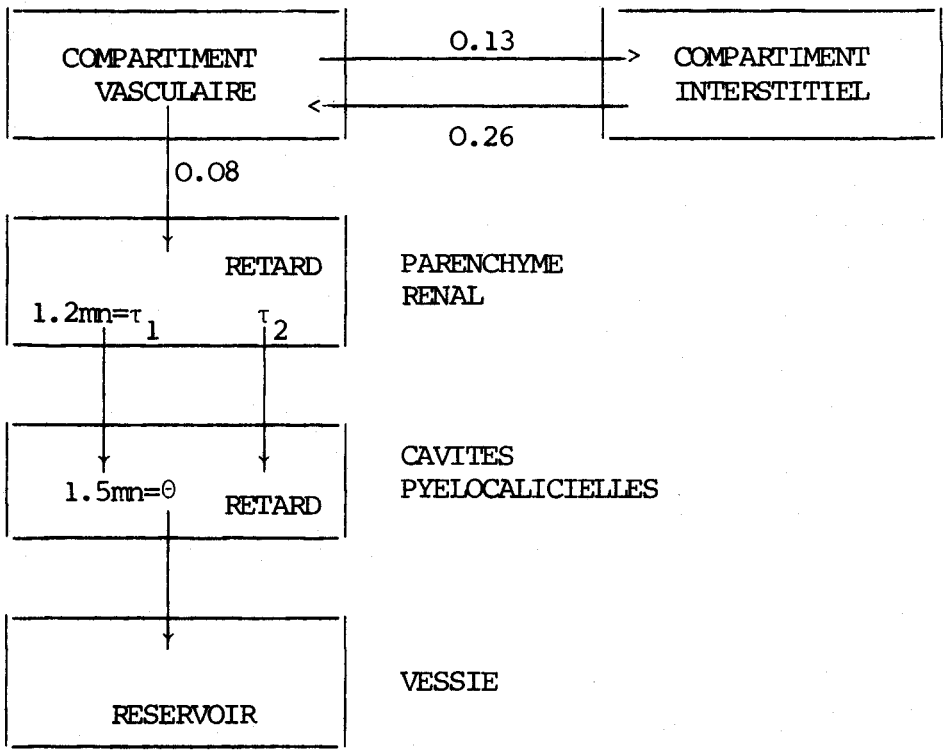


FIGURE III - Schéma du modèle

La modification des paramètres k_e, τ_2, α, β, θ permet la simulation de différents cas pathologiques. Si $\alpha + \beta < 1$ il y a stase parenchymateuse. Dans le cas normal : $\alpha=1$, $\beta=0$, $\theta=1.5$ mn, $k_e = 0.08\ mn^{-1}$.

VI.3 - Programmation SIPHALGOL et simulation

Les considérations précédentes suffisent pour programmer le modèle en SIPHALGOL. Les blocs seront numérotés :

Bloc 1 : Compartiment vasculaire
Bloc 2 : Compartiment interstitiel
Bloc 3 : Parenchyme rénal
Bloc 4 : Cavités pyélocalicielles
Bloc 5 : Vessie

Nous allons d'abord écrire un programme destiné à simuler le cas normal. Nous verrons ensuite comment rédiger un programme plus général, simulant d'autres cas.

On aura donc :

```
NBBLOC(5); PAS(0.1);
COMPARTIMENT(1); ALIMENTE(2); PARAMETRE(0.13);
AINSIQUE(3); PARAMETRE(0.08);
COMPARTIMENT(2); ALIMENTE(1); PARAMETRE(0.26);
RETARD(3); ALIMENTE(4); PARAMETRE(1.2);
RETARD(4); ALIMENTE(5); PARAMETRE(1.5);
RESERVOIR(5); SANSISSUE;
EXPERIENCE:DEPARTA(0); STOPA(20);
BLOC(1); AUTEMPS(0);  INJECTER(100); FINQUAND;
TRACER(CONTENU(3,1)+CONTENU(4,1)+0.18*(CONTENU(1,1)+CONTENU(2,1)));
ENFONCTIONDE(T);
EXECUTION;
```

La valeur 0.18 des proportions de radioactivité vasculaire et interstielle intervenant dans le nephrogramme a été déterminée empiriquement afin d'obtenir des courbes aussi vraisemblables que possible. Elle peut varier notablement d'un sujet à l'autre.

La présence dans ce programme de deux blocs retards (3 et 4) consécutifs se justifie par la possibilité ainsi offerte de distinguer, le cas échéant, les parts parenchymateuse et pyélocalicielle du nephrogramme. En se privant de cette possibilité on aurait pu les concentrer en un seul bloc retard dont le paramètre serait 2.7.

On pourra également rédiger des programmes rendant compte des cas pathologiques. Ainsi la simulation d'un allongement de 5 mn du temps sécrétoire, allongement

affectant successivement 20 puis 40 % du produit capté, serait programmée :

```
NBBLOC(5); PAS(0.1);
COMPARTIMENT(1); ALIMENTE(2); PARAMETRE(0.13);
AINSIQUE(3); PARAMETRE(0.08);
COMPARTIMENT(2); ALIMENTE(1); PARAMETRE(0.26);
RETARD(3); ALIMENTE(4); PARAMETRE(6.2); ET(0.2); PUIS(0.4);
ET(1.2);
RETARD(4); ALIMENTE(5); PARAMETRE(1.5);
RESERVOIR(5); SANSISSUE;
EXPERIENCE:identique à la partie EXPERIENCE du programme précédent.
```

La modification des paramètres des blocs 3 et 4 permettra de simuler d'autres cas (retards, stase, hydronéphrose, et combinaisons diverses des précédents). Ce type de programme peut toutefois sembler un peu grossier puisque la valeur des retards est, au plus, dédoublée et de toute façon fixe dans le temps alors que dans la réalité chaque nephron correspond à un temps de transit différent. Ces valeurs sont plus ou moins groupées suivant l'homogénéïté des nephrons et se répartissent autour d'un ou plusieurs pôles. C'est ce qui explique l'aspect souvent arrondi du sommet des Nephrogrammes réels comparés aux courbes obtenues par la simulation qui, elles, présentent des points anguleux marqués.

Un effet plus réaliste pourra être obtenu en multipliant les valeurs de retards, en leur affectant des proportions de produit reflétant par exemple une distribution approximativement normale, ou encore en faisant varier aléatoirement la valeur des retards au cours de l'expérience.

Le programme pourrait ainsi devenir, par exemple :

```
SIMULATION(TOUTES);
NBBLOC(5); PAS(0.1);
COMPARTIMENT(1); ALIMENTE(2); PARAMETRE(0.13);
AINSIQUE(3); PARAMETRE(0.08);
COMPARTIMENT(2); ALIMENTE(1); PARAMETRE(0.26);
RETARD(3); ALIMENTE(4);
PARAMETRE(7); ET(0.2); PUIS(0.4); PUIS(0);
ET(2); ET(0.16); PUIS(0.12); PUIS(0.18); PUIS(0.14);
ET(1.5); ET(0.36); PUIS(0.42); PUIS(0.63); PUIS(0.49);
ET(1); SIMULATION(3); ET(0.09); PUIS(0.07);
SIMULATION(TOUTES);
RETARD(4); ALIMENTE(5);
PARAMETRE(1.8); ET(0.1); ET(1.6); ET(0.8); ET(1.4); ET(0.1);
RESERVOIR(5); SANSISSUE;
```

```
EXPERIENCE: DEPARTA(0); STOPA(20);
BLOC(1); AUTEMPS(0); INJECTER(100); FINQUAND;
BLOC(3); VERS(4);
PARAMETRE(UNIF(6,7)); ET(CONSTANT);
ET(UNIF(1.5,2)); ET(CONSTANT);
ET(UNIF(1,1.5)); ET(CONSTANT);
ET(UNIF(0.5,1)); SIMULATION(3); A(4); ET(CONSTANT);
SIMULATION(TOUTES);
BLOC(4); VERS(5);
ET(UNIF(1.4,1.6)); ET(CONSTANT);
ET(UNIF(1.2,1.4)); ET(CONSTANT);
TRACER(CONTENU(3,1)+CONTENU(4,1)+0.18*(CONTENU(1,1)+CONTENU(2,1)));ENFONCTIONDE(T);
EXECUTION;
```

Ce programme décrit quatre variantes du modèle qui font l'objet de quatre simulations, numérotées de 1 à 4. Le retard 3 est à l'origine de quatre voies vers le bloc 4. Les retards correspondants varient aléatoirement au cours de l'expérience respectivement entre 6 et 7 mn, entre 1.5 et 2 mn, entre 1 et 1.5 mn et entre 0.5 et 1 mn.

La proportion affectée à la première voie n'est non nulle que pour les première et deuxième simulations. On obtient ainsi un retard de transit portant sur 20 puis 40 % du produit capté.

Les proportions affectées aux trois dernières voies sont toujours choisies de telle sorte que le produit se répartisse entre elles selon la distribution : 20%, 70 %, 10 %. Enfin, dans les deux dernières variantes, une stase portant sur 10 puis 30 % du produit capté est simulée. Dans tous les cas le produit transitant par le quatrième bloc subit un retard variable au cours de l'expérience et qui est pour 80 % compris entre 1.4 et 1.6 mn, pour 10 % entre 1.6 et 1.8 et pour les derniers 10 % entre 1.2 et 1.4 mn.

Les courbes obtenues auront un aspect encore plus réaliste si on ajoute aux expressions considérées un faible bruit de fond, en écrivant par exemple :

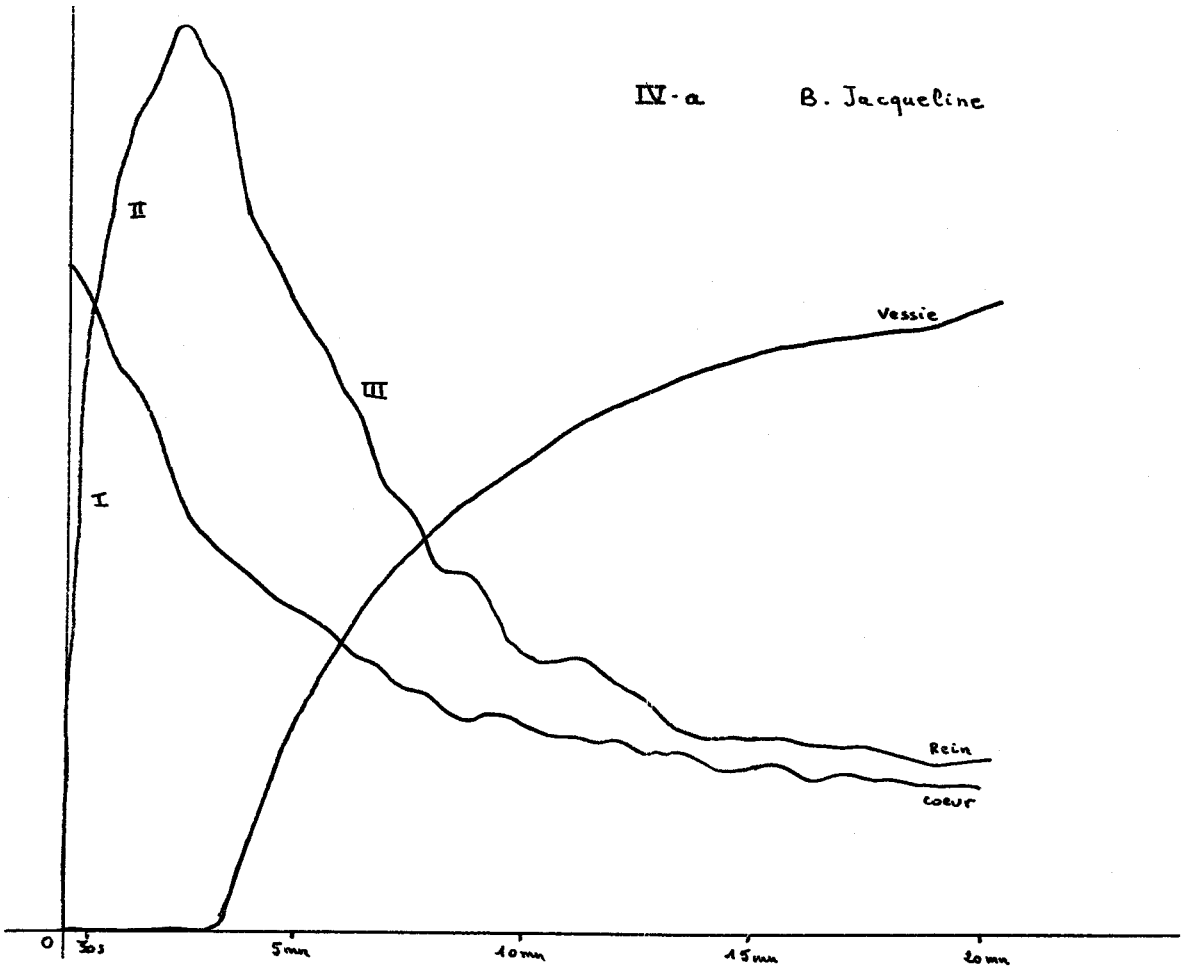
```
TRACER(CONTENU(3,1)+CONTENU(4,1)+0.18 (CONTENU(1,1)+CONTENU(2,1))+UNIF(-0.3,+0.3));
ENFONCTIONDE(T);
```

L'ensemble de figure suivant fait apparaître :

- Des tracés réels expérimentaux avec le diagnostic qui leur a été associé.
- Les tracés théoriques correspondant à la simulation de ces cas. Il faut noter que dans ces simulations l'injection a été supposée instantanée. C'est-à-dire,

compte-tenu du caractère discret des calculs, étendu sur la durée d'un pas d'intégration. Ceci explique la présence sur les courbes obtenues d'un segment I rapidement ascendant et qui tendrait même vers une verticale si le pas devenait infiniment petit. Ce segment ne peut être comparé avec le segment I d'invasion vasculaire des courbes expérimentales. Il faut en fait considérer que l'origine commune des courbes se situe à la naissance du segment II. On pourrait envisager de rendre compte dans la simulation des phénomènes intervenant pendant les premières secondes suivant l'injection, mais ceux-ci sont assez mal connus et le premier segment est en fait de peu d'intérêt.

IV-a B. Jacqueline



IV-b

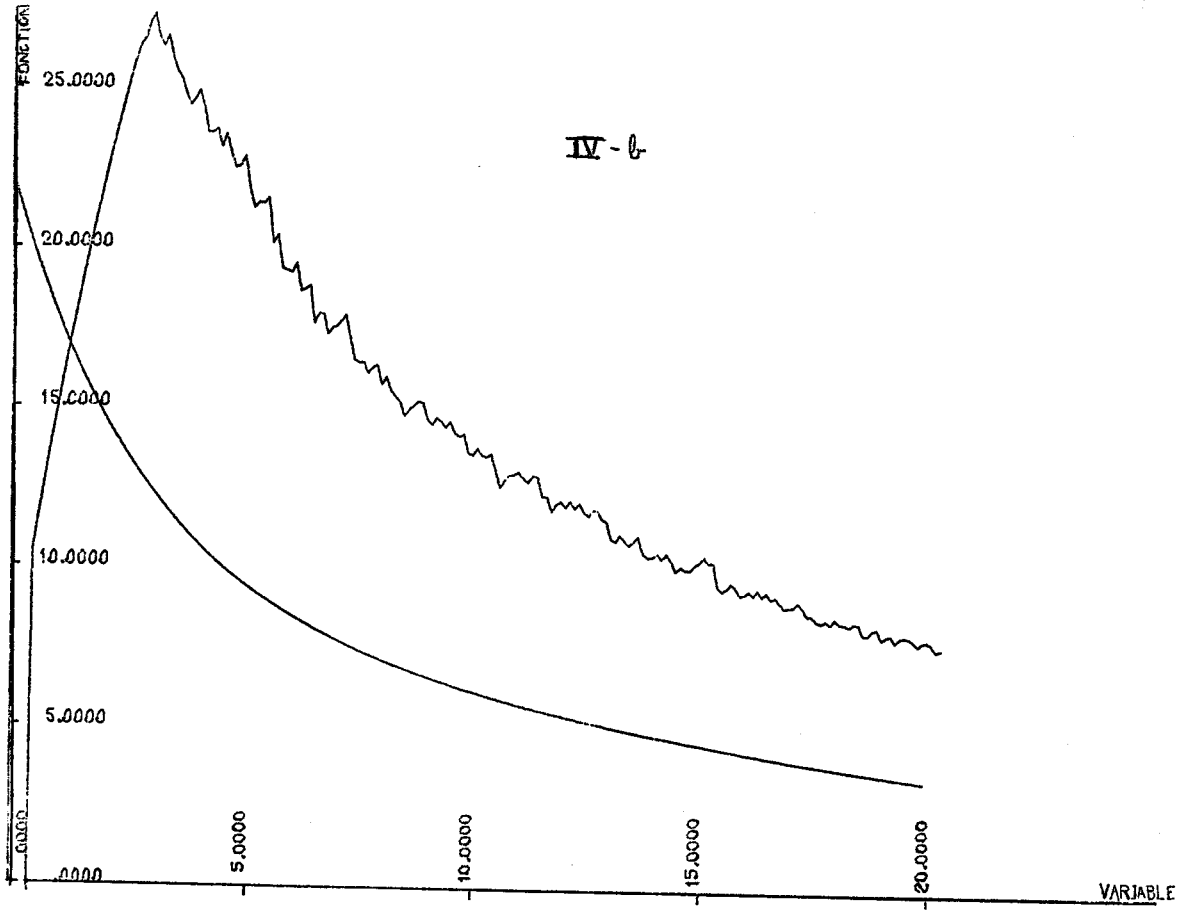


FIGURE IV : Nephrogramme normal

4-a : Sur les Nephrogrammes réels on enregistre simultanément les courbes rénales droite et gauche, la courbe cardiaque et la courbe vésicale. Dans les figures présentées une seule courbe rénale a été reproduite.

Ici le néphrogramme est normal. Le sommet apparaît en place normale aux environs de la 3ème minute.

La décroissance à partir du sommet est importante.

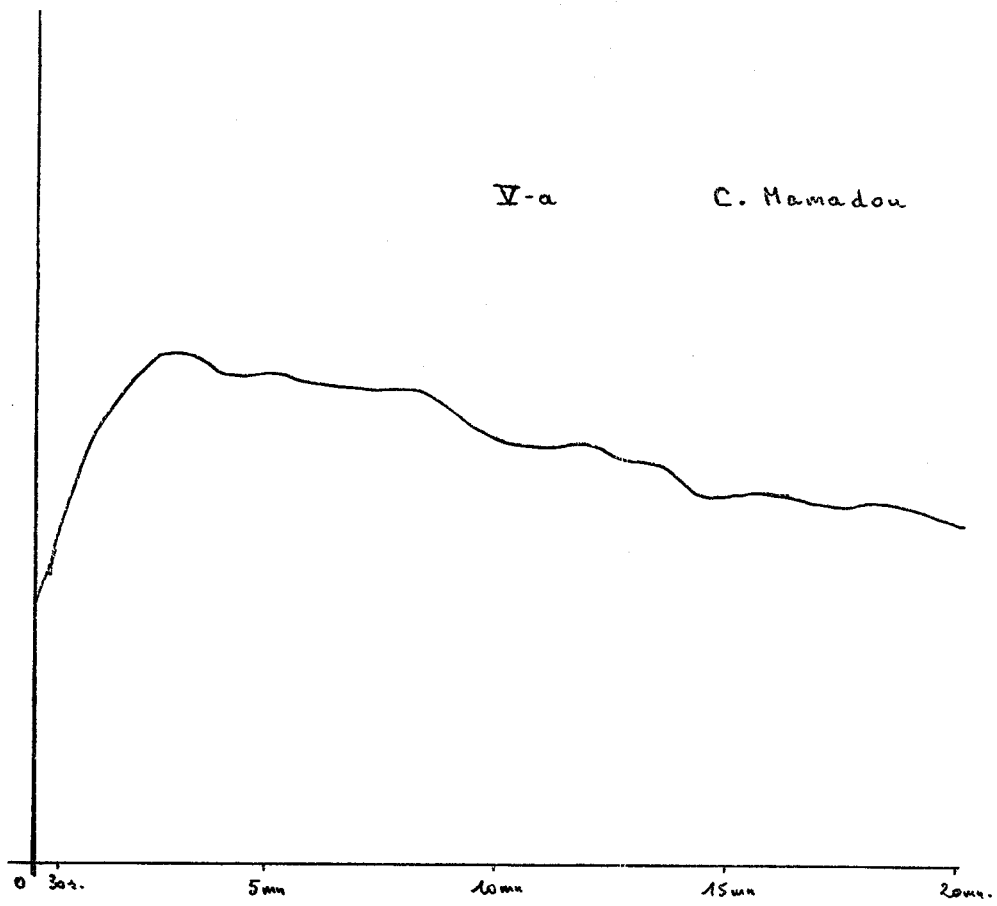
4-b : Les courbes rénale et vasculaire tracées ici correspondent à un modèle normal où :

$k_e = 0.1\ mn^{-1}$ $k_{12} = 0.13\ mn^{-1}$ $k_{21} = 0.2\ mn^{-1}$

τ_1 compris entre 0.5 et 1 mn pour 10 % du produit
" " 1 et 1.5 mn pour 70 % "
" " 1.5 et 2 mn pour 20 % "

θ compris entre 1.2 et 1.4 mn pour 10 % du produit
" " 1.4 et 1.6 mn pour 80 % "
" " 1.6 et 1.8 mn pour 10 % "

En outre un léger bruit de fond est ajouté à la courbe. Le coefficient représentant les parts vasculaires et interstitielles du nephrogramme a été pris ici égal à 0.1. Dans les figures suivantes la valeur 0.18 a été retenue.


V-a
C. Mamadou
0
30s.
5mn
10mn
15mn
20mn.

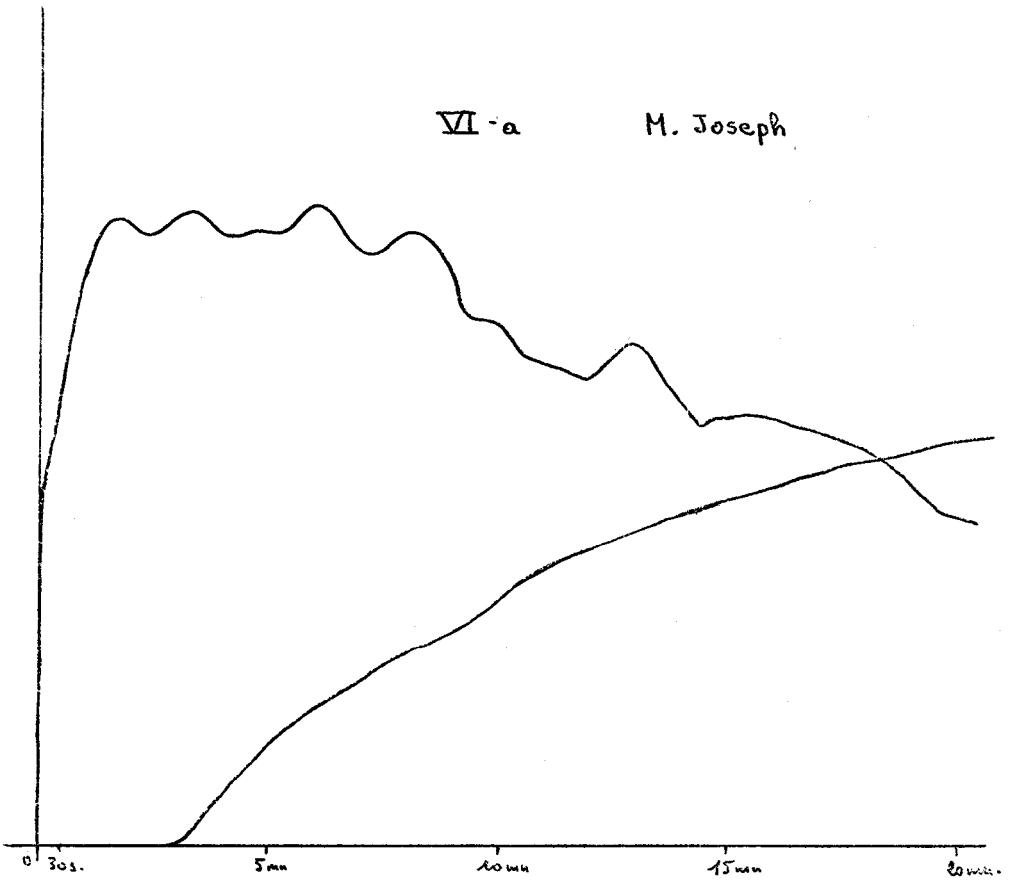

FONCTION
V - b
55.0000
50.0000
45.0000
40.0000
35.0000
30.0000
25.0000
20.0000
15.0000
10.0000
5.0000
5.0000
10.0000
15.0000
20.0000
VARIABLE

FIGURE V : Retard de sécrétion et stase parenchymateuse

5-a ; La légère bosse qui affecte le segment descendant aux environs de la 8ème minute est caractéristique d'un ralentissement du transit affectant une proportion relativement faible de nephrons. En outre la décroissance de ce même segment est ensuite inférieure à la normale, traduisant une stase parenchymateuse, elle aussi assez légère.

5-b : Cette courbe rénale simulée est obtenue en allongeant pour 20 % du produit capté le temps de transit jusqu'à des valeurs uniformément réparties entre 6 et 7 mn. En outre 10 % supplémentaires ne sortent pas du bloc retard parenchymateux, simulant une stase qui s'ajoute au retard de sécrétion.

VI - a M. Joseph



VI - b

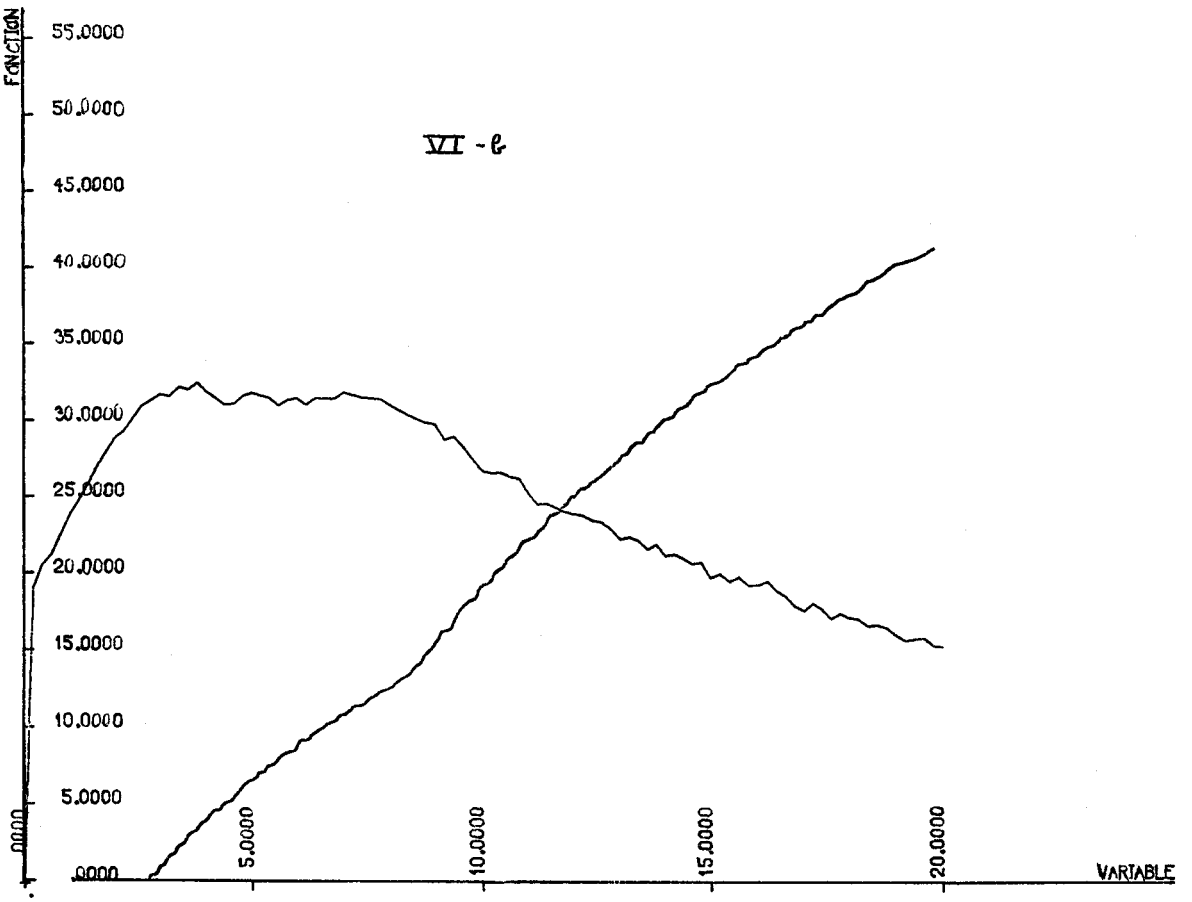
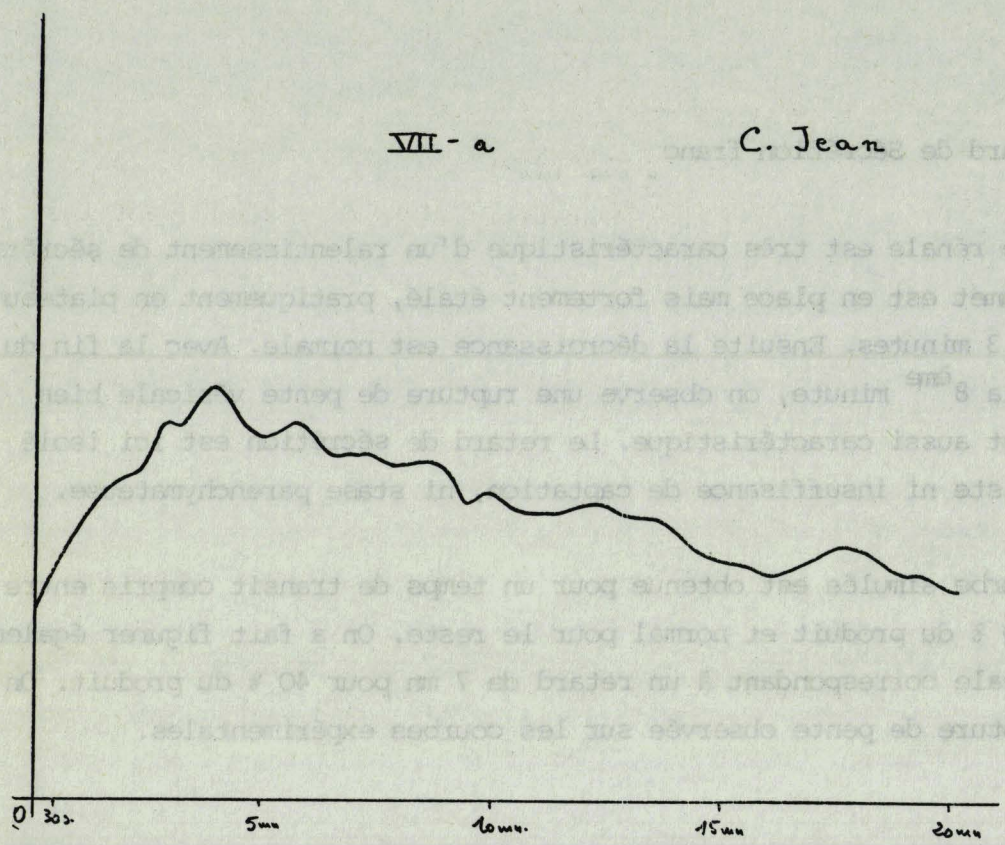


FIGURE VI : Retard de Sécrétion franc

6-a : La courbe rénale est très caractéristique d'un ralentissement de sécrétion. En effet le sommet est en place mais fortement étalé, pratiquement en plateau durant environ 3 minutes. Ensuite la décroissance est normale. Avec la fin du plateau, vers la 8ème minute, on observe une rupture de pente vésicale bien marquée, qui est aussi caractéristique. Le retard de sécrétion est ici isolé puisqu'il n'existe ni insuffisance de captation, ni stase parenchymateuse.

6-b : Cette courbe simulée est obtenue pour un temps de transit compris entre 6 et 7 mn pour 40 % du produit et normal pour le reste. On a fait figurer également la courbe vésicale correspondant à un retard de 7 mn pour 40 % du produit. On retrouve la rupture de pente observée sur les courbes expérimentales.

VII - a C. Jean



VII - b

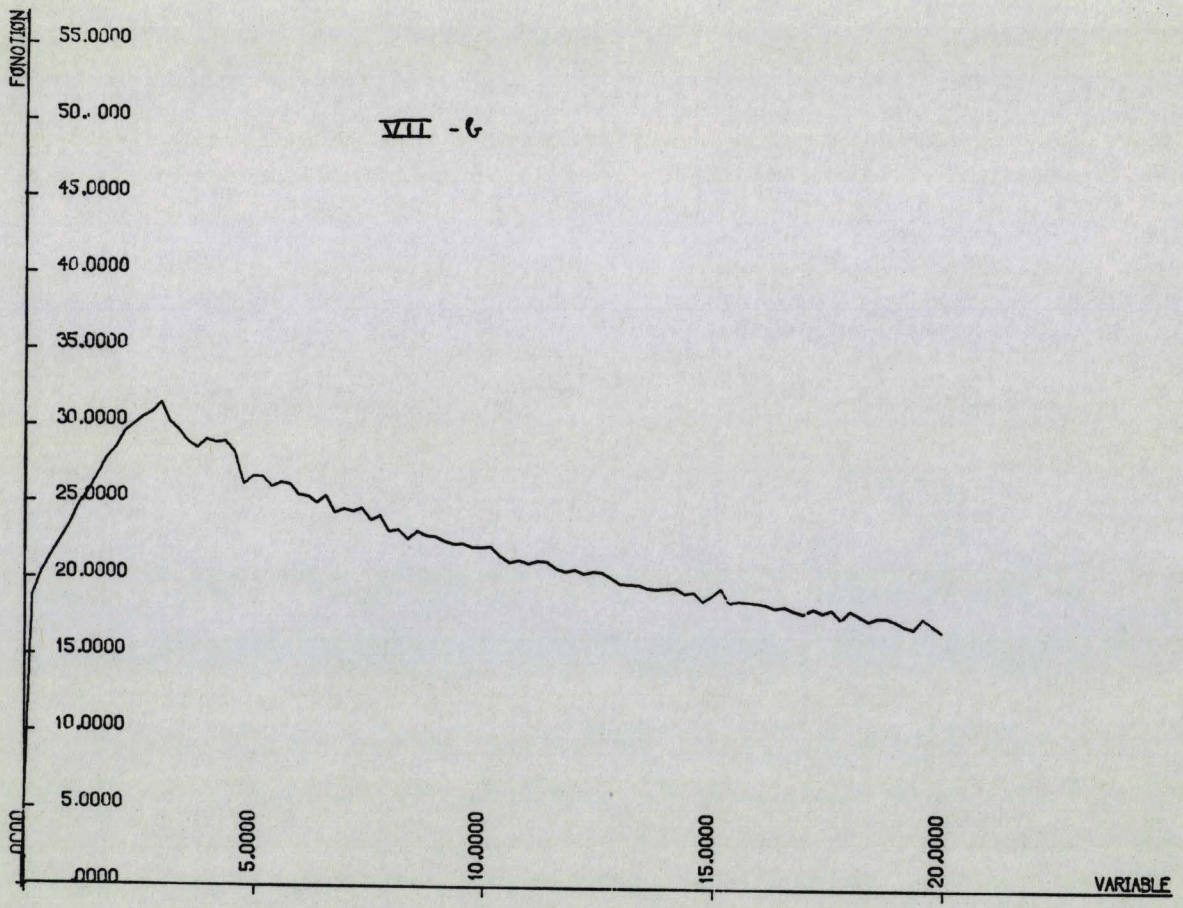
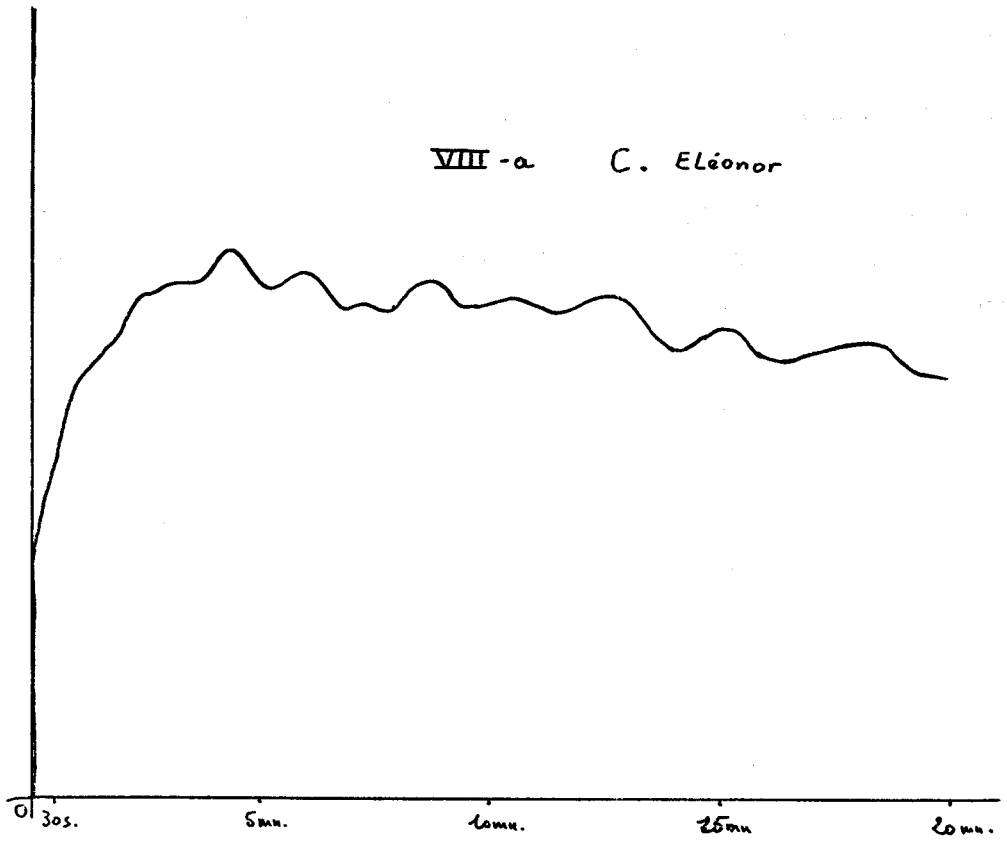


FIGURE VII : Stase parenchymateuse partielle et isolée

7-a : la décroissance à partir du sommet, qui est en place normale, est insuffisante. La pente du deuxième segment est normale. Cet aspect est typique d'une stase parenchymateuse isolée.

7-c : Cette courbe est obtenue par simulation d'une stase affectant 10 % du produit.

VIII -a C. Eléonor



VIII - b

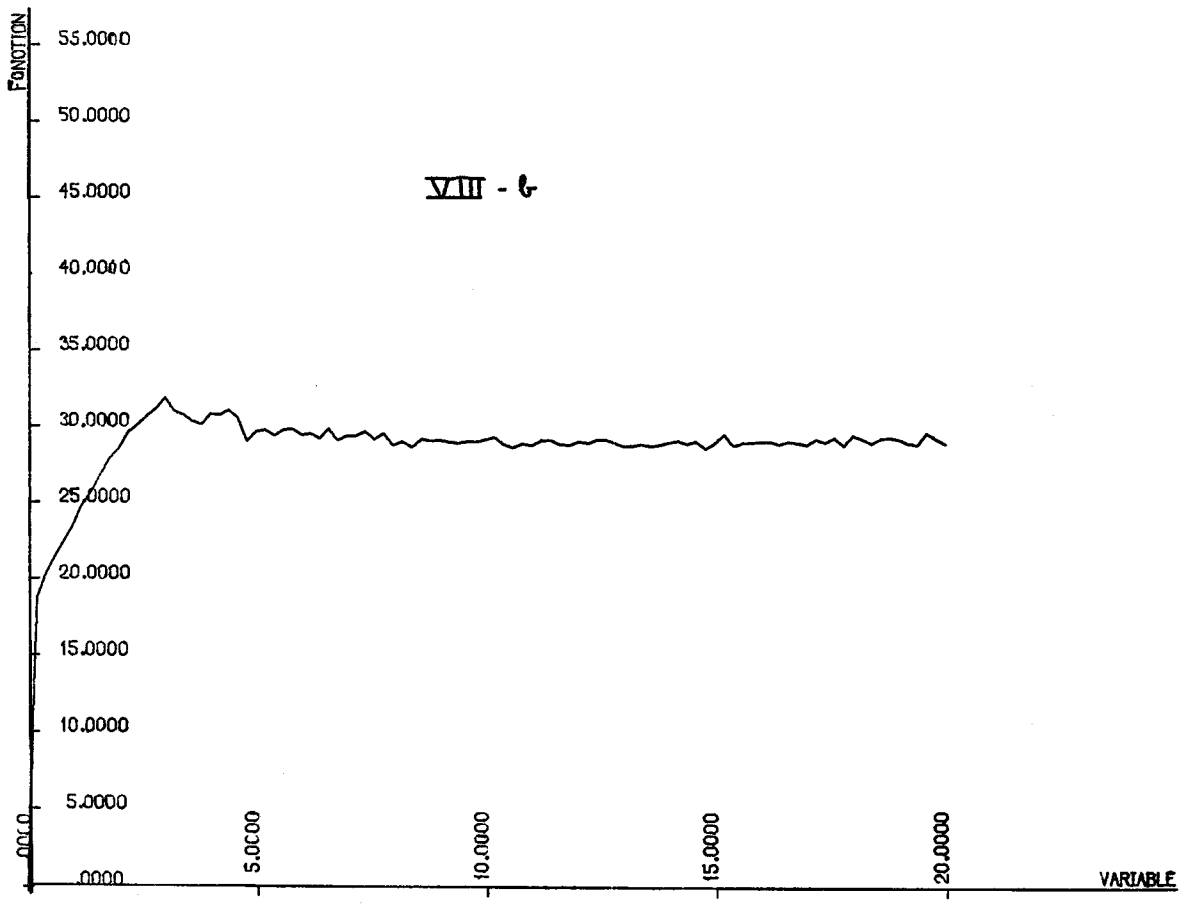
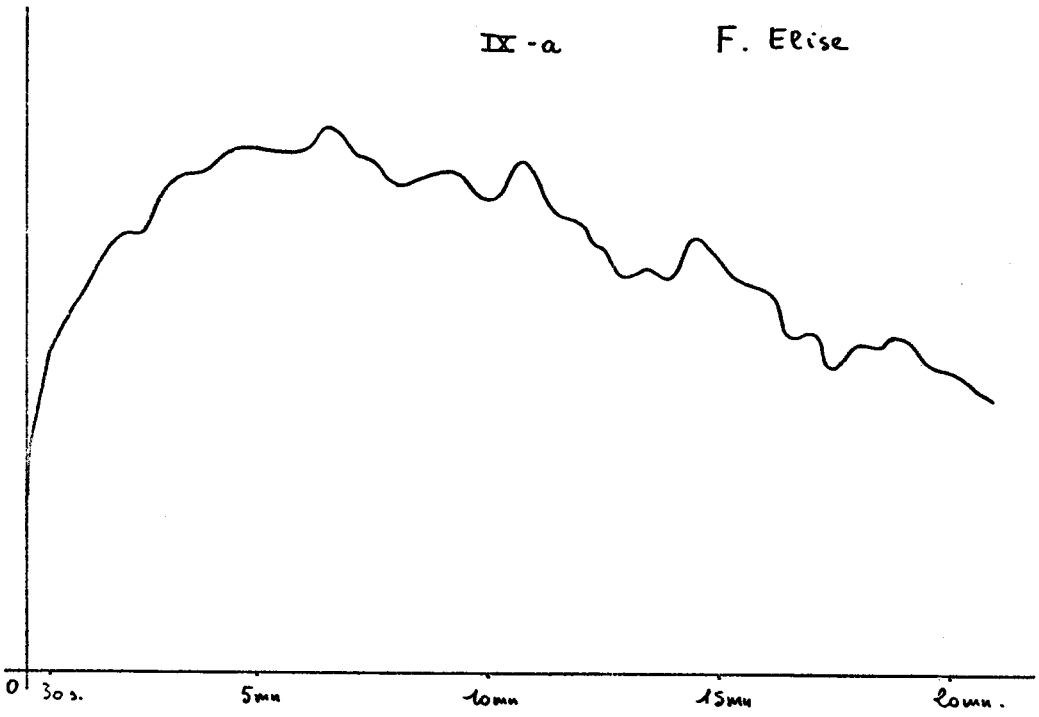


FIGURE VIII : Stase parenchymateuse importante

8-a : On observe une captation normale avec un sommet en place, mais la décroissance est ensuite pratiquement nulle. On peut donc conclure à une stase parenchymateuse très importante.

8-b : La simulation d'une stase affectant 30 % du produit capté produit cette courbe rénale.

IX -a F. Elise



IX -b

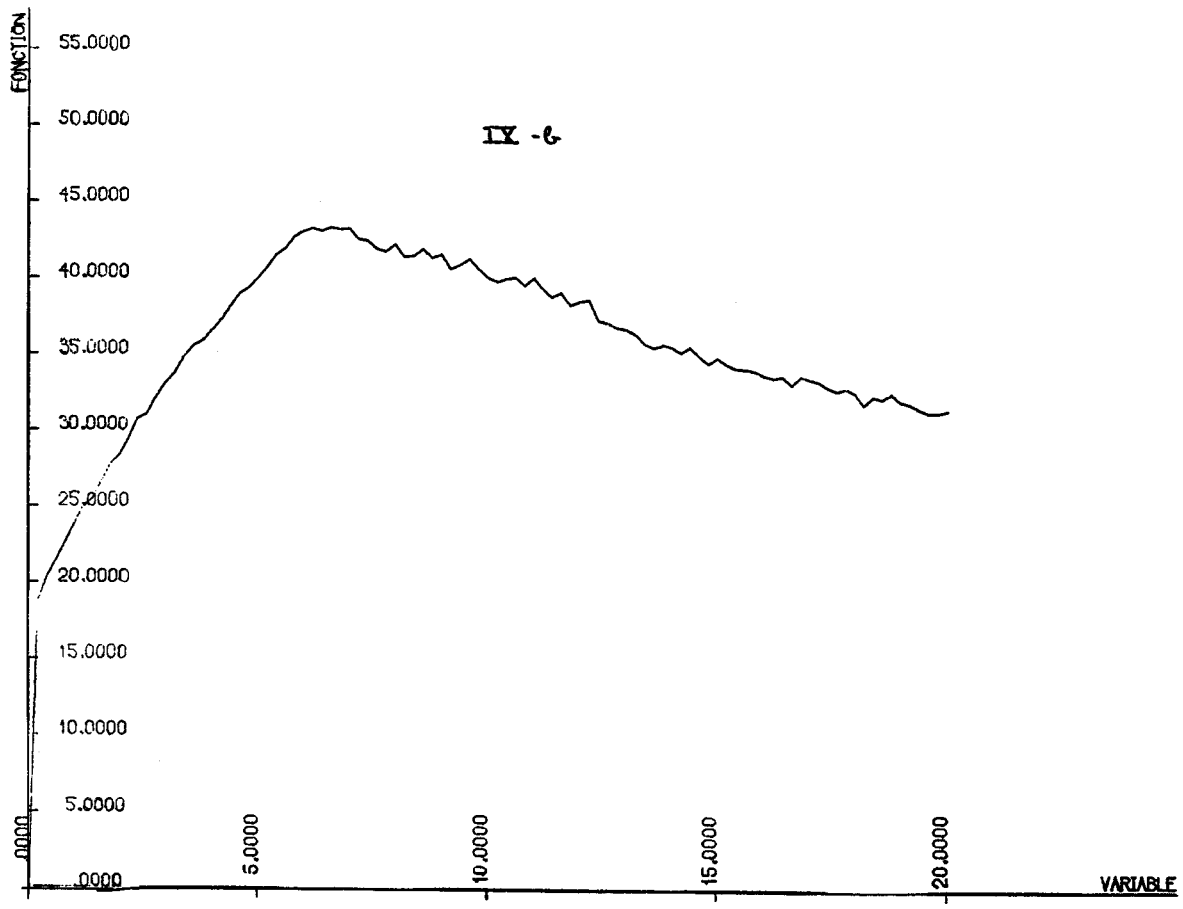
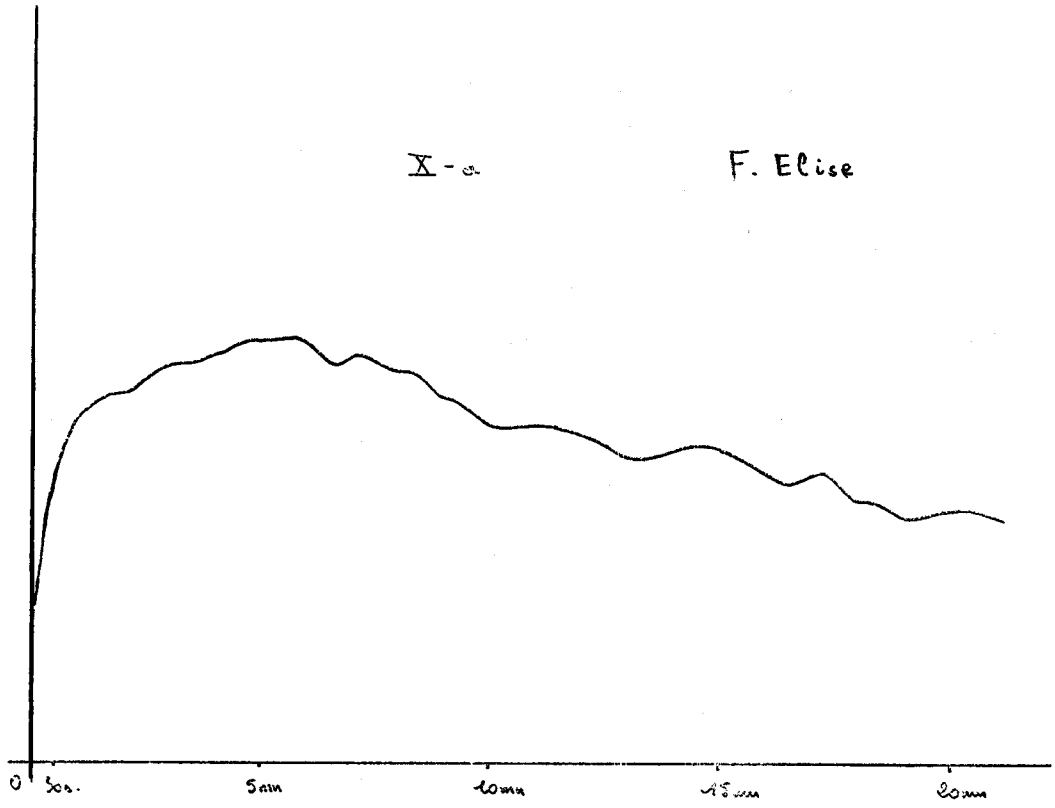


FIGURE IX : Légère hydronephrose, stase importante et retard de sécrétion

9-a : La captation est tout à fait normale mais le sommet est retardé. Le temps de renouvellement pyélocaliciel élevé (4,8 mn) est à l'origine de ce phénomène. A partir du sommet la décroissance est insuffisante (stase) et la concavité du segment descendant est inversé (retard de transit).

9-b : Ici le temps θ de renouvellement pyélocaliciel est compris entre 4 et 6 mn. Une stase affecte 20 % du produit et 20 % supplémentaires subissent en outre un retard de sécrétion allongé de 4 minutes environ.

X - a F. Elise



X - b

FIGURE X : Insuffisance de captation associée à une stase et à un retard de transit.

10-a : La captation de l'Hippuran par ce rein est assez réduite. En effet la pente du deuxième segment est faible. En outre le sommet est étalé et la décroissance du segment III nettement insuffisante. Il existe une stase parenchymateuse associée à une insuffisance de captation et à un ralentissement de sécrétion.

10-b : Cette courbe est réalisée avec une captation diminuée de moitié ($k_e = 0.04$) et une stase portant sur 20 % du produit. Le temps de sécrétion est en outre allongé de 2 mn pour 10 % du produit.

XI - a M. Robert

XI - b

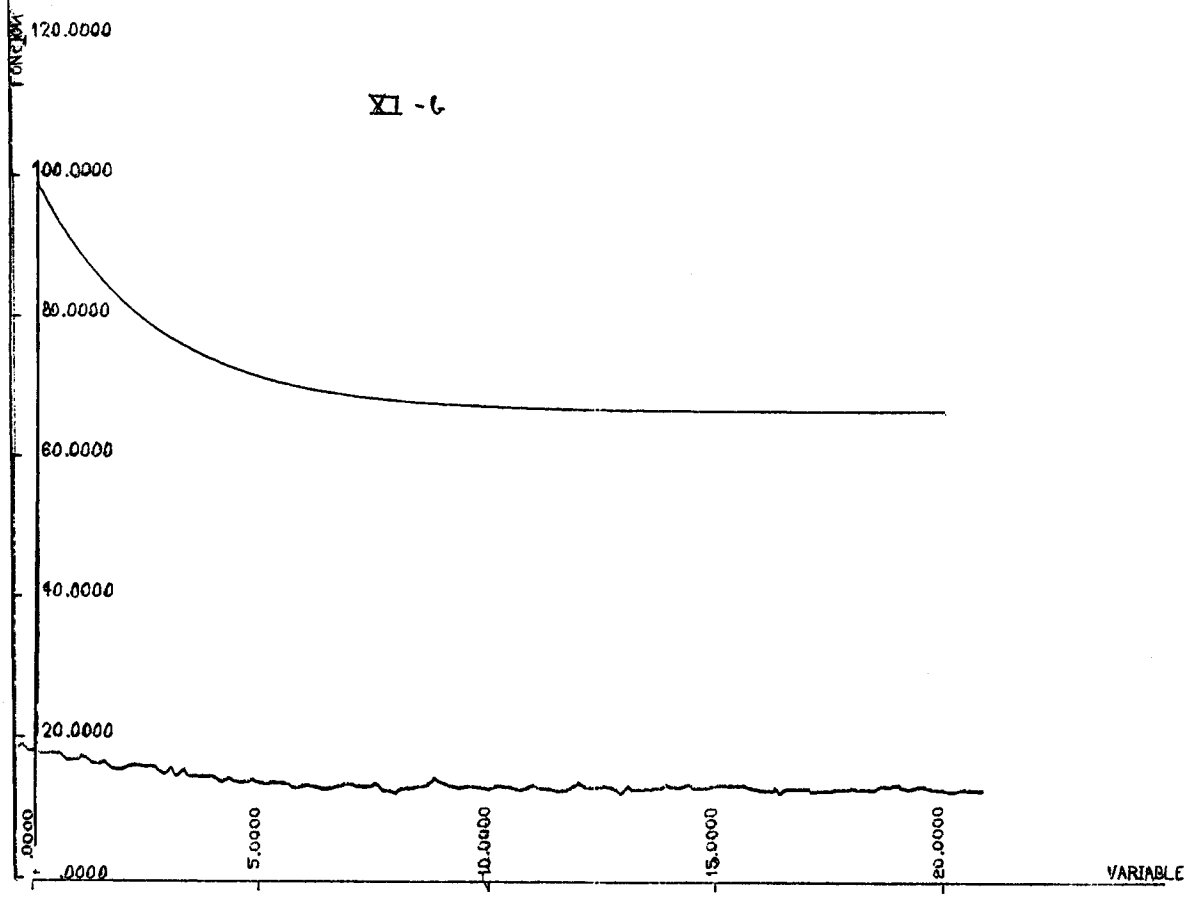


FIGURE XI : Insuffisance totale

11-a : La captation est ici totalement inexistante. La courbe rénale reste plate et la courbe cardiaque tend rapidement vers une horizontale.

11-b : La valeur de k_e est nulle.

RESUME ET CONCLUSION

L'introduction de l'emploi de l'ordinateur a facilité l'application à la physiologie et à la biologie des techniques de simulation. Celles-ci ont été et sont encore à l'origine de nombreux travaux. Les modèles qui y sont présentés correspondent à des processus de nature très diverses. Pourtant un grand nombre d'entre eux peuvent être classés comme continus, dynamiques et déterministes. C'est en particulier le cas des processus d'échange, rencontrés par exemple en physiologie respiratoire, dans l'étude des régulations hormonales, des mouvements liquidiens ou des migrations d'éléments marqués, en pharmacologie etc ...

L'accès du physiologiste ou du médecin au niveau informatique se fait le plus souvent à travers des formulations successives du modèle :

- formulation libre d'abord, résultat de l'étude du système,
- formulation schématique ensuite. Transcription symbolique de la précédente, elle utilise couramment une représentation par blocs, entre lesquels ont lieu les phénomènes d'échange.
- Formulation mathématique, constituée d'un système d'équations décrivant, en fonction des paramètres choisis, la variation de l'état du modèle en cours du temps.
- Formulation informatique enfin, consistant en un programme d'ordinateur, rédigé soit dans un langage universel, soit dans un langage spécialisé.

SIPHALGOL se présente, pour le physiologiste, comme un "langage" formant des "phrases" assez semblables aux phrases du langage naturel et rapprochant étroitement la formulation informatique et la formulation schématique.

Pour l'informaticien, SIPHALGOL est un ensemble de procédures ALGOL 60, utilisables immédiatement avec un compilateur de ce langage, et dont l'appel permet de décrire le modèle et l'expérience puis d'exécuter la simulation. L'étape de formulation mathématique est ainsi réduite au minimum.

Dans son état actuel, SIPHALGOL permet donc, par sa simplicité d'utilisation, un abord plus commode et plus rapide du niveau informatique par le physiologiste.

Certains points sont cependant susceptibles d'amélioration :

- L'utilisation de SIPHALGOL n'est pratiquement pas conversationnelle. Les instructions de description de variantes ne permettent de demander plusieurs simulations

que si les paramètres de chaque expérience ont été prévus à l'avance. Les modifications opérées au vu des résultats des expériences précédentes nécessitent une réécriture du programme et une nouvelle compilation.

- Les procédures sont purement internes et chaque exécution d'un programme nécessite leur compilation préalable, augmentant ainsi assez notablement le temps et le coût de la simulation.
- Les procédures d'ajustement des paramètres, bien que réalisables, ne semblent pas très performantes en raison des temps élevés qu'elles entraînent.
- La description de certains aspects est parfois un peu délicate ou lourde et pourrait vraisemblablement être facilitée par l'adjonction de nouvelles notions à celles existant déjà.

C'est certainement dans ces directions que pourraient être orientées des tentatives d'améliorations, en particulier par l'utilisation d'autres langages qu'ALGOL 60, tels par exemple qu'ALGOL 68.

SIPHALGOL devrait donc être un instrument d'usage facile et répété pour toutes les tâches où la simulation des processus physiologiques peut se révéler utile, que ce soit comme une aide à la recherche ou, ce qui n'est pas à négliger, à l'enseignement.

A N N E X E A

Les règles d'utilisation de SIPHALGOL

LES REGLES D'UTILISATION DE SIPHALGOL

Le programme contient deux parties distinctes, description du MODELE et de l'EXPERIENCE, qui doivent être données dans cet ordre. Dans tout le programme le choix des unités de temps, de volume etc ... est à la charge de l'utilisateur. Il lui appartient de veiller à ce qu'elles soient cohérentes.

lère partie : Description du MODELE

On doit y trouver :

* le nombre de produits en présence :
  NBPROD(p);

* le nombre total de blocs constituant le modèle :
  NBBLOC(n);

* le pas qui doit être utilisé pour l'intégration des équations différentielles :
  PAS(h);

* la description de la structure du modèle. Elle est constituée d'une suite de descriptions de liaisons.
  Chaque liaison est identifiée par : - le produit concerné
  - le bloc origine
  - le bloc extrémité

  Elle est précisée par : - le type du bloc origine
  - le ou les paramètres attachés à cette liaison

  le produit doit être spécifié en premier lieu par l'instruction :
  PRODUIT(p);
  Si un seul produit est en jeu dans le modèle cette instruction n'est pas nécessaire.
  Le bloc origine et son type sont ensuite donnés par :
  type(n);
  où "type" est un mot à choisir dans le catalogue des types de blocs (voir Annexe B).
  Le bloc extrémité est donné par :
  ALIMENTE(b);

Les paramètres sont fournis par autant d'instructions
PARAMETRE(k) ;
qu'il est nécessaire. Les paramètres doivent être fournis dans l'ordre indiqué dans le catalogue des types de blocs. Leur expression doit être une constante. Si certains paramètres doivent prendre des valeurs variables au cours de l'expérience on l'indiquera en remplaçant leur valeur par VARIABLE. On aura alors la forme :
PARAMETRE(VARIABLE) ;
Dans tous les cas le mot PARAMETRE peut être remplacé par le mot ET.
Dans le cas où plusieurs paramètres d'une liaison sont égaux ou sont variables on peut utiliser la forme :
ILYA(n) ; PARAMETRE(...) ;
où n est le nombre de paramètres égaux ou variables.
Le premier paramètre des blocs RETARD ne peut pas être spécifié VARIABLE. Si toutefois on désire qu'il ne soit pas constant on indiquera comme argument de PARAMETRE une valeur maximale qui ne pourra en aucun cas être dépassée par le retard. La valeur effective, et variable, du retard sera déterminée dans la partie EXPERIENCE comme s'il avait été spécifié VARIABLE.
Une possibilité supplémentaire est offerte pour les blocs de type JOKER. Une des liaisons issues de ces blocs, et une seule par bloc, peut être définie comme étant une liaison de vidange. C'est-à-dire que, les débits de sortie par les autres liaisons issues de ce bloc ayant été calculés, tout le reste du produit contenu dans le bloc est éliminé vers le bloc extrémité de la liaison. Il suffit de remplacer le mot PARAMETRE par le mot VIDANGE :
par exemple : JOKER(1); ALIMENTE(2); VIDANGE;
Il faut noter d'ailleurs que le contenu d'un bloc origine d'une liaison de vidange devrait en toute rigueur être constamment nul. En pratique il est égal à la quantité de produit entrée au pas précédent. Cette quantité dépend bien évidemment de la longueur du pas choisie et tend vers zéro avec elle. Il est donc déconseillé de faire mention dans une expression (voir paramètres variables) de ce contenu qui serait sans signification réelle.

Si plusieurs liaisons concernant le même produit ont le même bloc origine, alors :

a) ce bloc doit avoir le même type pour ces liaisons.

b) Leurs descriptions peuvent être groupées, le bloc origine et son type n'étant indiqué qu'une seule fois en tête. Le mot alimente peut alors être

remplacé par le mot AINSIQUE.
Les blocs de type RESERVOIR ne sont l'origine d'aucune liaison. Ils devront néanmoins être décrits. Ils le seront de la façon suivante :
RESERVOIR(b) ; SANSISSUE ;
Le terme SANSISSUE est facultatif et peut être omis sans inconvénient.

Enfin, si plusieurs liaisons concernent le même produit, leurs descriptions peuvent être également regroupées et le produit spécifié une seule fois en tête.

Cette dernière remarque est d'ailleurs valable en tout point du programme. L'indication fournie par une instruction PRODUIT ou BLOC reste valable tant qu'une autre n'a pas été prise en compte. Les liaisons peuvent être décrites dans un ordre quelconque, indépendamment de la numérotation des produits et des blocs.

* Si des valeurs anciennes doivent être appelées dans la seconde partie les instructions correspondantes de mise en mémoire doivent apparaître dans la première partie. Elles sont de la forme :
GARDER(quantité) ; PENDANT(t) ;
La "quantité" à stocker peut être :
CONTENU(b,p) : quantité de produit p contenu dans le bloc b au temps T
ENTREE(b,p) : débit moyen du produit p entrant dans le bloc b entre les temps T-H et T
SORTIE(b,p) : débit moyen du produit p sortant du bloc b entre les temps T-H et T
APPORT(b_1,b_2,p) : débit moyen du produit p transitant du bloc b_1 au bloc b_2 entre les temps T-H et T

où H est le pas d'intégration.

Même s'il n'y a qu'un seul produit l'argument p doit être précisé et est alors égal à 1. L'instruction PAS doit toujours précéder la première instruction GARDER. L'argument "t" de PENDANT représente la durée du stockage. Lorsque la mise en mémoire est demandée pour deux des quantités CONTENU, ENTREE et SORTIE relatives au même bloc et au même produit, pendant des périodes qui peuvent être différentes, la troisième quantité peut être utilisée comme si elle avait été mémorisée pendant la plus petite des deux périodes.

Si par exemple on écrit :
GARDER(CONTENU(1,1)); PENDANT(5);
GARDER(ENTREE(1,1)); PENDANT(10);

outre les stockages demandés on disposera du stockage du débit de sortie, pendant 5 unités de temps, comme si on avait écrit
GARDER(SORTIE(1,1)); PENDANT(5);

Enfin ces demandes de stockage peuvent être formulées à n'importe quel endroit de la partie MODELE, après qu'ait été donnés le nombre de produits, le nombre de blocs et le pas.

* On peut en outre associer à chaque bloc, un volume. Il s'agit d'un volume total indépendant du produit. Il pourra être utilisé dans des expressions dans la seconde partie. Pour ce faire il suffira d'utiliser le mot CAPACITE après que le numéro du bloc ait été spécifié, par exemple :
COMPARTIMENT(1); CAPACITE(50); ALIMENTE(2); etc.
BLOC(2); CAPACITE(100);
Si le volume associé à un bloc doit être variable au cours de l'expérience, on n'en fera pas mention dans la première partie et l'expression permettant de le calculer sera donnée dans la seconde partie (voir page A-7).

2ème partie : Description de l'expérience

Elle doit être introduite par l'étiquette EXPERIENCE :
Deux sections préliminaires peuvent ensuite trouver place en tête de cette seconde partie. L'une ou l'autre, ou les deux, peuvent être omises.

* Le stockage d'une quantité durant une période t_1 permet de faire référence à sa valeur au temps $T-t_1$, au maximum. Toutefois si T est inférieur à T_o+t_1 (où T_o est la valeur initiale du temps), $T-t_1$ est alors inférieur à T_o et aucune exécution n'a pu permettre de calculer la valeur demandée, pour cette valeur du temps. Aussi dans ce cas les valeurs sont elles supposées a priori nulles. Si on désire leur affecter une valeur non nulle, c'est-à-dire fixer l'état pré-initial du modèle, on pourra le faire par l'emploi des instructions suivantes :
TPETIT;
FAIRE(exquantité(..., TO-T1));
EGALE(f(T1));
FINTPETIT;
où "exquantité" représente une des expressions :
EXCONTENU(b, p, t) égale au contenu du bloc b en produit p au temps t < T
EXENTREE(b, p, t) " au débit d'entrée "
EXSORTIE(b, p, t) " au débit de sortie "

EXAPPORT(b_1, b_2, p, t) égale au débit entre le bloc b_1 et le bloc b_2 en produit p au temps t < T

f(T1) est une fonction quelconque de T1.

L'emploi des identificateurs TO et T1 est obligatoire.

L'effet de ces instructions sera d'affecter, avant toute exécution et pour les temps TO-T1 inférieurs à TO, correspondant au stockage, les valeurs f(T1) à la grandeur stockée.

Si plusieurs grandeurs stockées doivent être ainsi affectées, les instructions FAIRE et EGALE correspondantes seront groupées entre les deux instructions TPETIT et FINTPETIT, qui doivent n'apparaître qu'une seule fois.

* Il faut ensuite, si nécessaire, préciser les conditions initiales. Cette partie du programme commence par l'instruction :
INITIALEMENT;

Le contenu de tous les blocs, en tout produit, est présumé nul au départ de l'expérience. On pourra le forcer à une valeur non nulle de la façon suivante :

D'abord spécifier le produit :
PRODUIT(p);

S'il n'y a qu'un seul produit ceci n'est pas nécessaire.

Spécifier ensuite le bloc :
BLOC(b);

Donner le contenu :
CONTIENT(q);
q est une valeur numérique.

Dans certains cas il faudra également préciser le débit moyen initial (c'est-à-dire en fait entre les temps T_0-H et T_0) du produit dans une liaison. On écrira alors :
PRODUIT(p); BLOC(b_1); DEBITE(q); VERS(b_2)
Ceci est nécessaire en particulier quand la valeur non nulle de ce débit intervient dans l'expression d'un paramètre où d'une quelconque quantité à calculer (voir pages suivantes).

Comme toujours plusieurs instructions relatives au même bloc ou au même produit peuvent être groupées après une seule instruction BLOC ou PRODUIT.

La fin de l'énoncé des conditions initiales est marqué par l'instruction :
FININIT;

* Une instruction doit ensuite fixer la valeur initiale du temps :
DEPARTA(t_o) ;

* Une seconde instruction peut éventuellement fixer la valeur finale du temps :
STOPA(t_2);
Si l'exécution n'a pas été interrompue plus tôt, elle le sera dès que la valeur courante du temps sera supérieure à t_2.

Ensuite les huit types d'instructions suivants, facultatifs, peuvent intervenir dans un ordre quelconque. L'ordre indiqué est cependant recommandé.

* Il est possible de supprimer une liaison du système pour un certain produit.
Il suffira d'indiquer, après spécification du produit :
COUPER(i,j);
i et j étant les numéros des blocs origine et extrémité de la liaison.
Si on désire couper toutes les liaisons issues d'un même bloc (resp. : arrivant à un même bloc) on pourra écrire :
COUPER(i,TOUS); (resp : COUPER(TOUS,j) ;)
Si on désire couper toutes les liaisons arrivant et partant d'un même bloc on écrira :
EXCLURE(i);

Les liaisons coupées pourront être rétablies à l'aide des instructions :
RETABLIR(i,j); RETABLIR(i,TOUS); RETABLIR(TOUS,j);

De même on reprendra en compte les blocs exclus par :
REINCLURE(i) ;

Dans la plupart des cas ces instructions seront utilisées avec les instructions conditionnelles. (Voir page A-9).
Enfin si l'une quelconque de ces instructions de verrouillage ou de déverrouillage doit être appliquée à tous les produits on pourra spécifier auparavant :
PRODUIT(TOUS) ;

Il conviendra alors de préciser à nouveau le produit pour les instructions suivantes.

* L'expérience peut être interrompue par l'instruction :
STOP;
associée à une instruction conditionnelle. (voir page A- 9).

Les quatre types d'instructions qui suivent ont une "expression" en argument. Dans tous les cas il s'agit d'une expression arithmétique dont les termes peuvent être :

- des constantes numériques
- la variable T représentant le temps
- les quantités APPORT(b_1,b_2,p)

ENTREE(b,p)
SORTIE(b,p)
CONTENU(b,p)
EXAPPORT(b_1,b_2,p,t)
EXENTREE(b,p,t)
EXSORTIE(b,p,t)
EXCONTENU(b,p,t) déjà définies

} où l'argument p peut être le mot TOTAL ou TOTALE. On obtient alors la somme des quantités concernées pour tous les produits.

- VOLUME(b) : égale à la quantité donnée, pour ce bloc, en argument de CAPACITE. Si aucune valeur n'a été donnée alors VOLUME(b) = CONTENU(b,TOTAL)
- CONCENTRATION(b,p) = CONTENU(b,p)/VOLUME(b) (si VOLUME(b)=0, CONCENTRATION (b,p)=0)
- les fonctions standard du compilateur (sin, cos, sqrt etc...)
- les quantités UNIF(a,b) et NORMAL(m,s) donnant des nombres pseudo-aléatoires extraits de distributions respectivement uniforme entre a et b, et normale de moyenne m et d'écart type s.

* Si le volume associé à un bloc a une valeur variable celle-ci sera fournie par une expression en argument de l'instruction CAPACITE :
BLOC(b); CAPACITE(q);

* On doit indiquer s'il y a lieu, l'expression des paramètres variables. Le procédé est semblable à celui utilisé dans la première partie pour les paramètres constants. La liaison est repérée par le bloc origine, le bloc extrémité et le produit.

Les blocs origine et extrémité sont donnés par les instructions :
BLOC(b_1); VERS(b_2);

Les mots PARAMETRE ou ET donnent ensuite les paramètres par des expressions au sens défini ci-dessus.
En outre, les paramètres constants ne doivent plus être donnés mais leur existence doit cependant être mentionnée.

L'argument des mots PARAMETRE ou ET correspondants sera alors le mot CONSTANT :
Exemple : BLOC(1); VERS(2); ILYA(2); PARAMETRE (CONSTANT);
ET(0.8*(CONTENU(3,2)-20)); ET(CONSTANT);

* Les phénomènes autres que le simple échange doivent également être décrits. Ils comprennent les injections, les perfusions, les synthèses et les utilisations. Ces deux derniers termes correspondent au cas où la variation, positive ou négative, du contenu d'un bloc n'est pas uniquement due à des échanges avec d'autres blocs.
Le produit et le bloc ayant été spécifiés à l'aide des instructions PRODUIT et BLOC, une des instructions :
INJECTER(q);
PERFUSER(q);
SYNTHETISE(q);
UTILISE(q);
définira la quantité de produit concernée.

L'injection est supposée instantanée et q représente la quantité de produit injectée. Dans les trois autres cas, q est le débit de la perfusion, de la synthèse ou de l'utilisation.
Cet argument q est une expression quelconque. Si le contenu d'un bloc doit être forcé à zéro sans qu'il y ait transfert vers un autre bloc, on pourra utiliser :
VIDER(b);

* Enfin il faut indiquer la forme sous laquelle les résultats de l'expérience doivent être présentés. Les instructions correspondantes pourront varier suivant les possibilités du matériel et les fonctions de sortie du compilateur utilisé. Les instructions qui suivent ont été réalisés dans le programme prototype.

- Impression des résultats intermédiaires :

Des valeurs peuvent être imprimées au cours de l'exécution.
On utilisera les instructions :
AFFICHER(q); INTITULE("ch");
q est ici encore une expression dont la valeur est alors imprimée, repérée par le libellé "ch", ainsi que la valeur courante du temps.

- Tracé de courbe :

On peut également demander le tracé d'une courbe représentant l'évolution d'une

grandeur en fonction d'une autre. On utilisera :

TRACER(q_1);

ENFONCTIONDE(q_2);

q_1 et q_2 sont deux expressions.

La courbe peut être tracée sur table traçante ou sur imprimante. Cette précision est fournie par :

SURIMP;

ou

SURTAB;

Dans le cas du tracé sur imprimante les échelles, axes, cotations etc... sont automatiquement définis. Dans le cas de la table traçante ces indications peuvent être optionnellement fournies par l'utilisateur, à l'aide des instructions :

ECHX(e); : échelle en x

ECHY(e); : " y

ORIGX(o); : origine sur l'axe des x

ORIGY(o); : " " y

QUOTX(i); : intervalle de cotation sur l'axe des x

QUOTY(i); : " " " y

MINIMX(m); : valeur minimum en X

MINIMY(m); : " " Y

- Etat du système :

L'instruction :

ETAT(t);

permet l'impression d'un tableau récapitulatif de l'état du système au temps t.

* Les instructions conditionnelles : on peut contraindre des instructions ou suites d'instructions à n'avoir d'effet que si certaines conditions sont vérifiées. Les instructions qui peuvent être ainsi conditionnées sont :

PARAMETRE(k);

ET(k);

COUPER(i,j);

RETABLIR(i,j);

EXCLURE(i);

REINCLURE(i);

VIDER(b);

TRACER(q);

ENFONCTIONDE(q);

DEPUIS(t);

TOUSLES(ht);

JUSQUA(t);

} voir paragraphe suivant

SYNTHETISE(q);
UTILISE(q);
INJECTER(q);
PERFUSER(q);
AFFICHER(q);
INTITULE("ch");
ETAT(t);
STOP;

Les contraintes sont exprimées à l'aide des instructions :
QUAND(r);
AUTREMENT;
FINQUAND;
r est une relation entre deux expressions arithmétiques, qui peut être ou non réalisée.
Ces instructions peuvent être utilisées de deux façons différentes :

1) QUAND(r); ..... FINQUAND;

les instructions placées entre QUAND et FINQUAND sont prises en compte uniquement si la relation r est vraie. Dans le cas contraire elles sont ignorées.

2) QUAND(r); ..... AUTREMENT; ..... FINQUAND;

les instructions placées entre QUAND et AUTREMENT sont prises en compte chaque fois que r est vraie et celles placées entre AUTREMENT et FINQUAND sont dans ce cas ignorées. Dans le cas contraire ce sont évidemment les premières qui sont sautées et les secondes exécutées.

Il faut noter que les opérations de verrouillage et déverrouillage de liaisons présentent un caractère rémanent. Si une liaison a été coupée dans certaines conditions (par une suite d'instructions :

QUAND(r); ..... COUPER(i,j); ..... FINQUAND;

elle reste coupée, même si les conditions ne sont plus vérifiées, jusqu'à ce qu'une instruction RETABLIR puisse être prise en compte à son sujet.

L'instruction conditionnelle :
QUAND(T=t); ..... FINQUAND;
peut être avantageusement remplacée par :
AUTEMPS(t); ..... FINQUAND;
On peut imbriquer jusqu'à 10 groupes QUAND; .....; FINQUAND;

* Les instructions contrôlées

Il s'agit en fait d'un type particulier d'instructions conditionnelles : la

contrainte est ici exclusivement temporelle : les ordres correspondants ne prendront effet que pour des valeurs bien déterminées du temps. Les instructions susceptibles de subir ces contraintes sont :

VIDER	SYNTHETISE	INTITULE
COUPER	UTILISE	TRACER
EXCLURE	PERFUSER	ENFONCTIONDE
RETABLIR	INJECTER	
REINCLURE	AFFICHER	

Les conditions de validité sont données par :
DEPUIS(t_1); TOUSLES(h_t); JUSQUA(t_2);
précédant l'instruction.
Un ou deux quelconques de ces trois ordres peuvent être omis. Ils peuvent être donnés dans n'importe quel ordre mais s'ils sont tous deux présents, DEPUIS doit précéder TOUSLES.
Quand les trois ordres sont donnés, l'instruction qui les suit n'est effectuée que pour des valeurs du temps commençant à t_1 et progressant jusqu'à t_2 exclus par pas de h_t.

Lorsque JUSQUA est présent, l'instruction visée n'est pas exécutée pour la valeur t_2 du temps.
Si on désire l'exécution pour cette valeur, il faudra écrire :
JUSQUA(t_2); COMPRIS;

Si DEPUIS (resp. JUSQUA) est omis la validité commence (resp. se termine) avec l'expérience. Si enfin TOUSLES est omis, h_t est égal à H, pas d'intégration.

En tous cas $h_t \geq H$

Ces ordres peuvent être utilisés conjointement avec les instructions conditionnelles ordinaires.

* Description de variante :

On peut demander la réalisation de plusieurs simulations successives portant sur des variantes du modèle ou de l'expérience. Les modifications peuvent donc aussi bien porter sur les indications fournies dans la partie MODELE que dans la partie EXPERIENCE.

Deux procédés sont possibles :

1) Il faut que soit précisée pour chaque instruction ou suite d'instructions, quelle est la simulation qui est concernée par ces instructions. Ceci est fait

au moyen de l'instruction :
SIMULATION(i);

Toutes les instructions suivantes, jusqu'à la prochaine instruction SIMULATION(j); (j≠i), ne seront prises en compte que lors de la $i^{ème}$ expérience. Le programme devra donc obligatoirement commencer par une instruction SIMULATION.

Bien entendu si une seule simulation est demandée l'instruction SIMULATION(1) pourra être omise. A l'intérieur de chacune des deux parties du programme on pourra, soit regrouper toutes les instructions relatives à une même simulation, soit les exprimer en plusieurs parties, en spécifiant à chaque fois le numéro de la simulation.

Une instruction ou un groupe d'instructions valables pour toutes les simulations demandées pourra être exprimé une seule fois en le faisant précéder de :
SIMULATION (TOUTES);

Si des instructions doivent être prises en compte pour des expériences dont le numéro est compris entre 2 valeurs s_1 et s_2 (s_1 et s_2 inclus) on pourra utiliser :
SIMULATION(s_1); A(s_2);

2) Une procédure simplifiée est utilisable lorsque les modifications concernent uniquement une ou plusieurs informations apportées par des instructions appartenant à la liste suivante :

PAS	UTILISE	JUSQUA	QUOTY
PARAMETRE	INJECTER	AFFICHER	MINIMX
ET	PERFUSER	ETAT	MINIMY
PENDANT	DEBITE	ECHX	
DEPARTA	TRACER	ECHY	
STOPA	ENFONCTIONDE	ORIGX	
CONTIENT	DEPUIS	ORIGY	
SYNTHETISE	TOUSLES	QUOTX	

Dans ces cas, assez fréquents, il suffira de faire suivre autant de fois que nécessaire l'instruction considérée de :
PUIS(x);
où x représente la nouvelle valeur ou la nouvelle expression de l'argument de l'instruction à modifier. Le nombre d'expériences réalisées sera le nombre maximum de PUIS suivant une instruction. Si une instruction est suivie d'un nombre de PUIS inférieur à ce nombre de simulations l'argument considéré prendra

les valeurs indiquées jusqu'à épuisement de la liste et gardera pour les expériences suivantes la dernière valeur prise.
Toutefois, si l'argument du dernier PUIS est le mot FANTOME, l'instruction considérée ne sera plus exécutée lors des simulations suivantes.

Exemple :
COMPARTIMENT(1); ALIMENTE(2); PARAMETRE(O.1); PUIS(O.11); PUIS(O.12);
PUIS(O.13); ET(15); PUIS(2O); PUIS(FANTOME);
conduira à quatre simulations successives (4 valeurs données pour le premier paramètre). Les valeurs des deux paramètres seront :

Simulation	1er paramètre	2ème paramètre
1	O.1	15
2	O.11	20
3	O.12	O (non fourni)
4	O.13	O (non fourni)

Un paramètre qui était constant lors d'une expérience pourra devenir variable lors de la simulation suivante :
COMPARTIMENT(1); ALIMENTE(2); PARAMETRE(O.1); PUIS(VARIABLE);
et réciproquement.
Les instructions correspondantes de la partie EXPERIENCE devront alors être ainsi rédigées :
BLOC(1); VERS(2); PARAMETRE(CONSTANT); PUIS(CONTENU(1,1)/CONTENU(2,1));

Les deux procédés décrits ci-dessus peuvent être utilisés simultanément. L'exemple suivant montre comment les instructions prennent alors effet :
SIMULATION(TOUTES); COMPARTIMENT(1); ALIMENTE(2);
PARAMETRE(O.1); PUIS(O.11); SIMULATION(2); ET(15);
PUIS(2O); SIMULATION(3); ET(1.5); PUIS(1.9);

Il y aura 4 simulations successives. Dans chacune existera une relation entre les blocs 1 et 2.
Les valeurs prises par les paramètres sont indiquées page suivante.

Si des instructions PUIS interviennent dans un groupe conditionnel ce ne peut être qu'avec la forme complète :
QUAND( ); ..... ; AUTREMENT; ..... ; FINQUAND;

Simulation	1er paramètre	2ème paramètre	3ème paramètre
1	0.1	0 (non donné)	0 (non donné)
2	0.11	15	0 (non donné)
3	0.11	20	1.5
4	0.11	20	1.9

En outre les nombres des instructions PUIS avant AUTREMENT et après doivent être égaux et ces instructions se correspondre deux à deux.

* Pour terminer la partie EXPERIENCE doit être close par l'instruction EXPERIENCE;

A N N E X E B

Catalogue et description des
différents types de blocs

Q : contenu du bloc

E : quantité de produit entrant dans le bloc entre les temps T et T+dT

S : " " sortant " "

P1, P2, P3 : premier, deuxième et troisième paramètres attachés à chaque liaison. Ils doivent être donnés dans cet ordre dans le programme.

TYPE	Nombre de PARAMETRES	FONCTION
RESERVOIR	0	Ce bloc accumule tout le produit qui y parvient sans rien évacuer. $S = 0$
EMBRANCHEMENT	1	Ce bloc répartit entre ses différentes sorties le produit qui lui parvient, dans les proportions indiquées par le paramètre affecté à chaque voie. La somme de ces paramètres doit bien entendu être égale à 1. Un des paramètres peut être omis. Il est alors égal à $1 - \Sigma P$. $S = P1 \times E$ Deux blocs de ce type ne peuvent être consécutifs.
COMPARTIMENT	1 à 3	Il s'agit d'une extension de la notion habituelle de compartiment. Si Q est la quantité de produit contenu dans le bloc à l'instant T, la quantité sortant entre les temps T et T+dT est : $S = (P1 \times (Q-P2)^{+} + P3) \times dT$ où $(Q - P2)^{+} = Q - P2$ si $Q - P2 > 0$ $= 0$ sinon La possibilité d'utiliser des paramètres non constants dépendant d'autres grandeurs du système permet en particulier de décrire les phénomènes de régulation. Les paramètres P1, P2 et P3 doivent être donnés dans cet ordre dans le programme. On peut omettre P3, ou P2 et P3. Dans ce cas les paramètres omis sont supposés nuls.
JOKER	1	Comme son nom l'indique, ce bloc est assez universel. En effet c'est la valeur du paramètre qui détermine la quantité sortante :

TYPE	Nombre de PARAMETRES	FONCTION
		Si P1 est positif : $S = P1 \times dT$ Un paramètre constant simulera donc un bloc à débit constant. Un paramètre variable permettra de recréer presque toutes les fonctions. Par convention, si P1 est négatif : $S = -P1$
RETARD	illimité	Les liaisons issues de ces blocs doivent être considérées comme formées de une ou plusieurs "voies". D'autre part, contrairement à ce qui se passe pour les autres types de blocs, les paramètres dans ce cas n'agissent pas sur la sortie en temps T, mais sur les sorties futures. Leur signification est la suivante : Les paramètres doivent être considérés par couple (2 paramètres par voie). Ils sont donc en général en nombre pair. Mais dans certains cas (voir plus loin) le dernier peut être omis. Le premier paramètre de chaque couple donne une valeur de retard. Le second donne la proportion du produit entré dans le bloc au temps T qui subira ce retard. En d'autres termes s'il y a 2 n paramètres, le produit à l'entrée est divisé en n parts. La $i^{ème}$ part est égale à $P_{2i} * E$ et elle subit un retard à la traversée du bloc, de P_{2i-1}. Par exemple : RETARD(1); ALIMENTE(2); PARAMETRE(15); ET(0.6); ET(10); ET(0.4); indique que 60 % du produit entré au temps T ne ressortiront du bloc que 15 unités de temps plus tard et que les 40 % restant sortent 10 unités de temps après leur entrée. Le premier paramètre doit être supérieur à toutes les autres valeurs de retard données dans la liste et ne

TYPE	Nombre de PARAMETRES	FONCTION
RETARD (suite)		peut être spécifié VARIABLE. Tous les autres peuvent l'être. Pour les temps $T < P_1$ la valeur de la sortie d'un bloc RETARD n'est pas calculable et est présumée nulle. Un bloc supplémentaire, de type JOKER par exemple, peut être substitué temporairement à un bloc RETARD dont la sortie dans ces circonstances devrait être non nulle. Enfin, un des paramètres "retards" peut être nul. Le paramètre qui lui est associé représente alors non plus une proportion mais un débit de sortie comme pour un Joker. Le produit sortant par cette "voie" de la liaison est équitablement prélevé sur toutes les autres voies et liaisons. Exemple : RETARD(1); ALIMENTE(2); PARAMETRE(15); ET(0.6); ET(10); ET(0.4); ET(0); ET(VARIABLE); : BLOC(1); VERS(2); ILYA(5); PARAMETRE(CONSTANT); ET(0.1 * CONTENU(1,1)); indique qu'à chaque instant outre les sorties retardées ordinaires, 10 % du contenu du bloc 1 transite vers le bloc 2. Ce produit est prélevé sur toutes les autres sorties du bloc. Simplification permise : Si le retard est à l'origine d'une seule liaison et si la somme des paramètres d'ordre pair (proportions) est égale à 1, le dernier paramètre "proportion" peut être omis. Dans l'exemple précédent on pouvait supprimer ET(0.4); Si cette somme n'est pas égale à 1 une partie du produit ne sort jamais du retard qui se comporte à son égard comme un réservoir.

ANNEXE C

Listing du programme d'interprétation.

```
0000    'BEGIN''COMMENT'**************************PROLOGUE.PROCEDURES.ET.
0000    INITIALISATIONS*****;
0000    'INTEGER'NDB,INX,NMR,INXP,INXM,NP,VARIABLE,NIT,TRC,NDPT,ADS,INS,LS,N
0000    OP,RS,DI,TOUS,NOB,NBI,ITR,PTR,NO,NOE,NOEX,NDEX,TPRO,IP,IPI,NTT,ATP,N
0000    TP,TOUTES,NOBA,I,NTL,NAU,TOTAL,TOTALE,MOT,NOS,FST,MOTI,T2I,NQ;
0001    'REAL'T,TMA,HX,FO,VA,TP,HT,RND,BI,MR,PR,OX,OY,SX,SY,QX,QY,TI,MAV,MAF
0001    ,MIV,MIF,TO,MIX,MIY;
0002    'BOOLEAN'RTD,NVLD,COR,PVLD,CI,NPS,CI,QDX,QDY,ODX,ODY,PP,CST,MNX,MNY,
0002    FANT,BAT;
0003    'BOOLEAN'AX;
0004    'INTEGER''ARRAY'S2[1:100];
0005    'BOOLEAN''ARRAY'BP,BQ[1:20];
0006    'REAL''ARRAY'ST[0:02003],TAC[1:2,1:402],BID[1:200];
0007    'BOOLEAN''PROCEDURE'CLE;
0008    'CODE';
0009    'PROCEDURE'ICT;
0010    'CODE';
0011    'PROCEDURE'BASCSO;
0012    'CODE';
0013    'PROCEDURE'PLOTCURVE;
0014    'CODE';
0015    'PROCEDURE'NEWCURVE;
0016    'CODE';
0017    'PROCEDURE'SCALEX;
0018    'CODE';
0019    'PROCEDURE'SCALEY;
0020    'CODE';
0021    'PROCEDURE'MINX;
0022    'CODE';
0023    'PROCEDURE'MINY;
0024    'CODE';
0025    'PROCEDURE'PLOTAX;
0026    'CODE';
0027    'PROCEDURE'ENDLIN;
0028    'CODE';
0029    'PROCEDURE'COURBE;
0030    'CODE';
0031    'PROCEDURE'IMPLAN(CH,I1,I2);
0032    'VALUE'CH,I1,I2;
0033    'INTEGER'I1,I2;
0034    'STRING'CH;
0035    'BEGIN'TEXT(CW);
0036    PRINT(3);
0037    I:=0;
0038    'FOR'INX:=I1'STEP'1'UNTIL'I2'DO''BEGIN'EDIT("L11.5\,ST[INX]);
0039    TEXT("?\);
0040    I:=I+1;
0041    'IF'I=10'THEN''BEGIN'PRINT(1);
0042    I:=0;
0043    'END';
0044    'END';
0045    'IF'I>0'THEN'PRINT(2);
0046    'END';
0047    'PROCEDURE'BIP(I);
0048    'VALUE'I;
0049    'INTEGER'I;
0050    'BEGIN'MOT:=MOT+1;
```

```
0051    'IF'NIT=0'AND''NOT'PVLD'AND''NOT'NVLD'AND'BP[I]'THEN''BEGIN'BP[I]:='
0051    FALSE';
0052    TEXT("BIP?\);
0053    EDIT("F2.0\,I);
0054    EDIT("F9.5\,T);
0055    PRINT(1);
0056    'END';
0057    'END';
0058    'REAL''PROCEDURE'MAX(A,B);
0059    'VALUE'A,B;
0060    'REAL'A,B;
0061    MAX:='IF'A>B'THEN'A'ELSE'B;
0062    'REAL''PROCEDURE'INF(X,Y);
0063    'VALUE'X,Y;
0064    'REAL'X,Y;
0065    INF:='IF'X<Y'THEN'X'ELSE'Y;
0066    'PROCEDURE'PAC;
0067    'BEGIN'TEXT("MODELE?TROP?IMPORTANT\);
0068    PRINT(1);
0069    'GOTO'FERR;
0070    'END';
0071    'PROCEDURE'TV(I);
0072    'VALUE'I;
0073    'INTEGER'I;
0074    'BEGIN'TEXT("T?=?\);
0075    EDIT("L18.12\,T);
0076    TEXT("TI?=?\);
0077    EDIT("L18.12\,TI);
0078    PRINT(1);
0079    TEXT("REFERENCE?TROP?ANCIENNE?AU?BLOC?:?\);
0080    EDIT("F2.0\,I);
0081    PRINT(1);
0082    'GOTO'FERR;
0083    'END';
0084    'PROCEDURE'RES(K,FES);
0085    'VALUE'K;
0086    'INTEGER'K;
0087    'LABEL'FES;
0088    'BEGIN''INTEGER'SP;
0089    'BOOLEAN'EX,PD,PDB;
0090    TPRO:=K;
0091    'IF'NPS'THEN''BEGIN'IP:=IP+1;
0092    NO:='IF'S2[IP]>0'THEN'NAU'ELSE'NOE;
0093    NO:=NO+1;
0094    MOT:=MOT+1;
0095    'END';
0096    SP:=S2[IP];
0097    EX:='TRUE';
0098    PD:=PDB:='FALSE';
0099    'IF'PVLD'OR'(NPS'AND'NOEX>NO-1'AND'SP#0)'OR'NVLD'THEN'EX:='FALSE';
0100    'IF'SP>NO-NAU-1'THEN''BEGIN'PD:='TRUE';
0101    'IF''NOT'NPS'THEN''GOTO'PB;
0102    'IF'NO=NOEX'OR'(NOEX>NO'AND'NO-NAU=SP)'THEN'PDB:='TRUE';
0103    PB:'END';
0104    PVLD:=NOE#NOEX;
0105    'IF'PD'AND''NOT'PDB'THEN'PVLD:='TRUE';
0106    'IF'PDB'THEN'PVLD:='FALSE';
0107    'IF''NOT'EX'THEN''GOTO'FES;
```

```
0108      'END';
0109      'REAL''PROCEDURE'CONSTANT;
0110      'BEGIN'CONSTANT:=ST[INXP-ST[INXP]-3];
0111      CST:='TRUE';
0112      'END';
0113      'REAL''PROCEDURE'FANTOME;
0114      'BEGIN'FANTOME:=1;
0115      FANT:='TRUE';
0116      'END';
0117      'REAL''PROCEDURE'UNIF(A,B);
0118      'VALUE'A,B;
0119      'REAL'A,B;
0120      'BEGIN''REAL'RI;
0121      RND:=RND*81;
0122      'IF'RND>=MR'THEN'RND:=RND/MR;
0123      RI:=RND-ENTIER(RND);
0124      UNIF:=(B-A)*RI+A;
0125      'END';
0126      'REAL''PROCEDURE'NORMAL(M,S);
0127      'VALUE'M,S;
0128      'REAL'M,S;
0129      'BEGIN''REAL'X,Y;
0130      X:=UNIF(0,1);
0131      'IF'X=1'THEN''BEGIN'NORMAL:=99999.9999;
0132      'GOTO'SEF;
0133      'END';
0134      Y:=M+S/SQRT(8/3.14159)*LN((1+X)/(1-X));
0135      SEF:'IF'UNIF(0,1)<=0.5'THEN'Y:=2*M-Y;
0136      NORMAL:=Y;
0137      'END';
0138      'REAL''PROCEDURE'NORBOR(A,B);
0139      'VALUE'A,B;
0140      'REAL'A,B;
0141      'BEGIN''REAL'EM,ES,IX;
0142      EM:=(A+B)/2;
0143      ES:=(B-A)/6;
0144      RE:IX:=NORMAL(EM,ES);
0145      'IF'IX>B'OR'IX<A'THEN''GOTO'RE;
0146      NORBOR:=IX;
0147      'END';
0148      'PROCEDURE'NBPROD(N);
0149      'VALUE'N;
0150      'INTEGER'N;
0151      'BEGIN'MOT:=MOT+1;
0152      'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0153      NDPT:=N;
0154      FST:=N*NDB*(NDB+7);
0155      DI:=ADS:=FST+NDB+1;
0156      SEF:'END';
0157      'PROCEDURE'NBBLOCS(N);
0158      'VALUE'N;
0159      'INTEGER'N;
0160      'BEGIN'MOT:=MOT+1;
0161      'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0162      NDB:=N;
0163      INX:=1;
0164      FST:=NDPT*N*(N+7);
0165      DI:=ADS:=FST+NDB+1;
```

```
0166    SEF:'END';
0167    'PROCEDURE'PAS(H);
0168    'VALUE'H;
0169    'REAL'H;
0170    'BEGIN'RES(13,SEF);
0171    HX:=H;
0172    TI:=HX;
0173    SEF:'END';
0174    'PROCEDURE'PRODUIT(I);
0175    'VALUE'I;
0176    'INTEGER'I;
0177    'BEGIN'MOT:=MOT+1;
0178    'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0179    NOP:=I;
0180    INX:=(I-1)*NDB*(NDB+7)+1+(NOB-1)*(NDB+7);
0181    SEF:'END';
0182    'PROCEDURE'COMPARTIMENT(I);
0183    'VALUE'I;
0184    'INTEGER'I;
0185    'BEGIN'MOT:=MOT+1;
0186    'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0187    INX:=(I-1)*(NDB+7)+(NOP-1)*(NDB+7)*NDB+1;
0188    ST[INX+NDB+3]:=1;
0189    NOB:=I;
0190    SEF:'END';
0191    'PROCEDURE'RETARD(I);
0192    'VALUE'I;
0193    'INTEGER'I;
0194    'BEGIN'MOT:=MOT+1;
0195    'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0196    INX:=(I-1)*(NDB+7)+(NOP-1)*(NDB+7)*NDB+1;
0197    ST[INX+NDB+3]:=2;
0198    NOB:=I;
0199    SEF:'END';
0200    'PROCEDURE'RESERVOIR(I);
0201    'VALUE'I;
0202    'INTEGER'I;
0203    'BEGIN'MOT:=MOT+1;
0204    'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0205    INX:=(I-1)*(NDB+7)+(NOP-1)*(NDB+7)*NDB+1;
0206    ST[INX+NDB+3]:=3;
0207    NOB:=I;
0208    SEF:'END';
0209    'PROCEDURE'EMBRANCHEMENT(I);
0210    'VALUE'I;
0211    'INTEGER'I;
0212    'BEGIN'MOT:=MOT+1;
0213    'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0214    INX:=(I-1)*(NDB+7)+(NOP-1)*(NDB+7)*NDB+1;
0215    ST[INX+NDB+3]:=4;
0216    NOB:=I;
0217    SEF:'END';
0218    'PROCEDURE'JOKER(I);
0219    'VALUE'I;
0220    'INTEGER'I;
0221    'BEGIN'MOT:=MOT+1;
0222    'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0223    INX:=(I-1)*(NDB+7)+(NOP-1)*(NDB+7)*NDB+1;
```

```
0224      ST[INX+NDB+3]:=5;
0225      NOB:=1;
0226      SEF:'END';
0227      'PROCEDURE'BLOC(I);
0228      'VALUE'I;
0229      'INTEGER'I;
0230      'BEGIN'MOT:=MOT+1;
0231      'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0232      NOB:=1;
0233      INX:=(I-1)*(NDB+7)+(NOP-1)*(NDB+7)*NDB+1;
0234      SEF:'END';
0235      'PROCEDURE'ALIMENTE(I);
0236      'VALUE'I;
0237      'INTEGER'I;
0238      'BEGIN'MOT:=MOT+1;
0239      'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0240      NP:=1;
0241      INXM:=INX+I-1;
0242      NOBA:=1;
0243      'IF'ST[INXM]>0'THEN''GOTO'SEF;
0244      INXP:=INXP-ST[INXP]-3;
0245      ST[INXM]:=INXP;
0246      'IF'ST[INX+NDB+3]=2'THEN'RTD:='TRUE';
0247      SEF:'END';
0248      'PROCEDURE'PARAMETRE(K);
0249      'VALUE'K;
0250      'REAL'K;
0251      'BEGIN''INTEGER'I,J;
0252      RES(1,SEF);
0253      'IF'INXP<=0'THEN''GOTO'SEF;
0254      'IF'CST'THEN''BEGIN'ST[INXP]:=ST[INXP]+NP;
0255      NP:=1;
0256      'GOTO'SEF;
0257      'END';
0258      'IF'RTD'THEN''BEGIN'ST[INXP]:=1;
0259      ST[INXP-3]:=ADS;
0260      ST[ADS]:=ENTIER(K/HX)+1;
0261      ST[ADS+1]:=2003;
0262      ST[ADS+2]:=ST[ADS];
0263      ADS:=ADS+ST[ADS]+3;
0264      RTD:='FALSE';
0265      DI:=ADS;
0266      'IF'ADS>LS'THEN'PAC;
0267      'END';
0268      'FOR'I:=1'STEP'1'UNTIL'NP'DO''BEGIN'ST[INXP-ST[INXP]-3]:=K;
0269      'IF'INXP-ST[INXP]-3<=ADS'THEN'PAC;
0270      ST[INXP]:=ST[INXP]+1;
0271      'END';
0272      NP:=1;
0273      LS:=INF(LS,INXP-ST[INXP]-3);
0274      SEF:CST:='FALSE';
0275      'END';
0276      'PROCEDURE'ET(K);
0277      'VALUE'K;
0278      'REAL'K;
0279      PARAMETRE(K);
0280      'PROCEDURE'AINSIQUE(I);
0281      'VALUE'I;
```

```
0282    'INTEGER'I;
0283    ALIMENTE(I);
0284    'PROCEDURE'SANSISSUE;
0285    'BEGIN'MOT:=MOT+1;
0286    'END';
0287    'PROCEDURE'ILYA(I);
0288    'VALUE'I;
0289    'INTEGER'I;
0290    'BEGIN'MOT:=MOT+1;
0291    'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0292    ST[INXM]:=INXP;
0293    NP:=1;
0294    SEF:'END';
0295    'PROCEDURE'VIDANGE;
0296    MOT:=MOT+1;
0297    'PROCEDURE'CAPACITE(Q);
0298    'VALUE'Q;
0299    'REAL'Q;
0300    'BEGIN'RES(12,SEF);
0301    ST[FST+NOB]:=Q;
0302    SEF:'END';
0303    'PROCEDURE'GARDER(X);
0304    'VALUE'X;
0305    'REAL'X;
0306    'BEGIN'MOT:=MOT+1;
0307    'END';
0308    'PROCEDURE'PENDANT(TS);
0309    'VALUE'TS;
0310    'REAL'TS;
0311    'BEGIN'RES(2,SEF);
0312    'IF'RS=2'THEN''BEGIN''IF'ST[INS]=0'OR'ST[ST[INS]]<ENTIER(TS/HX)'THEN
0312    ''BEGIN'ST[INS]:=ADS;
0313    ST[ADS]:=ENTIER(TS/HX);
0314    ST[ADS+1]:=INS+1;
0315    ST[ADS+2]:=ST[ADS];
0316    ADS:=ADS+ST[ADS]+3;
0317    'END';
0318    'GOTO'FS;
0319    'END';
0320    DS:'IF'RS=0'THEN'TS:=TS+HX;
0321    DT:'IF'ST[INS]=0'THEN''BEGIN'ST[INS]:='IF'RS=-1'THEN'-ADS'ELSE'ADS;
0322    ST[ADS]:=ENTIER(TS/HX);
0323    ST[ADS+1]:='IF'RS>=0'THEN'INS-4'ELSE'INS-3;
0324    ST[ADS+2]:=ST[ADS];
0325    ADS:=ADS+ST[ADS]+3;
0326    'GOTO'FS;
0327    'END';
0328    'BEGIN''INTEGER'AD,TR;
0329    TR:=ENTIER(TS/HX)-1;
0330    'IF'RS=0'THEN''GOTO'MN;
0331    AD:=ABS(ST[INS]);
0332    TR:=TR+1;
0333    'IF'ST[AD]<TR'THEN''BEGIN'TR:=ST[AD];
0334    'IF'AD+ST[AD]+2=ADS'THEN'ADS:=AD;
0335    ST[INS]:=ADS;
0336    ST[ADS]:=ENTIER(TS/HX);
0337    ST[ADS+1]:='IF'RS=1'THEN'INS-4'ELSE'INS-3;
0338    ST[ADS+2]:=ST[ADS];
```

```
0339     ADS:=ADS+ST[ADS]+3;
0340     'END';
0341     INS:=INS-1;
0342     MN:'IF'ST[INS]=0'OR'ST[ST[INS]]<=TR'THEN''BEGIN'ST[INS]:=ADS;
0343     ST[ADS]:=TR+1;
0344     ST[ADS+1]:=INS-4;
0345     ST[ADS+2]:=ST[ADS];
0346     ADS:=ADS+ST[ADS]+3;
0347     'END';
0348     'END';
0349     FS:DI:=ADS;
0350     'IF'ADS>LS'THEN'PAC;
0351     SEF:'END';
0352     'REAL''PROCEDURE'VOLUME(I);
0353     'VALUE'I;
0354     'INTEGER'I;
0355     'BEGIN''INTEGER'J;
0356     'REAL'Q;
0357     'IF'I=0'THEN''BEGIN'Q:=0;
0358     'FOR'J:=FST+1'STEP'1'UNTIL'FST+NDB'DO'Q:=Q+ST[J];
0359     'END''ELSE'Q:=ST[FST+1];
0360     VOLUME:='IF'Q=0'THEN'CONTENU(I,TOTAL)'ELSE'Q;
0361     'END';
0362     'REAL''PROCEDURE'CONTENU(I,P);
0363     'VALUE'I,P;
0364     'INTEGER'I,P;
0365     'BEGIN'CONTENU:=1;
0366     'IF'PVLD'OR'NVLD'OR''NOT'COR'THEN''GOTO'NUL;
0367     'IF'P=0'THEN''BEGIN''INTEGER'J;
0368     'REAL'X;
0369     X:=0;
0370     'FOR'J:=(I-1)*(NDB+7)+NDB+1'STEP'NDB*(NDB+7)'UNTIL'FST'DO'X:=X+ST[J]
0370     ;
0371     CONTENU:=X;
0372     'GOTO'NUL;
0373     'END';
0374     RS:=0;
0375     INS:=(I-1)*(NDB+7)+NDB+5+(P-1)*NDB*(NDB+7);
0376     CONTENU:=ST[INS-4];
0377     NUL:'END';
0378     'REAL''PROCEDURE'ENTREE(I,P);
0379     'VALUE'I,P;
0380     'INTEGER'I,P;
0381     'BEGIN'ENTREE:=1;
0382     'IF'PVLD'OR'NVLD'OR''NOT'COR'THEN''GOTO'NUL;
0383     'IF'P=0'THEN''BEGIN''INTEGER'J;
0384     'REAL'X;
0385     X:=0;
0386     'FOR'J:=(I-1)*(NDB+7)+NDB+2'STEP'NDB*(NDB+7)'UNTIL'FST'DO'X:=X+ST[J]
0386     ;
0387     ENTREE:=X/HX;
0388     'GOTO'NUL;
0389     'END';
0390     RS:=1;
0391     INS:=(NDB+7)*(NDB*(P-1)+I)-1;
0392     ENTREE:=ST[INS-4]/HX;
0393     NUL:'END';
0394     'REAL''PROCEDURE'SORTIE(I,P);
```

```
0395      'VALUE'I,P;
0396      'INTEGER'I,P;
0397      'BEGIN'SORTIE:=1;
0398      'IF'PVLD'OR'NVLD'OR''NOT'COR'THEN''GOTO'NUL;
0399      'IF'P=0'THEN''BEGIN''INTEGER'J;
0400      'REAL'X;
0401      X:=0;
0402      'FOR'J:=(I-1)*(NDB+7)+NDB+3'STEP'NDB*(NDB+7)'UNTIL'FST'DO'X:=X+ST[J]
0402      ;
0403      SORTIE:=X/HX;
0404      'GOTO'NUL;
0405      'END';
0406      RS:=-1;
0407      INS:=(NDB+7)*(NDB*(P-1)+I)-1;
0408      SORTIE:=ST[INS-3]/HX;
0409      NUL:'END';
0410      'REAL''PROCEDURE'APPORT(I,J,P);
0411      'VALUE'I,J,P;
0412      'INTEGER'I,J,P;
0413      'BEGIN'APPORT:=1;
0414      'IF'PVLD'OR'NVLD'OR''NOT'COR'THEN''GOTO'NUL;
0415      'IF'P=0'THEN''BEGIN''INTEGER'K;
0416      'REAL'X;
0417      X:=0;
0418      'FOR'K:=(I-1)*(NDB+7)+J'STEP'NDB*(NDB+7)'UNTIL'FST'DO''IF'ST[K]>0'TH
0418      EN'X:=X+ST[ST[K]-1];
0419      APPORT:=X/HX;
0420      'GOTO'NUL;
0421      'END';
0422      RS:=2;
0423      INS:=ST[(P-1)*NDB*(NDB+7)+(I-1)*(NDB+7)+J]-2;
0424      'IF'INS<0'THEN''BEGIN'APPORT:=0;
0425      'GOTO'NUL;
0426      'END';
0427      APPORT:=ST[INS+1]/HX;
0428      NUL:'END';
0429      'REAL''PROCEDURE'CONCENTRATION(I,P);
0430      'VALUE'I,P;
0431      'INTEGER'I,P;
0432      'BEGIN''REAL'Q;
0433      Q:=VOLUME(I);
0434      CONCENTRATION:='IF'Q=0'THEN'0'ELSE'CONTENU(I,P)/Q;
0435      'END';
0436      'REAL''PROCEDURE'EXEX(I,P,TI);
0437      'VALUE'I,P,TI;
0438      'INTEGER'I,P;
0439      'REAL'TI;
0440      'BEGIN''INTEGER'J,K;
0441      J:=(NDB+7)*(NDB*(P-1)+I)-1;
0442      K:=ENTIER((T-TI)/HX);
0443      'IF'ST[ABS(ST[J])]<K'THEN'TV(I);
0444      'IF'K=0'THEN'EXEX:=SORTIE(I,P)*HX'ELSE''BEGIN''INTEGER'L,LI;
0445      L:=ABS(ST[J])+2;
0446      LI:=ST[L]+K-1;
0447      LI:='IF'LI<=ST[L-2]'THEN'LI+L'ELSE'LI+L-ST[L-2];
0448      EXEX:=ST[LI];
0449      'END';
0450      'END';
```

```
0451    'REAL''PROCEDURE'EXCONTENU(I,P,TI);
0452    'VALUE'I,P,TI;
0453    'INTEGER'I,P;
0454    'REAL'TI;
0455    'BEGIN''INTEGER'J,K,PI,X;
0456    EXCONTENU:=1;
0457    'IF'PVLD'OR'NVLD'OR''NOT'COR'THEN''GOTO'NUL;
0458    K:=ENTIER((T-TI)/HX);
0459    'IF'K=0'THEN''BEGIN'EXCONTENU:=CONTENU(I,P);
0460    'GOTO'NUL;
0461    'END';
0462    PI:=(P-1)*NDB*(NDB+7)+I*(NDB+7)-2;
0463    X:=0;
0464    'FOR'J:='IF'P=0'THEN'I*(NDB+7)-2'ELSE'PI'STEP'NDB*(NDB+7)'UNTIL''IF'
0464    P=0'THEN'FST'ELSE'PI'DO''BEGIN''INTEGER'L,LI;
0465    ATP:=ST[J];
0466    'IF'ST[ATP]<K'THEN'TV(I);
0467    L:=ATP+2;
0468    LI:=ST[L]+K-1;
0469    LI:='IF'LI<=ST[ATP]'THEN'LI+L'ELSE'LI+L-ST[ATP];
0470    X:=X+ST[LI];
0471    'END';
0472    EXCONTENU:=X;
0473    NUL:'END';
0474    'REAL''PROCEDURE'EXENTREE(I,P,TI);
0475    'VALUE'I,P,TI;
0476    'INTEGER'I,P;
0477    'REAL'TI;
0478    'BEGIN''INTEGER'J,K,PI,P2,X;
0479    EXENTREE:=1;
0480    'IF'PVLD'OR'NVLD'OR''NOT'COR'THEN''GOTO'NUL;
0481    K:=ENTIER((T-TI)/HX);
0482    'IF'K=0'THEN''BEGIN'EXENTREE:=ENTREE(I,P);
0483    'GOTO'NUL;
0484    'END';
0485    PI:=(NDB+7)*(NDB*(P-1)+I)-1;
0486    P2:='IF'P=0'THEN'0'ELSE'P-1;
0487    X:=0;
0488    'FOR'J:='IF'P=0'THEN'I*(NDB+7)-1'ELSE'PI'STEP'NDB*(NDB+7)'UNTIL''IF'
0488    P=0'THEN'FST'ELSE'PI'DO''BEGIN''INTEGER'L,LI;
0489    P2:=P2+1;
0490    ATP:=ST[J];
0491    'IF'ATP>0'THEN''BEGIN''IF'ST[ATP]<K'THEN'TV(I);
0492    L:=ATP+2;
0493    LI:=ST[L]+K-1;
0494    LI:='IF'LI<=ST[ATP]'THEN'LI+L'ELSE'LI+L-ST[ATP];
0495    X:=X+ST[LI];
0496    'END''ELSE'X:=EXCONTENU(I,P2,TI)-EXCONTENU(I,P2,TI-HX)+EXEX(I,P2,TI)
0496    +X;
0497    'END';
0498    EXENTREE:=X/HX;
0499    NUL:'END';
0500    'REAL''PROCEDURE'EXSORTIE(I,P,TI);
0501    'VALUE'I,P,TI;
0502    'INTEGER'I,P;
0503    'REAL'TI;
0504    'BEGIN''INTEGER'J,K,PI,P2,X;
0505    EXSORTIE:=1;
```

```
0506    'IF'PVLD'OR'NVLD'OR''NOT'COR'THEN''GOTO'NUL;
0507    K:=ENTIER((T-TI)/HX);
0508    'IF'K=0'THEN''BEGIN'EXSORTIE:=SORTIE(I,P);
0509    'GOTO'NUL;
0510    'END';
0511    PI:=(NDB+7)*(NDB*(P-1)+1)-1;
0512    P2:='IF'P=0'THEN'0'ELSE'P-1;
0513    X:=0;
0514    'FOR'J:='IF'P=0'THEN'I*(NDB+7)-1'ELSE'PI'STEP'NDB*(NDB+7)'UNTIL''IF'
0514    P=0'THEN'FST'ELSE'PI'DO''BEGIN''INTEGER'L,LI;
0515    P2:=P2+1;
0516    ATP:=-ST[J];
0517    'IF'ATP>0'THEN''BEGIN''IF'ST[ATP]<K'THEN'TV(I);
0518    L:=ATP+2;
0519    LI:=ST[L]+K-1;
0520    LI:='IF'LI<=ST[ATP]'THEN'LI+L'ELSE'LI+L-ST[ATP];
0521    X:=X+ST[LI];
0522    'END''ELSE'X:=EXCONTENU(I,P2,TI-HX)-EXCONTENU(I,P2,TI)+EXENTREE(I,P2
0522    ,TI)*HX+X;
0523    'END';
0524    EXSORTIE:=X/HX;
0525    NUL:'END';
0526    'REAL''PROCEDURE'EXAPPORT(I,J,P,TI);
0527    'VALUE'I,J,P,TI;
0528    'INTEGER'I,J,P;
0529    'REAL'TI;
0530    'BEGIN''INTEGER'K,L,PI,X;
0531    EXAPPORT:=1;
0532    'IF'PVLD'OR'NVLD'OR''NOT'COR'THEN''GOTO'NUL;
0533    L:=ENTIER((T-TI)/HX);
0534    'IF'L=0'THEN''BEGIN'EXAPPORT:=APPORT(I,J,P);
0535    'GOTO'NUL;
0536    'END';
0537    X:=0;
0538    'FOR'PI:='IF'P=0'THEN'1'ELSE'P'STEP'1'UNTIL''IF'P=0'THEN'NDPT'ELSE'P
0538    'DO''BEGIN''INTEGER'L2,LI;
0539    K:=ABS(ST[(PI-1)*NDB*(NDB+7)+(I-1)*(NDB+7)+J])-2;
0540    ATP:=ST[K];
0541    'IF'ST[ATP]<L'THEN'TV(I);
0542    L2:=ATP+2;
0543    LI:=ST[L2]+L-1;
0544    LI:='IF'LI<=ST[ATP]'THEN'LI+L2'ELSE'LI+L2-ST[ATP];
0545    X:=X+ST[LI];
0546    'END';
0547    EXAPPORT:=X/HX;
0548    NUL:'END';
0549    'PROCEDURE'DEPARTA(TI);
0550    'VALUE'TI;
0551    'REAL'TI;
0552    'BEGIN'RES(3,SEF);
0553    'IF'T=-99999'THEN''BEGIN''IF'IPI<0'THEN''BEGIN'IPI:=IP-1;
0554    MOTI:=MOT-1;
0555    'END';
0556    NBI:=ENTIER(TI/HX);
0557    T:=TP:=TMA:=TO:=TI;
0558    'FOR'I:=1'STEP'1'UNTIL'200'DO'BID[I]:=TMA;
0559    'END';
0560    NTL:=0;
```

```
0561     NOP:=NOB:=1;
0562     INX:=1;
0563     SEF:'END';
0564     'PROCEDURE'STOPA(T2);
0565     'VALUE'T2;
0566     'REAL'T2;
0567     'BEGIN'RES(4,SEF);
0568     'IF'NBI>ENTIER(T2/HX)'THEN''GOTO'FSIM;
0569     SEF:'END';
0570     'PROCEDURE'STOP;
0571     'BEGIN'MOT:=MOT+1;
0572     'IF'PVLD'OR'NVLD'OR'NIT=1'THEN''GOTO'SEF;
0573     'GOTO'FSIM;
0574     SEF:'END';
0575     'PROCEDURE'TPETIT;
0576     'BEGIN'MOT:=MOT+1;
0577     'IF'T>TMA'THEN''BEGIN'NVLD:='TRUE';
0578     'GOTO'TG;
0579     'END';
0580     MOTI:=MOT-1;
0581     NTP:=NTP+1;
0582     ATP:=0;
0583     IPI:=IP;
0584     TG:'END';
0585     'PROCEDURE'FAIRE(Q);
0586     'VALUE'Q;
0587     'REAL'Q;
0588     'BEGIN'MOT:=MOT+1;
0589     'IF''NOT'PVLD'AND''NOT'NVLD'THEN''BEGIN'TI:=NTP*HX;
0590     'IF'ATP<=0'THEN''BEGIN'TEXT("FAIRE?:?ARGUMENT?ILLICITE.\);
0591     PRINT(1);
0592     'GOTO'FERR;
0593     'END';
0594     'END';
0595     'END';
0596     'PROCEDURE'FGALE(E);
0597     'VALUE'E;
0598     'REAL'F;
0599     'BEGIN'RES(19,SEF);
0600     'IF'ST[ATP]>NTT'THEN'NTT:=ST[ATP];
0601     'IF'NTP=0'THEN'ST[ST[ATP+1]]:=E'ELSE''IF'NTP<=ST[ATP]'THEN''BEGIN''I
0601     NTEGER'L,LI;
0602     LI:='IF'NTP=1'THEN'ATP+2+ST[ATP]'ELSE'ATP+1+NTP;
0603     ST[LI]:=E;
0604     'END';
0605     TI:=HX;
0606     ATP:=0;
0607     SEF:'END';
0608     'PROCEDURE'FINTPETIT;
0609     'BEGIN''INTEGER'I;
0610     MOT:=MOT+1;
0611     TI:=HX;
0612     'IF'NTP<NTT'AND'T<=TMA'THEN''BEGIN''FOR'I:=IPI+1'STEP'1'UNTIL'IP'DO'
0612     S2[I]:=0;
0613     IP:=IPI;
0614     MOT:=MOTI;
0615     'GOTO'EXPERIENCE;
0616     'END';
```

```
0617    NVLD:='FALSE';
0618    'END';
0619    'PROCEDURE'INITIALEMENT;
0620    'BEGIN''IF'IPI<0'THEN''BEGIN'IPI:=IP;
0621    MOTI:=MOT;
0622    'END';
0623    MOT:=MOT+1;
0624    'IF'T>TMA'THEN'NVLD:='TRUE';
0625    'END';
0626    'PROCEDURE'FININIT;
0627    'BEGIN'MOT:=MOT+1;
0628    NVLD:='FALSE';
0629    'END';
0630    'PROCEDURE'QUAND(B);
0631    'VALUE'B;
0632    'BOOLEAN'B;
0633    'BEGIN'MOT:=MOT+1;
0634    PP:='FALSE';
0635    NQ:=NQ+1;
0636    BQ[NQ]:=NVLD;
0637    NQ:=NQ+1;
0638    BQ[NQ]:='NOT'B;
0639    NVLD:=NVLD'OR''NOT'B;
0640    'END';
0641    'PROCEDURE'AUTEMPS(TI);
0642    'VALUE'TI;
0643    'REAL'TI;
0644    'BEGIN''INTEGER'N;
0645    'BOOLEAN'BO;
0646    MOT:=MOT+1;
0647    PP:='FALSE';
0648    BO:='TRUE';
0649    N:=ENTIER(TI/HX);
0650    'IF'NBI=N'THEN'BO:='FALSE';
0651    NQ:=NQ+1;
0652    BQ[NQ]:=NVLD;
0653    NQ:=NQ+1;
0654    BQ[NQ]:=BO;
0655    NVLD:=NVLD'OR'BO;
0656    'END';
0657    'PROCEDURE'AUTREMENT;
0658    'BEGIN'MOT:=MOT+1;
0659    BQ[NQ]:='NOT'BQ[NQ];
0660    NVLD:=BQ[NQ-1]'OR'BQ[NQ];
0661    'END';
0662    'PROCEDURE'FINQUAND;
0663    'BEGIN'MOT:=MOT+1;
0664    'IF'NQ=0'THEN''BEGIN'TEXT("MANQUE?UN?QUAND\);
0665    PRINT(1);
0666    'GOTO'FERR;
0667    'END';
0668    NVLD:=BQ[NQ-1];
0669    NQ:=NQ-2;
0670    'IF'PP'THEN'COR:='TRUE';
0671    'END';
0672    'PROCEDURE'SIMULATION(I);
0673    'VALUE'I;
0674    'INTEGER'I;
```

```
0675    'BEGIN'NOE:='IF'I=0'THEN'NOEX'ELSE'I;
0676    MOT:=MOT+1;
0677    'IF'NOEX=1'THEN''BEGIN''IF'NOE>NDEX'THEN'NDEX:=NOE;
0678    'END';
0679    NAU:='IF'I=0'THEN'1'ELSE'I;
0680    PVLD:=NOE#NOEX;
0681    'END';
0682    'PROCEDURE'A(I);
0683    'VALUE'I;
0684    'INTEGER'I;
0685    'BEGIN'MOT:=MOT+1;
0686    NOE:='IF'NOEX>=NAU'AND'NOEX<=I'THEN'NOEX'ELSE'I;
0687    'IF'NOEX=I'AND'I>NDEX'THEN'NDEX:=I;
0688    NAU:=I;
0689    PVLD:=NOE#NOEX;
0690    'END';
0691    'PROCEDURE'PUIS(X);
0692    'VALUE'X;
0693    'REAL'X;
0694    'BEGIN''INTEGER'SP;
0695    'SWITCH'AIG:=T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,T13,T14,T15,T16,
0695    T17,T18,T19,T20,T21,T22,T23,T24,T25;
0696    MOT:=MOT+1;
0697    'IF'NOEX=1'AND'T<=TMA'THEN''BEGIN'S2[IP]:=S2[IP]+1;
0698    'IF'S2[IP]+NAU>NDEX'THEN'NDEX:=NDEX+1;
0699    'GOTO'TF;
0700    'END';
0701    'IF'FANT'THEN''GOTO'TT;
0702    'IF'NVLD'THEN''GOTO'TT;
0703    NO:=NO+1;
0704    'IF''NOT'PVLD'THEN''BEGIN'NPS:='FALSE';
0705    'GOTO'AIG[TPRO];
0706    T1:PARAMETRE(X);
0707    'GOTO'TF;
0708    T2:PENDANT(X);
0709    'GOTO'TF;
0710    T3:DEPARTA(X);
0711    'GOTO'TF;
0712    T4:STOPA(X);
0713    'GOTO'TF;
0714    T5:CONTIENT(X);
0715    'GOTO'TF;
0716    T6:SYNTHETISE(X);
0717    'GOTO'TF;
0718    T7:TRACER(X);
0719    'GOTO'TF;
0720    T8:ENFONCTIONDE(X);
0721    'GOTO'TF;
0722    T9:DEPUIS(X);
0723    'GOTO'TF;
0724    T10:TOUSLES(X);
0725    'GOTO'TF;
0726    T11:JUSQUA(X);
0727    'GOTO'TF;
0728    T12:CAPACITE(X);
0729    'GOTO'TF;
0730    T13:PAS(X);
0731    'GOTO'TF;
```

```
0732    TI4:ETAT(X);
0733    'GOTO'TF;
0734    TI5:ECHY(X);
0735    'GOTO'TF;
0736    TI6:ECHX(X);
0737    'GOTO'TF;
0738    TI7:QUOTX(X);
0739    'GOTO'TF;
0740    TI8:QUOTY(X);
0741    'GOTO'TF;
0742    TI9:EGALE(X);
0743    'GOTO'TF;
0744    T20:DEBITE(X);
0745    'GOTO'TF;
0746    T21:ORIGX(X);
0747    'GOTO'TF;
0748    T22:ORIGY(X);
0749    'GOTO'TF;
0750    T23:INJECTER(X);
0751    'GOTO'TF;
0752    T24:MINIMX(X);
0753    'GOTO'TF;
0754    T25:MINIMY(X);
0755    'GOTO'TF;
0756    'END';
0757    TT:PVLD:=NOE#NOEX;
0758    SP:=S2[IP];
0759    'IF'SP>NO-NAU-1'THEN''BEGIN''IF'NO=NOEX'OR'(NOEX>NO'AND'NO-NAU=SP)'T
0759    HEN'PVLD:='FALSE''ELSE'PVLD:='TRUE';
0760    'END';
0761    TF:NPS:='TRUE';
0762    CST:='FALSE';
0763    FANT:='FALSE';
0764    'END';
0765    'PROCEDURE'COUPER(I,J);
0766    'VALUE'I,J;
0767    'INTEGER'I,J;
0768    'BEGIN''INTEGER'ADL,IT,JT,II,JI,IM,JM,AM,PI,PT,PM;
0769    MOT:=MOT+1;
0770    PP:='TRUE';
0771    'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0772    'IF''NOT'COR'OR'NIT=1'THEN''GOTO'PEF;
0773    'IF'I=TOUS'THEN''BEGIN'II:=1;
0774    IT:=NDB;
0775    'END''ELSE''BEGIN'II:=I;
0776    IT:=I;
0777    'END';
0778    'FOR'IM:=II'STEP'1'UNTIL'IT'DO''BEGIN''IF'J=TOUS'THEN''BEGIN'JI:=1;
0779    JT:=NDB;
0780    'END''ELSE''BEGIN'JI:=J;
0781    JT:=J;
0782    'END';
0783    'FOR'JM:=JI'STEP'1'UNTIL'JT'DO''BEGIN''IF'NOP=TOUS'THEN''BEGIN'PI:=1
0783    ;
0784    PT:=NDPT;
0785    'END''ELSE'PI:=PT:=NOP;
0786    'FOR'PM:=PI'STEP'1'UNTIL'PT'DO''BEGIN'ADL:=(PM-1)*NDB*(NDB+7)+(IM-1)
0786    *(NDB+7)+JM;
```

```
787     'IF'ST[ADL]>0'THEN'ST[ADL]:=-ST[ADL];
788     'END';
789     'END';
790     'END';
791     PEF:COR:='TRUE';
792     SEF:'END';
793     'PROCEDURE'EXCLURE(I);
794     'VALUE'I;
795     'INTEGER'I;
796     'BEGIN''INTEGER'J;
797     MOT:=MOT+1;
798     PP:='TRUE';
799     'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
800     'IF''NOT'COR'OR'NIT=I'THEN''GOTO'PEF;
801     'FOR'J:=1'STEP'1'UNTIL'NDB'DO''BEGIN'MOT:=MOT-1;
802     COUPER(I,J);
803     MOT:=MOT-1;
804     COUPER(J,I);
805     'END';
806     'FOR'J:='IF'I=0'THEN'1'ELSE'NOP'STEP'1'UNTIL''IF'I=0'THEN'NDPT'ELSE'
806     NOP'DO'ST[(NDB+7)*(I+NDB*(J-1))]:=-99999;
807     PEF:COR:='TRUE';
808     SEF:'END';
809     'PROCEDURE'RETABLIR(I,J);
810     'VALUE'I,J;
811     'INTEGER'I,J;
812     'BEGIN''INTEGER'ADL,IT,JT,II,JI,IM,JM,AM,PI,PT,PM;
813     MOT:=MOT+1;
814     PP:='TRUE';
815     'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
816     'IF''NOT'COR'OR'NIT=1'THEN''GOTO'PEF;
817     'IF'I=TOUS'THEN''BEGIN'II:=1;
818     IT:=NDB;
819     'END''ELSE''BEGIN'II:=I;
820     IT:=I;
821     'END';
822     'FOR'IM:=II'STEP'1'UNTIL'IT'DO''BEGIN''IF'J=TOUS'THEN''BEGIN'JI:=1;
823     JT:=NDB;
824     'END''ELSE''BEGIN'JI:=J;
825     JT:=J;
826     'END';
827     'FOR'JM:=JI'STEP'1'UNTIL'JT'DO''BEGIN''IF'NOP=TOUS'THEN''BEGIN'PI:=1
827     ;
828     PT:=NDPT;
829     'END''ELSE'PI:=PT:=NOP;
830     'FOR'PM:=PI'STEP'1'UNTIL'PT'DO''BEGIN'ADL:=(PM-1)*NDB*(NDB+7)+(IM-1)
830     *(NDB+7)+JM;
831     'IF'ST[ADL]<0'THEN'ST[ADL]:=-ST[ADL];
832     'END';
833     'END';
834     'END';
835     PEF:COR:='TRUE';
836     SEF:'END';
837     'PROCEDURE'REINCLURE(I);
838     'VALUE'I;
839     'INTEGER'I;
840     'BEGIN''INTEGER'J;
841     MOT:=MOT+1;
```

```
0842    PP:='TRUE';
0843    'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0844    'IF''NOT'COR'OR'NIT=1'THEN''GOTO'PEF;
0845    'FOR'J:=1'STEP'1'UNTIL'NDB'DO''BEGIN'MOT:=MOT-1;
0846    RETABLIR(I,J);
0847    MOT:=MOT-1;
0848    RETABLIR(J,I);
0849    'END';
0850    'FOR'J:='IF'I=0'THEN'1'ELSE'NOP'STEP'1'UNTIL''IF'I=0'THEN'NDPT'ELSE'
0850    NOP'DO'ST[(NDB+7)*(I+NDB*(J-1))]:=0;
0851    PEF:COR:='TRUE';
0852    SEF:'END';
0853    'PROCEDURE'CONTIENT(X);
0854    'VALUE'X;
0855    'REAL'X;
0856    'BEGIN'RES(5,SEF);
0857    ST[INX+NDB]:=X;
0858    SEF:'END';
0859    'PROCEDURE'VIDER(I);
0860    'VALUE'I;
0861    'INTEGER'I;
0862    'BEGIN'MOT:=MOT+1;
0863    PP:='TRUE';
0864    'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0865    'IF''NOT'COR'THEN''GOTO'PEF;
0866    INX:=(I-1)*(NDB+7)+(NOP-1)*NDB*(NDB+7)+1;
0867    ST[INX+NDB]:=0;
0868    PEF:COR:='TRUE';
0869    SEF:'END';
0870    'PROCEDURE'DEBITE(Q);
0871    'VALUE'Q;
0872    'REAL'Q;
0873    'BEGIN'RES(20,SEF);
0874    PR:=Q;
0875    SEF:'END';
0876    'PROCEDURE'VERS(I);
0877    'VALUE'I;
0878    'INTEGER'I;
0879    'BEGIN''INTEGER'INJ;
0880    MOT:=MOT+1;
0881    'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
0882    INXM:=INX+I-1;
0883    INXP:=ST[INXM];
0884    'IF'PR<0'THEN''BEGIN'NOBA:=1;
0885    'IF'INXP<=0'THEN''GOTO'SEF;
0886    NP:=1;
0887    ST[INXP]:='IF'ST[INX+NDB+3]=2'THEN'1'ELSE'0;
0888    'END''ELSE''BEGIN''IF'INXP>0'THEN''BEGIN'ST[INX+NDB+2]:=ST[INX+NDB+2
0888    ]+PR*HX;
0889    INJ:=(NOP-1)*NDB*(NDB+7)+(I-1)*(NDB+7)+NDB+1;
0890    ST[INJ]:=ST[INJ]+PR*HX;
0891    ST[INXP-1]:=PR*HX;
0892    'END';
0893    'END';
0894    SEF:PR:=-1;
0895    'END';
0896    'PROCEDURE'SYNTHETISE(Q);
0897    'VALUE'Q;
```

```
0898    'REAL'Q;
0899    'BEGIN''INTEGER'AM,SP;
0900    'BOOLEAN'NVDI;
0901    NVDI:=NVLD;
0902    NVLD:='FALSE';
0903    RES(6,SEF);
0904    NVLD:=NVDI;
0905    NOS:=NOS+1;
0906    'IF'NIT=0'AND'NVLD'THEN''GOTO'SEF;
0907    'IF''NOT'COR'THEN''GOTO'PEF;
0908    'IF'NIT=1'THEN''BEGIN''FOR'AM:=DI+1'STEP'5'UNTIL'ADS-4'DO''IF'ST[AM]
0908    =NOS'THEN''BEGIN'ST[AM+3]:=Q*HX;
0909    'GOTO'PEF;
0910    'END';
0911    'END''ELSE''BEGIN'ST[ADS]:=NOB;
0912    ST[ADS+1]:=NOS;
0913    ST[ADS+2]:=NOP;
0914    ST[ADS+3]:=ST[ADS+4]:=Q*HX;
0915    ADS:=ADS+5;
0916    'IF'ADS>LS'THEN'PAC;
0917    'END';
0918    PEF:COR:='TRUE';
0919    SEF:NVLD:=NVDI;
0920    'END';
0921    'PROCEDURE'UTILISE(Q);
0922    'VALUE'Q;
0923    'REAL'Q;
0924    SYNTHETISE(-Q);
0925    'PROCEDURE'PERFUSER(Q);
0926    'VALUE'Q;
0927    'REAL'Q;
0928    SYNTHETISE(Q);
0929    'PROCEDURE'INJECTER(Q);
0930    'VALUE'Q;
0931    'REAL'Q;
0932    'BEGIN''INTEGER'AM;
0933    PP:='TRUE';
0934    RES(23,SEF);
0935    'IF''NOT'COR'OR'NIT=1'THEN''GOTO'PEF;
0936    ST[ADS]:=NOB;
0937    ST[ADS+1]:=-1;
0938    ST[ADS+2]:=NOP;
0939    ST[ADS+3]:=ST[ADS+4]:=Q;
0940    ADS:=ADS+5;
0941    'IF'ADS>LS'THEN'PAC;
0942    PEF:COR:='TRUE';
0943    SEF:'END';
0944    'PROCEDURE'TRACER(F);
0945    'VALUE'F;
0946    'REAL'F;
0947    'BEGIN'PP:='TRUE';
0948    RES(7,SEF);
0949    'IF'COR'THEN''BEGIN'FO:=F;
0950    BAT:='FALSE';
0951    'END';
0952    COR:='TRUE';
0953    SEF:'END';
0954    'PROCEDURE'ENFONCTIONDE(V);
```

```
0955      'VALUE'V;
0956      'REAL'V;
0957      'BEGIN''INTEGER'I,J;
0958      PP:='TRUE';
0959      RES(8,SEF);
0960      'IF'NIT=1'OR'BAT'THEN''GOTO'PEF;
0961      'IF'COR'THEN''BEGIN'ITR:=ITR+1;
0962      'IF'ITR=PTR'THEN''BEGIN'TRC:=TRC+1;
0963      'IF'TRC=403'THEN''BEGIN'J:=1;
0964      'FOR'I:=2'STEP'1'UNTIL'201'DO''BEGIN'J:=J+2;
0965      TAC[1,I]:=TAC[1,J];
0966      TAC[2,I]:=TAC[2,J];
0967      'END';
0968      PTR:=2*PTR;
0969      TRC:=202;
0970      'END';
0971      'IF'TRC=1'THEN''BEGIN'MAV:=MIV:=V;
0972      MAF:=MIF:=F0;
0973      'END''ELSE''BEGIN'MAV:=MAX(MAV,V);
0974      MAF:=MAX(MAF,F0);
0975      MIV:=-MAX(-MIV,-V);
0976      MIF:=-MAX(-MIF,-F0);
0977      'END';
0978      TAC[1,TRC]:=V;
0979      TAC[2,TRC]:=F0;
0980      ITR:=0;
0981      'END';
0982      'END';
0983      PEF:COR:='TRUE';
0984      SEF:PP:='TRUE';
0985      BAT:='TRUE';
0986      'END';
0987      'PROCEDURE'QUOTX(Q);
0988      'VALUE'Q;
0989      'REAL'Q;
0990      'BEGIN'RES(17,SEF);
0991      QX:=Q;
0992      QDX:='FALSE';
0993      SEF:'END';
0994      'PROCEDURE'QUOTY(Q);
0995      'VALUE'Q;
0996      'REAL'Q;
0997      'BEGIN'RES(18,SEF);
0998      QY:=Q;
0999      QDY:='FALSE';
1000      SEF:'END';
1001      'PROCEDURE'ECHY(E);
1002      'VALUE'E;
1003      'REAL'E;
1004      'BEGIN'RES(15,SEF);
1005      SY:=E;
1006      SEF:'END';
1007      'PROCEDURE'ECHX(E);
1008      'VALUE'E;
1009      'REAL'E;
1010      'BEGIN'RES(16,SEF);
1011      SX:=E;
1012      SEF:'END';
```

```
013     'PROCEDURE'ORIGX(0);
014     'VALUE'0;
015     'REAL'0;
016     'BEGIN'RES(21,SEF);
017     0X:=0;
018     ODX:='FALSE';
019     SEF:'END';
020     'PROCEDURE'ORIGY(0);
021     'VALUE'0;
022     'REAL'0;
023     'BEGIN'RES(22,SEF);
024     OY:=0;
025     ODY:='FALSE';
026     SFF:'END';
027     'PROCEDURE'MINIMY(MX);
028     'VALUE'MX;
029     'REAL'MX;
030     'BEGIN'RES(25,SEF);
031     MIY:=MX;
032     MNY:='TRUE';
033     SFF:'END';
034     'PROCEDURE'MINIMX(MX);
035     'VALUE'MX;
036     'REAL'MX;
037     'BEGIN'RES(24,SEF);
038     MIX:=MX;
039     MNX:='TRUE';
040     SEF:'END';
041     'PROCEDURE'SURIMP;
042     'BEGIN'MOT:=MOT+1;
043     'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
044     CI:='TRUE';
045     SEF:'END';
046     'PROCEDURE'SURTAB;
047     'BEGIN'MOT:=MOT+1;
048     'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
049     CI:='FALSE';
050     SEF:'END';
051     'PROCEDURE'SANSAX;
052     'BEGIN'MOT:=MOT+1;
053     'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
054     AX:='FALSE';
055     SFF:'END';
056     'PROCEDURE'DEPUIS(TI);
057     'VALUE'TI;
058     'REAL'TI;
059     'BEGIN'RES(9,SEF);
060     'IF'NBI<ENTIER(TI/HX)+NIT'THEN''BEGIN'COR:='FALSE';
061     TP:=TI;
062     'END';
063     SEF:'END';
064     'PROCEDURE'TOUSLES(PT);
065     'VALUE'PT;
066     'REAL'PT;
067     'BEGIN'RES(10,SEF);
068     NTL:=NTL+1;
069     'IF'COR'THEN''BEGIN''IF'T-NIT*HX<BID[NTL]'THEN'COR:='FALSE''ELSE''BE
069     GIN''IF'NIT=1'THEN'BID[NTL]:=BID[NTL]+PT;
```

```
1070    'END';
1071    'END''ELSE'BID[NTL]:=TP;
1072    TP:=TMA;
1073    SEF:'END';
1074    'PROCEDURE'JUSQUA(T2);
1075    'VALUE'T2;
1076    'REAL'T2;
1077    'BEGIN'RES(11,SEF);
1078    T21:=ENTIER(T2/HX);
1079    'IF'NBI>=T21+NIT'THEN'COR:='FALSE';
1080    SEF:'END';
1081    'PROCEDURE'COMPRIS;
1082    'BEGIN'MOT:=MOT+1;
1083    'IF'PVLD'OR'NVLD'THEN''GOTO'SEF;
1084    'IF'NBI=T21+NIT'THEN'COR:='TRUE';
1085    SEF:T21:=-99999;
1086    'END';
1087    'PROCEDURE'AFFICHER(F);
1088    'VALUE'F;
1089    'REAL'F;
1090    TRACER(F);
1091    'PROCEDURE'INTITULE(CH);
1092    'VALUE'CH;
1093    'STRING'CH;
1094    'BEGIN'MOT:=MOT+1;
1095    'IF'PVLD'OR'NVLD'OR'CLF(1)'THEN''GOTO'SEF;
1096    'IF'NIT=1'OR'BAT'THEN''GOTO'PEF;
1097    'IF'COR'THEN''BEGIN'TEXT(CH);
1098    SPACE(5);
1099    EDIT("L18.12\,F0);
1100    SPACE(5);
1101    TEXT("T=\);
1102    EDIT("F6.2\,T);
1103    PRINT(1);
1104    'END';
1105    PEF:COR:='TRUE';
1106    SEF:PP:='TRUE';
1107    BAT:='TRUE';
1108    'END';
1109    'PROCEDURE'ETAT(TI);
1110    'VALUE'TI;
1111    'REAL'TI;
1112    'BEGIN'PP:='TRUE';
1113    RES(14,SEF);
1114    'IF'CLF(1)'THEN''GOTO'SEF;
1115    'IF''NOT'COR'THEN''GOTO'PEF;
1116    'IF'T>TI-HX'AND'T<=TI'AND'NIT=0'THEN''BEGIN''INTEGER'P,I,J;
1117    SEF:BASCSO;
1118    PRINT(5);
1119    TEXT("T=\);
1120    EDIT("F5.2\,T);
1121    PRINT(1);
1122    'FOR'P:=1'STEP'1'UNTIL'NDPT'DO''BEGIN'TEXT("PRODUIT?\);
1123    EDIT("F2.0\,P);
1124    PRINT(2);
1125    TEXT("TYPE????:\);
1126    'FOR'I:=1'STEP'1'UNTIL'NDB'DO''BEGIN''INTEGER'K;
1127    K:=(P-1)*NDB*(NDB+7)+(I-1)*(NDB+7)+NDB+4;
```

```
1128     'IF'ST[K]=0'THEN''BEGIN'SPACE(10);
1129     'GOTO'FT;
1130     'END';
1131     'IF'ST[K]=1'THEN''BEGIN'TEXT("?COMPART.?\);
1132     'GOTO'FT;
1133     'END';
1134     'IF'ST[K]=2'THEN''BEGIN'TEXT("??RETARD??\);
1135     'GOTO'FT;
1136     'END';
1137     'IF'ST[K]=3'THEN''BEGIN'TEXT("RESERVOIR?\);
1138     'GOTO'FT;
1139     'END';
1140     'IF'ST[K]=4'THEN''BEGIN'TEXT("??EMBRAN.?\);
1141     'GOTO'FT;
1142     'END';
1143     'IF'ST[K]=5'THEN'TEXT("???JOKER??\);
1144     FT:TEXT(":\);
1145     'END';
1146     PRINT(2);
1147     SPACE(4);
1148     TEXT("-----\);
1149     'FOR'I:=1'STEP'1'UNTIL'11*NDB'DO'TEXT("-\);
1150     PRINT(1);
1151     TEXT("BLOC.???:\);
1152     'FOR'I:=1'STEP'1'UNTIL'NDB'DO''BEGIN'SPACE(10);
1153     TEXT(":\);
1154     'END';
1155     PRINT(1);
1156     SPACE(5);
1157     TEXT(".OR:\);
1158     'FOR'I:=1'STEP'1'UNTIL'NDB'DO''BEGIN'SPACE(10);
1159     TEXT(":\);
1160     'END';
1161     PRINT(1);
1162     SPACE(6);
1163     TEXT(".?:\);
1164     'FOR'I:=1'STEP'1'UNTIL'NDB'DO''BEGIN'SPACE(4);
1165     EDIT("F2.0\,I);
1166     SPACE(4);
1167     TEXT(":\);
1168     'END';
1169     PRINT(1);
1170     SPACE(4);
1171     TEXT("FX?.:\);
1172     'FOR'I:=1'STEP'1'UNTIL'NDB'DO''BEGIN'SPACE(10);
1173     TEXT(":\);
1174     'END';
1175     PRINT(1);
1176     SPACE(4);
1177     TEXT("-----\);
1178     'FOR'I:=1'STEP'1'UNTIL'11*NDB'DO'TEXT("-\);
1179     PRINT(1);
1180     'FOR'I:=1'STEP'1'UNTIL'NDB'DO''BEGIN'SPACE(8);
1181     TEXT(":\);
1182     'FOR'J:=1'STEP'1'UNTIL'NDB'DO''BEGIN'SPACE(10);
1183     TEXT(":\);
1184     'END';
1185     PRINT(1);
```

```
1186    SPACE(5);
1187    EDIT("F2.0\,I);
1188    TEXT("?:\);
1189    'FOR'J:=(P-1)*NDB*(NDB+7)+1'STEP'NDB+7'UNTIL'P*(NDB+7)*NDB-7'DO''BEG
1189    IN''IF'ST[J]<=0'THEN'SPACE(10)'ELSE'EDIT("L10.4\,ST[ST[J]-11);
1190    TEXT(":\);
1191    'END';
1192    PRINT(1);
1193    SPACE(4);
1194    TEXT("----\);
1195    'FOR'J:=1'STEP'1'UNTIL'11*NDB'DO'TEXT("-\);
1196    PRINT(1);
1197    'END';
1198    SPACE(8);
1199    TEXT(":\);
1200    'FOR'J:=1'STEP'1'UNTIL'NDB'DO''BEGIN'SPACE(10);
1201    TEXT(":\);
1202    'END';
1203    PRINT(1);
1204    TEXT("?CONTENU:\);
1205    'FOR'I:=1'STEP'1'UNTIL'NDB'DO''BEGIN'J:=(P-1)*NDB*(NDB+7)+(I-1)*(NDB
1205    +7)+NDB+1;
1206    EDIT("L10.4\,ST[J]);
1207    TEXT(":\);
1208    'END';
1209    PRINT(1);
1210    SPACE(4);
1211    TEXT("----\);
1212    'FOR'I:=1'STEP'1'UNTIL'11*NDB'DO'TEXT("-\);
1213    PRINT(1);
1214    SPACE(8);
1215    TEXT(":\);
1216    'FOR'J:=1'STEP'1'UNTIL'NDB'DO''BEGIN'SPACE(10);
1217    TEXT(":\);
1218    'END';
1219    PRINT(1);
1220    TEXT("?ENTREE?:\);
1221    'FOR'I:=1'STEP'1'UNTIL'NDB'DO''BEGIN'J:=(P-1)*NDB*(NDB+7)+(I-1)*(NDB
1221    +7)+NDB+2;
1222    EDIT("L10.4\,ST[J]);
1223    TEXT(":\);
1224    'END';
1225    PRINT(1);
1226    SPACE(4);
1227    TEXT("----\);
1228    'FOR'I:=1'STEP'1'UNTIL'11*NDB'DO'TEXT("-\);
1229    PRINT(1);
1230    SPACE(8);
1231    TEXT(":\);
1232    'FOR'J:=1'STEP'1'UNTIL'NDB'DO''BEGIN'SPACE(10);
1233    TEXT(":\);
1234    'END';
1235    PRINT(1);
1236    TEXT("?SORTIE?:\);
1237    'FOR'I:=1'STEP'1'UNTIL'NDB'DO''BEGIN'J:=(P-1)*NDB*(NDB+7)+(I-1)*(NDB
1237    +7)+NDB+3;
1238    EDIT("L10.4\,ST[J]);
1239    TEXT(":\);
```

```
240     'END';
241     PRINT(1);
242     SPACE(4);
243     TEXT("----\);
244     'FOR'I:=1'STEP'1'UNTIL'11*NDB'DO'TEXT("-\);
245     PRINT(1);
246     PRINT(2);
247     'END';
248     BASCSO;
249     'END';
250     PEF:COR:='TRUE';
251     SEF:'END';
252     'PROCEDURE'EXECUTION;
253     'BEGIN''INTEGER'IBA,IL,IBP,ADP,IBV,INXV;
254     'REAL'SI,SI,SN,RJK;
255     DEX:MOT:=MOT+1;
256     'IF'CLE(0)'THEN''GOTO'FSIM;
257     'IF'NO>0'THEN''BEGIN'TEXT("MANQUE?UN?FINQUAND\);
258     PRINT(1);
259     'GOTO'FERR;
260     'END';
261     'IF'CLF(4)'THEN''GOTO'FERR;
262     NVLD:='FALSE';
263     PVLD:='FALSE';
264     'IF'NIT=0'THEN''BEGIN''INTEGER'I,J;
265     I:=FST+NDB+1;
266     RE:'IF'I>=DI'THEN''GOTO'PDST;
267     ST[I+2]:='IF'ST[I+2]=1'THEN'ST[I]'ELSE'ST[I+2]-1;
268     J:=ST[I+2]+I+2;
269     ST[J]:=ST[ST[I+1]];
270     I:=I+ST[I]+3;
271     'GOTO'RE;
272     PDST:INX:=2003;
273     'FOR'INX:=INX-3-ST[INX]'WHILE'INX>LS'DO'ST[INX-1]:=0;
274     NIT:=1;
275     'END''ELSE'NIT:=0;
276     'FOR'INX:=1'STEP'NDB+7'UNTIL'(NDB*NDPT-1)*(NDB+7)+1'DO''BEGIN'ST[INX
276     +NDB+1]:=ST[INX+NDB+2]:=0;
277     'END';
278     'FOR'IL:=DI'STEP'5'UNTIL'ADS-5'DO''BEGIN'IBP:=ST[IL];
279     INX:=(ST[IL+2]-1)*NDB*(NDB+7)+(IBP-1)*(NDB+7)+NDB+2;
280     SI:=(ST[IL+3]+ST[IL+4])/2;
281     'IF'SI>0'THEN'ST[INX]:=ST[INX]+SI'ELSE'ST[INX+1]:=ST[INX+1]-SI;
282     'END';
283     SI:=0;
284     'FOR'NOP:=1'STEP'1'UNTIL'NDPT'DO''BEGIN'INX:=(NOP-1)*NDB*(NDB+7)-NDB
284     -6;
285     'FOR'IBP:=1'STEP'1'UNTIL'NDB'DO''BEGIN'INX:=INX+NDB+7;
286     'IF'ST[INX+NDB+6]=-99999'THEN''GOTO'EXCL;
287     RJK:=ST[INX+NDB+2];
288     'IF'ST[INX+NDB+3]=3'OR'ST[INX+NDB+3]=4'THEN''GOTO'AB;
289     'FOR'INXM:=INX'STEP'1'UNTIL'INX+NDB-1'DO''BEGIN''IF'ST[INXM]<=0'THEN
289     ''GOTO'PL;
290     'IF'ST[ST[INXM]]=0'THEN''GOTO'PL;
291     IBA:=(INXM-INX)*(NDB+7)+1+(NOP-1)*(NDB+7)*NDB;
292     'IF'ST[INX+NDB+3]=1'THEN''BEGIN''INTEGER'TC;
293     'REAL'P1,P2,P3,QR;
294     INXP:=ST[INXM];
```

```
295     TC:=ST[INXP];
296     P2:=P3:=0;
297     P1:=ST[INXP-3];
298     TC:=TC-1;
299     'IF'TC>0'THEN''BEGIN'P2:=ST[INXP-4];
300     TC:=TC-1;
301     'END';
302     'IF'TC>0'THEN'P3:=ST[INXP-5];
303     QR:=MAX(ST[INX+NDB]-P2,0);
304     SI:=(P1*QR+P3)*HX;
305     'IF'NIT=0'THEN'SI:=(SI+ST[ST[INXM]-1])/2;
306     'GOTO'FINEX;
307     'END';
308     'IF'ST[INX+NDB+3]=2'THEN''BEGIN''INTEGER'L;
309     SI:=0;
310     INXP:=ST[INXM];
311     L:=ST[INXP-3];
312     L:='IF'ST[L+2]=1'THEN'L+ST[L]+2'ELSE'ST[L+2]+1+L;
313     'FOR'I:=INXP-4'STEP'-2'UNTIL'INXP-2-ST[INXP]'DO''IF'ST[I]=0'THEN'SI
313     =ST[I-1]*HX+SI;
314     'IF'NIT=0'THEN'SI:=(SI+ST[ST[INXM]-1]-ST[L])/2;
315     RJK:=RJK+SI;
316     SI:=SI+ST[L];
317     'GOTO'FINEX;
318     'END';
319     'IF'ST[INX+NDB+3]=5'THEN''BEGIN''REAL'P,QR;
320     P:=ST[ST[INXM]-3];
321     'IF'NIT=1'THEN'SI:='IF'P<0'THEN'-P'ELSE'P*HX'ELSE''BEGIN'QR:=ST[ST[I
321     NXM]-1];
322     SI:='IF'P<0'THEN'QR'ELSE'(QR+P*HX)/2;
323     'END';
324     'GOTO'FINEX;
325     'END';
326     FINEX:ST[INX+NDB+2]:=ST[INX+NDB+2]+SI;
327     ST[IBA+NDB+1]:=ST[IBA+NDB+1]+SI;
328     ST[ST[INXM]-1]:=SI;
329     SI:=0;
330     PL:'END';
331     AB:'IF'NIT=1'THEN'ST[INX+NDB+6]:=ST[INX+NDB]'ELSE''BEGIN'ST[INX+NDB]
331     :=ST[INX+NDB+6];
332     'IF'ST[INX+NDB+3]=2'AND'RJK#0'AND'ST[INX+NDB]#0'THEN''BEGIN''INTEGER
332     'J,L1;
333     'REAL'C0;
334     C0:=1-RJK/ST[INX+NDB];
335     'FOR'INXM:=INX'STEP'1'UNTIL'INX+NDB-1'DO''BEGIN''IF'ST[INXM]<=0'THEN
335     ''GOTO'KL;
336     J:=ST[ST[INXM]-3];
337     IBA:=J+ST[J]+2;
338     J:=J+2;
339     L1:=J+ST[J];
340     L1:='IF'L1=1'THEN'IBA'ELSE'L1-1;
341     'FOR'J:=J+1'WHILE'J<L1'DO'ST[J]:=C0*ST[J];
342     'FOR'J:=L1+1'STEP'1'UNTIL'IBA'DO'ST[J]:=C0*ST[J];
343     KL:'END';
344     'END';
345     'END';
346     EXCL:'END';
347     INX:=(NOP-1)*NDB*(NDB+7)-6;
```

```
348     'FOR'IBP:=1'STEP'1'UNTIL'NDB'DO''BEGIN'INX:=INX+6;
349     'IF'ST[INX+NDB+1]#5'THEN''GOTO'CS;
350     'FOR'I:=1'STEP'1'UNTIL'NDB'DO''BEGIN'INXP:=ST[INX+1];
351     'IF'INXP>0'THEN''BEGIN''IF'ST[INXP]=0'THEN''BEGIN'INXV:=INX+NDB+1;
352     SI:=ST[INXV]-ST[INXV+2];
353     'IF'SI>0'THEN''BEGIN'ST[-ST[INX+1]-1]:=SI;
354     ST[INXV+2]:=ST[INXV];
355     INX:=INXV;
356     INXV:=(NOP-1)*(NDB+7)*NDB+(I-1)*(NDB+7)+NDB+2;
357     ST[INXV]:=ST[INXV]+SI;
358     ST[INX]:=0;
359     'GOTO'CE;
360     'END';
361     'GOTO'CS;
362     'END';
363     'END';
364     'END';
365     CS:INX:=INX+NDB+1;
366     CE:'END';
367     INX:=-NDB-6+(NOP-1)*NDB*(NDB+7);
368     'FOR'IBP:=1'STEP'1'UNTIL'NDB'DO''BEGIN'INX:=INX+NDB+7;
369     'IF'ST[INX+NDB+3]=4'THEN''BEGIN''INTEGER'IP;
1370    'REAL'SP;
1371    'BOOLEAN'RST;
1372    SP:=1;
1373    RST:='FALSE';
1374    'FOR'INXM:=INX'STEP'1'UNTIL'INX+NDB-1'DO''BEGIN''REAL'PAR;
1375    'IF'ST[INXM]<=0'THEN''GOTO'NL;
1376    IBA:=INX+(NDB+7)*(INXM-IBP-INX+1);
1377    'IF'ST[ST[INXM]]<1'THEN''BEGIN'PAR:=0;
1378    RST:='TRUE';
1379    IP:=INXM;
1380    'END''ELSE'PAR:=ST[ST[INXM]-3];
1381    SP:=SP-PAR;
1382    SI:=ST[ST[INXM]-1]:=PAR*ST[INX+NDB+1];
1383    ST[IBA+NDB+1]:=ST[IBA+NDB+1]+SI;
1384    ST[INX+NDB+2]:=ST[INX+NDB+2]+SI;
1385    NL:'END';
1386    'IF'SP<0'THEN''BEGIN'TEXT("SOMME?DES?PARAMETRES?DE?L'EMBRANCHEMENT?N
1386    0.?\);
1387    EDIT("F3.0\,IBP);
1388    TEXT(",?PRODUIT?NO.?\);
1389    EDIT("F2.0\,NOP);
1390    TEXT(",?SUPERIEURE?A?1\);
1391    PRINT(1);
1392    'GOTO'FERR;
1393    'END';
1394    'IF'RST'THEN''BEGIN'INXM:=IP;
1395    IBA:=INX+(NDB+7)*(INXM-IBP-INX+1);
1396    SI:=ST[ST[INXM]-1]:=SP*ST[INX+NDB+1];
1397    ST[IBA+NDB+1]:=ST[IBA+NDB+1]+SI;
1398    ST[INX+NDB+2]:=ST[INX+NDB+2]+SI;
1399    'END';
1400    'GOTO'PE;
1401    'END';
1402    'IF'NIT=1'THEN''GOTO'PE;
1403    'IF'ST[INX+NDB+3]=2'THEN''BEGIN''INTEGER'NR;
1404    'REAL'RR,REST;
```

```
405     SI:=ST[INX+NDB+1];
406     REST:=1;
407     'FOR'INXM:=INX'STEP'1'UNTIL'INX+NDB-1'DO''BEGIN''IF'ST[INXM]<=0'THEN
407     ''GOTO'NL1;
408     INXP:=ST[INXM];
409     IBA:=ST[INXP-3];
410     INXV:=IBA+2;
411     IBA:=INXV+ST[INXV]-1;
412     NR:=INXP-2*(1+ENTIER(ST[INXP]/2));
413     'FOR'IL:=INXP-4'STEP'-2'UNTIL'NR'DO''BEGIN''INTEGER'AR;
414     'IF'ST[IL]=0'THEN''GOTO'RO;
415     AR:=ENTIER(ST[IL]/HX);
416     'IF'AR>ST[INXV-2]-1'THEN''BEGIN'TEXT("RETARD?TROP?GRAND?:?PRODUIT?:?
416     \);
417     EDIT("F2.0\,NOP);
418     TEXT("?BLOC?:?\);
419     EDIT("F2.0\,IBP);
420     TEXT("?VERS?:?\);
421     EDIT("F2.0\,INXM-INX+1);
422     PRINT(1);
423     'GOTO'FERR;
424     'END';
425     AR:=IBA-AR;
426     'IF'AR<=INXV'THEN'AR:=ST[INXV-2]+AR;
427     RR:='IF'IL-1<INXP-ST[INXP]-2'THEN'REST'ELSE'ST[IL-1];
428     REST:=REST-RR;
429     ST[AR]:=ST[AR]+SI*RR;
430     RO:'END';
431     NL1:'IF'REST<0'THEN''BEGIN'TEXT("PRODUIT?NO.?\);
432     EDIT("F2.0\,NOP);
433     TEXT(",?RETARD?NO.?\);
434     EDIT("F2.0\,IBP);
435     TEXT(",?SOMME?DES?PROPORTIONS?SUPERIEURE?A?1\);
436     PRINT(1);
437     'GOTO'FERR;
438     'END';
439     'END';
440     'END';
441     PE:'END';
442     'END';
443     INX:=-6;
444     'FOR'NOP:=1'STEP'1'UNTIL'NOPT'DO''FOR'IBP:=1'STEP'1'UNTIL'NDB'DO''BE
444     GIN'INX:=INX+NDB+7;
445     ST[INX]:=ST[INX]+ST[INX+1]-ST[INX+2];
446     'IF'ST[INX]<-@-07'AND'NIT=0'AND'ST[INX+6]#-99999'THEN''BEGIN'NMR:=NM
446     R+1;
447     TEXT("T=\);
448     EDIT("F6.2\,T);
449     SPACE(5);
450     TEXT("PRODUIT?\);
451     EDIT("F2.0\,NOP);
452     SPACE(5);
453     TEXT("BLOC?\);
454     EDIT("F2.0\,IBP);
455     SPACE(5);
456     TEXT("CONTENU?NEGATIF?\);
457     EDIT("F9.4\,ST[INX]);
458     PRINT(1);
```

```
459     'IF'NMR>10'THEN''GOTO'FERR;
460     'END';
461     'END';
462     'IF'NIT=1'THEN''BEGIN'NB1:=NB1+1;
463     T:=NB1*HX;
464     'END''ELSE''BEGIN''FOR'IL:=D1'STEP'5'UNTIL'ADS-5'DO''BEGIN'ST[IL+4]:
464     =ST[IL+3]:=ST[IL+1]:=0;
465     'END';
466     ADS:=D1;
467     'END';
468     IP:=IP1;
469     'IF'CLE(0)'THEN''GOTO'FSIM;
470     SIMULATION(TOUTES);
471     MOT:=MOT1;
472     NOS:=0;
473     'GOTO'EXPERIENCE;
474     PDEX:'END';
475     ST[0]:=-1;
476     'FOR'INX:=1'STEP'1'UNTIL'100'DO'S2[INX]:=0;
477     VARIABLE:=-99999;
478     TOUS:=0;
479     TOTALE:=0;
480     TOTAL:=0;
481     TOUTES:=0;
482     NDEX:=1;
483     NOEX:=1;
484     ASIM:ICT(FERR);
485     'FOR'I:=1'STEP'1'UNTIL'10'DO'BP[I]:='TRUE';
486     TEXT("SIMULATION?NO.\);
487     EDIT("F2.0\,NOEX);
488     PRINT(6);
489     NOE:=NOEX;
490     NAU:=1;
491     C1:='FALSE';
492     FANT:='FALSE';
493     PVLD:='FALSE';
494     NVLD:='FALSE';
495     BAT:='TRUE';
496     COR:='TRUE';
497     NPS:='TRUE';
498     C1:='TRUE';
499     AX:='TRUE';
500     CST:='FALSE';
501     MNX:='FALSE';
502     MNY:='FALSE';
503     ODX:=ODY:=QDX:=QDY:='TRUE';
504     SX:=-1;
505     SY:=-1;
506     RND:=0.99;
507     MR:=26244;
508     B1:=3215;
509     NDB:=1;
510     ADS:=D1:=10;
511     NB1:=0;
512     NOP:=1;
513     NIT:=0;
514     NTP:=-1;
515     NMR:=0;
```

```
1516    INXP:=2003;
1517    LS:=2000;
1518    PR:=-1;
1519    'FOR'INX:=1'STEP'1'UNTIL'INXP'DO'ST[INX]:=0;
1520    IP1:=-1;
1521    IP:=0;
1522    TO:=T:=-99999;
1523    TP:=-99999;
1524    TMA:=-99999;
1525    T21:=-99999;
1526    NDPT:=1;
1527    ITR:=0;
1528    PTR:=1;
1529    TRC:=0;
1530    QX:=QY:=99999;
1531    NTT:=0;
1532    TI:=1;
1533    HT:=1;
1534    NTL:=0;
1535    HX:=1;
1536    NQ:=0;
1537    MOT:=0;
1538    NOS:=0;
1539    'COMMENT'**********************DESCRIPTION.DU.MODELE*************
1539    ***********;
1539    EXPERIENCE:'COMMENT'*************DESCRIPTION.DE.L.EXPERIENCE*******
1539    ***********;
1539    EXECUTION;
1540    'COMMENT'**********************EPILOGUE..ERREURS,COURBES,SIMUL.
1540    SUIVANTE*****;
1540    FSIM:TEXT("STOP?A?:?\);
1541    EDIT("F9.4\,T);
1542    SPACE(3);
1543    TEXT("NIT?=?\);
1544    EDIT("F1.0\,NIT);
1545    SPACE(3);
1546    TEXT("MOT?NUMERO?:?\);
1547    EDIT("F3.0\,MOT);
1548    PRINT(1);
1549    'IF'CLE(0)'THEN''BEGIN'TEXT("CLE?0\);
1550    PRINT(1);
1551    'END';
1552    'IF'CLE(1)'THEN''BEGIN'TEXT("CLE?1\);
1553    PRINT(1);
1554    'END';
1555    'IF'CLE(2)'THEN''BEGIN'TEXT("CLE?2\);
1556    PRINT(1);
1557    'END';
1558    'IF'CLE(3)'THEN''BEGIN'TEXT("CLE?3\);
1559    PRINT(1);
1560    'GOTO'ISR;
1561    'END';
1562    SUNO:'IF'CLE(5)'THEN''BEGIN'TEXT("CLE?5\);
1563    PRINT(1);
1564    'END';
1565    'IF'CLE(2)'THEN''GOTO'FICO;
1566    'IF'TRC=0'THEN''GOTO'FICO;
1567    'IF'MAF=MIF'THEN''BEGIN'TEXT("COURBE???FONCTION?CONSTANTE?=?\);
```

```
568     EDIT("L18.12\,MAF);
569     PRINT(1);
570     TEXT("VARIABLE?:?\);
571     EDIT("L18.12\,MIV);
572     TEXT("??A?\);
573     EDIT("L18.12\,MAV);
574     PRINT(1);
575     'GOTO'FIC0;
576     'END';
577     'IF'MAV=MIV'THEN''BEGIN'TEXT("COURBE??PAS?DE?PROGRESSION?DE?LA?VARIA
577     BLE?=?\);
578     EDIT("L18.12\,MAV);
579     PRINT(1);
580     'GOTO'FIC0;
581     'END';
582     'FOR'I:=TRC+1'STEP'1'UNTIL'402'DO''BEGIN'TAC[2,I]:=TAC[2,I-1];
583     TAC[1,I]:=TAC[1,I-1];
584     'END';
585     ICT(INC0);
586     'IF'CI'THEN''BEGIN'PRINT(-8);
587     TEXT("TRACE?SUR?IMPRIMANTE\);
588     PRINT(1);
589     COURBE(TAC,50);
590     'GOTO'FIC0;
591     'END';
592     NEWCURVE;
593     'IF'SX<0'THEN'SX:=ABS(MAV-MIV)/25;
594     'IF'SY<0'THEN'SY:=ABS(MAF-MIF)/19;
595     SCALEX(SX);
596     SCALEY(SY);
597     'IF'MNX'THEN'MINX(MIX);
598     'IF'MNY'THEN'MINY(MIY);
599     'IF'ODX'THEN'OX:=MIV;
600     'IF'ODY'THEN'OY:=MIF;
601     'IF'QDX'THEN'QX:=5*SX;
602     'IF'QDY'THEN'QY:=5*SY;
603     TEXT("TRACE?SUR?TABLE\);
604     PRINT(1);
605     PLOTCURVE(TAC,200);
606     'IF'AX'THEN'PLOTAX(OX,QX,"VARIABLE\,"F13.4\,OY,QY,"FONCTION\,"F10.4\
606     ,2);
607     ENDLIN;
608     FIC0:ICT(FERR);
609     'IF'NOEX#NDEX'THEN''BEGIN'NOEX:=NOEX+1;
610     'GOTO'ASIM;
611     'END';
612     'GOTO'FRE;
613     INC0:TEXT("INCIDENT?COURBE\);
614     PRINT(1);
615     'IF'CI'THEN'TEXT("IMPRIMANTE\)'ELSE'TEXT("TABLE\);
616     PRINT(1);
617     TEXT("SX?:?\);
618     EDIT("L18.12\,SX);
619     PRINT(1);
620     TEXT("SY?:?\);
621     EDIT("L18.12\,SY);
622     PRINT(1);
623     TEXT("OX?:?\);
```

```
24    EDIT("LI8.I2\,OX);
25    PRINT(1);
26    TEXT("OY?:?\);
27    EDIT("LI8.I2\,OY);
28    PRINT(1);
29    TEXT("QX?:?\);
30    EDIT("LI8.I2\,QX);
31    PRINT(1);
32    TEXT("QY?:?\);
33    EDIT("LI8.I2\,QY);
34    PRINT(1);
35    TEXT("MIX?:?\);
36    EDIT("LI8.I2\,MIX);
37    PRINT(1);
38    TEXT("MIY?:?\);
39    EDIT("LI8.I2\,MIY);
40    PRINT(1);
41    'FOR'I:=1'STEP'1'UNTIL'TRC'DO''BEGIN''IF''NOT'CLE(2)'THEN''BEGIN'EDI
41    T("LI8.I2\,TAC[1,1]);
42    SPACE(3);
43    EDIT("LI8.I2\,TAC[2,1]);
44    PRINT(1);
45    'END';
46    'END';
47    FERR:PRINT(-8);
48    TEXT("ERREUR?T?=?\);
49    EDIT("F9.4\,T);
50    SPACE(3);
51    TEXT("NIT?=?\);
52    EDIT("FI.0\,NIT);
53    SPACE(3);
54    TEXT("MOT?NUMERO?:?\);
55    EDIT("F3.0\,MOT);
56    PRINT(1);
57    'IF'CLE(4)'THEN''BEGIN'TEXT("CLE?4\);
58    PRINT(1);
59    'END';
60    ISR:'IF'CLE(5)'THEN''BEGIN'TEXT("CLE?5\);
61    PRINT(1);
62    'GOTO'PIM;
63    'END';
64    TEXT("NB.?PRODUITS?:?\);
65    EDIT("F2.0\,NDPT);
66    TEXT("???NB.?BLOCS?:?\);
67    EDIT("F2.0\,NDB);
68    TEXT("???PAS?:?\);
69    EDIT("LI0.5\,HX);
70    PRINT(1);
71    TEXT("VALEUR?DES?INDEX\);
72    PRINT(1);
73    TEXT("INX?=?\);
74    EDIT("F5.0\,INX);
75    TEXT(",?\);
76    TEXT("INXM?=?\);
77    EDIT("F5.0\,INXM);
78    TEXT(",?\);
79    TEXT("INXP?=?\);
80    EDIT("F5.0\,INXP);
```

```
1681      TEXT(",?\);
1682      TEXT("NOP?=?\);
1683      EDIT("F5.0\,NOP);
1684      TEXT(",?\);
1685      TEXT("NOB?=?\);
1686      EDIT("F5.0\,NOB);
1687      TEXT(",?\);
1688      TEXT("NOBA?=?\);
1689      EDIT("F5.0\,NOBA);
1690      TEXT(",?\);
1691      TEXT("NOS?=?\);
1692      EDIT("F5.0\,NOS);
1693      TEXT(",?\);
1694      TEXT("IP?=?\);
1695      EDIT("F5.0\,IP);
1696      TEXT(",?\);
1697      TEXT("TRC?=?\);
1698      EDIT("F5.0\,TRC);
1699      PRINT(1);
1700      TEXT("NVLD?EST?\);
1701      'IF'NVLD'THEN'TEXT("VRAI\)'ELSE'TEXT("FAUX\);
1702      SPACE(3);
1703      TEXT("PVLD?EST?\);
1704      'IF'PVLD'THEN'TEXT("VRAI\)'ELSE'TEXT("FAUX\);
1705      SPACE(3);
1706      TEXT("COR?EST?\);
1707      'IF'COR'THEN'TEXT("VRAI\)'ELSE'TEXT("FAUX\);
1708      PRINT(1);
1709      TEXT("TMA?=?\);
1710      EDIT("L18.12\,TMA);
1711      PRINT(1);
1712      NOB:=FST+NDB;
1713      FST:=8;
1714      BASCSO;
1715      PRINT(4);
1716      TEXT("IMPLANTATION\);
1717      PRINT(3);
1718      IMPLAN("STRUCTURE\,1,NOB);
1719      TEXT("//\);
1720      EDIT("F4.0\,NOB);
1721      TEXT("//\);
1722      PRINT(2);
1723      IMPLAN("STOCKAGE\,NOB+1,DI=1);
1724      TEXT("//\);
1725      EDIT("F4.0\,DI-1);
1726      TEXT("//\);
1727      PRINT(2);
1728      IMPLAN("STRESS\,DI,ADS-1);
1729      TEXT("VIDE?:?\);
1730      EDIT("F6.0\,ADS);
1731      TEXT("?A?\);
1732      EDIT("F6.0\,LS);
1733      PRINT(4);
1734      IMPLAN("STIPULATION\,LS+1,2000);
1735      PIM:'IF'CLE(3)'THEN''GOTO'SUNO;
1736      FRE:'END'
```

FIN DE COMPILATION

BIBLIOGRAPHIE

[1] - ACKERMAN E, ROSEWAR J.W, Mc GUCKIN W.F (1964) :
A mathematical model of the glucose tolerance test.
Phys. Med. Biol. 9, 203-213.

[2] - AGARD J. et coll. (1968) :
Les méthodes de simulation.
Dunod Paris.

[3] - ALFONSECA M. (1971) :
"SIAL/70", un languaje de simulacion analogico-digital.
Rev. de Telecommunicacion, 103.

[4] - BALLARD B.E, GOYAN J.E. (1966) :
Application of analog computer techniques to in-vivo drugs kinetic studies.
Med. Biol. Engineering 5, 483-490.

[5] - BEAUGAS G, COMYN G, PAGES J.C, PARMENTIER P. (1969) :
Simulation de processus physiologiques sur ordinateur.
Rev. Informatique Médicale 0, 15-25.

[6] - BEAUGAS G (1970) :
L'identification des systèmes dynamiques.
Etude I.B.M. n° 100.

[7] - BELLMAN R (1958) :
Dynamic programming and stochastic control processes.
Information and Control 1, 228-239.

[8] - BEKEY G.A (1970) :
System identification : an introduction and a survey.
Simulation oct. 70, 151-166.

[9] - BINET P, BEAUGAS G, LESIEUR, PAGES J.C, PERRIN P. (1968) :
Simulation des systèmes biologiques sur ordinateur par le langage C.S.M.P/1130.
Application à la splénoportographie isotopique.
Brochure I.B.M. Etude n° FF2-0064.

[10] - BUXTON J.N, LASKI J.G. (1963) :
Control and simulation language.
Computer Journal 5, p 194.

[11] - BRENNAN R.D, SILBERBERG M.Y. (1968) :
The system 360 continuous system modeling program.
Simulation 11, 6, 301-308.

[12] - CHANCE B, CHANCE E.M, SELLERS P.H. (1960) :
Analog and digital representations of enzyme kinetics.

[13] - COMYN G. (1970) :
Simulation de processus biologiques sur ordinateur.
Thèse de 3e cycle. Faculté des Sciences de Lille.

[14] - CROSBIE R.J, HARDY J.D, FESSENDE E. (1962) :
Electrical analog simulation of temperature regulation in man.
IRE trans. Bio. Med. Electron. 89, 245-252

[15] - DUCROCQ P.M, SULMAN C, SWYNGEDAUW J. (1970) :
Simulation sur ordinateur du Nephrogramme isotopique normal et pathologique.
Journées d'Inf. Med. 1970, 57-81.

[16] - DUCROCQ P.M. (1972) :
Un langage pour la simulation de certains processus physiologiques,
SIPHALGOL.
Journées d'Inf. Med. 1972, 183-200.

[17] - DUVELLEROY M, DURANTEAU A, FOURQUET M. (1966) :
Simulation sur calculateur analogique de l'équilibre acido-basique.
Anesth. Anal. Reanim. 23, 437-458.

[18] - DUVELLEROY M. (1970) :
Simulation des phénomènes physiologiques, 1ère et 2ème partie.
Rev. d'Inf. Méd. 3, 213-231, 4, 326-342.

[19] - EHRMANN R. (1970) :
Les langages de simulation.
Dunod Paris

[20] - EYKHOFF (1963) :
Some fundamental aspects of process-parameter estimation.
IEEE Trans. on Autom. Control Oct 63, 347-357.

[21] - FOIRET J.C, FURON D, MILBLED G. (1972) :
Modèle de stato-régulation.
Lille Medical 17, 9, 1230-1236.

[22] - GRODINS F.S, GRAY J.S, SCHROEDER K.R, NORINS A.L, JONESR W. (1954) :
Respiratory response to CO_2 inhalation. A theoretical study of a non linear biological regulator.
J. Appl. Physiology 7, 283-308.

[23] - GRODINS F.S (1963) :
Control theory and biological systems.
Columbia university Press. New York.

[24] - GRODINS F.S, JAMES G. (1963) :
Mathematical models of respiratory regulation.
Ann. N.Y. Acad. Sci. 109, 852-868.

[25] - GRODINS F.S, BUELL J, BART A.J. (1967) :
Mathematical analysis and digital simulation of the respiratory control system.
J. Appl. Physiol. 22, 260-276.

[26] - GRONER G.P, CLARK R.L, BERMAN R.A, DELAND E.C. (1971) :
BIOMOD : an interactive computer graphics system for modeling.
AFIPS conf. Proc. 39, 369-378.

[27] - GUYTON A.C, MILHORN H.T, COLEMAN T.G. (1967) :
Simulation of physiological mechanisms Part 1 et 2.
Simulation 9, 1, 15-20, 9, 3, 73-79.

[28] - HACHE J.C, FRANCOIS P, LIMOSIN J.J, TOULOTTE J.M. (1973) :
Méthode et intérêt de la simulation des affections maculaires sur ordinateur.
Journées d'Inf. Méd. 1973, 243-255.

[29] - HO Y.C, LEE R.C.K. (1965) :
Identification of linear dynamic systems.
Information and control 8, 93-110.

[30] - HOLMES J.K. (1969) :
Two stochastics approximation for identify linear systems.
IEEE Trans. on Autom. Control. Juin 69, 292-295.

[31] - KIEFER J, WOLFOWITZ J. (1952) :
Stochastic estimation of the maximum of a regression function.
Annals. of Math. Stat. 23, 462-466.

[32] - KIEN G.A, KOUSHANPOUR E. (1968) :
Digital computer simulation of the Nephron : the action of osmotic diuretics.
J. Pharmacol. exp. Therapeutics 163, 1, 198-209.

[33] - LEDLEY R.S. (1965) :
Use of computers in Biology and Medecine.
Mc Graw Hill Book company. New York.

[34] - LINCOLN T.L, HARRIS P.A, GROSS J.F. (1973) :
BIOMOD simulation of pharmacokinetics for leukemia chemotherapy.
Journée d'Inf. Med. 1973, 229-242.

[35] - MARTIN J. MONOT C. (1967) :
Analyse d'un modèle mathématique du fonctionnement rénal. Ajustement numérique sur ordinateur des paramètres à partir des courbes du nephrogramme isotopique à l'Hippuran.
Congrès international d'Urologie. Liège.

[36] - MONOT C, MARTIN J, NAJEAN Y, DRESCH C. (1969) :
Problèmes mathématiques posés par la simulation de la cinétique du fer.
Journées d'Inf. Méd. 1969, 219-240.

[37] - MONOT C, MARTIN J, NAJEAN Y, DRESCH C, FAILLE A.(1972) :
Simulation de la cinétique du fer radioactif dans les cas pathologiques.
Journées d'Inf. Méd. 1972, 183-200.

[38] - NEDEY R, TRICHET B, DUVELLEROY M, FOURQUET M, PLOTON Y, APTER H. (1967) :
Simulation de l'équilibre acido-basique de l'organisme.
5th International Congress of AICA. Lausanne 213-215.

[39] - PAGES J.C. (1965) :
Utilisation d'un ordinateur en pédagogie médicale.
Etude IBM n° 4.

[40] - PARMENTIER P, LESTRADET H, PAGES J.C. (1970) :
Expérimentation par les méthodes de simulation d'un nouveau modèle de la régulation glyco-énergétique.
Rev. Inf. Méd. 1,4.

[41] - REEVE, GUYTON. (1967) :
Physical basis of circulatory transport regulation and exchange.
W.B. Sanders ed. Philadelphia, Londres.

[42] - RICHALET. (1965) :
Etude prospective des principes de la contraction musculaire.
Contrat DRME n° 38-864.

[43] - ROBBINS H, MONRO S. (1951) :
A stochastic approximation method.
Annals of Math. Stat. 22, 400-407.

[44] - SHEPPARD C.W. (1962) :
Basic principles of tracers method.
Wiley Sons. Inc. Pub. New-York.

[45] - STARK L, IDLA M, WILLIS P.A. (1961) :
Dynamic characteristics of the motor coordination system in man.
Biophys. J. 6, 539-551.

[46] - STRAUSS J.L. (1967) :
The SCI continuous systems simulation language.
Simulation 9 n° 6, 281-303.

[47] - SWYNGEDAUW J, SULMAN C, DUCROCQ P.M. (1970) :
Simulation du nephrogramme isotopique.
Journées de simulation IBM.

[48] - SWYNGEDAUW J, SULMAN C. (1967) :
Decomposition analytique du nephrogramme. Stase pathologique de l'Hippuran marqué dans le parenchyme rénal.
Lille.

[49] - SWYNGEDAUW J, SULMAN C, DUCROCQ P.M. (1970) :
Modèle de secrétion rénale de l'Hippuran 131 - Simulation sur ordinateur.
Lille Médical 15, 4, 608-621.

[50] - TOCHER K.D, (1963) :
The art of simulation.
English universities Press. London.

[51] - TOCHER K.D. (1964) :
Review of simulation languages.
Operational Research Quaterly 16, 2, 189-216.

[52] - VALLEE G, BELIN J.P, BEAUGAS G, GOIOT L, PAGES J.C. (1968) :
Programme approchant par une courbe pluriexponentielle des données expérimentales discrètes obtenues chez l'homme par détection radioactive.
Etude IBM n° FF2-0056-0.

[53] - VENTER J.H. (1967) :
An extension of the Robbins-Monro procedure.
Ann. Math. Stat. 38, 181-190.

[54] - WOLFOWITZ J. (1952) :
On the stochastic approximation method of Robbins and Monro.
Annals of Math. Stat. 23, 457-461.