N° d'ordre : 393

THÈSE

présentée à l'

UNIVERSITÉ DES SCIENCES ET TECHNIQUES DE LILLE 1

pour l'obtention du titre de

DOCTEUR 3me CYCLE

par

Bertrand du BOIS de MEYRIGNAC

ÉTUDE ET PROGRAMMATION D'UN COUPLAGE ENTRE UNE CALCULATRICE HYBRIDE ET UN CALCULATEUR DE PROCESSUS

soutenue le 6 Juillet 1973 devant la Commission d'examen

Messieurs : P. VIDAL, Président
L. POVY, Rapporteur
J. M. TOULOTTE, Examinateur
CORDONNIER, Examinateur
M. MORIAMEZ, Invité
C. SOURISSE, Invité

AVANT PROPOS

Le travail présenté dans ce mémoire a été effectué au Laboratoire d'Automatique de l'Université des Sciences et Techniques de Lille I.

Nous adressons notre plus grande reconnaissance à Monsieur le Professeur VIDAL, qui nous a accueillis au sein du Laboratoire. Qu'il trouve ici l'expression de notre gratitude pour l'honneur qu'il nous fait d'accepter la présidence de notre jury.

Nous sommes également très honorés par la présence de Monsieur le Professeur MORIAMEZ et nous lui exprimons notre plus profonde gratitude pour le soutien qu'il nous a toujours apporté.

Que Monsieur POVY, Maître de conférences à la Faculté des Sciences de Lille I, trouve ici l'expression de nos plus vifs remerciements pour nous avoir suivis et guidés durant toute cette étude avec intérêt et sympathie.

Nous sommes très reconnaissants et remercions vivement Monsieur TOULOTTE, Maître de conférences à la Faculté des Sciences de Lille, pour les conseils qu'il nous a donnés tout au long de nos travaux.

Nous remercions très vivement Monsieur CORDONNIER Maître de conférences à la Faculté des Sciences de Lille, pour l'intérêt qu'il a bien voulu porté à nos travaux en acceptant de participer à notre jury de thèse.

Monsieur SOURISSE, Responsable du département de calcul hybride à la Télémécanique Electrique, nous a grandement honorés en acceptant de participer à notre jury. Qu'il trouve ici l'expression de nos sincères remerciements.

Enfin, nous tenons à rendre hommage à l'esprit d'équipe qui règne dans le Laboratoire où nous avons travaillé et nous adressons à chacun nos plus vifs remerciements.

AVERTISSEMENT

Le travail présenté dans ce mémoire est le fruit des efforts d'une équipe universitaire du Centre d'Automatique de l'Université des Sciences et Techniques de Lille I.

Dans une première partie, nous rappelons les principes de base permettant de définir les principales caractéristiques d'un couplage entre une machine analogique et un calculateur numérique. Ces différents concepts déterminent la structure générale d'un ensemble hybride de type II.

Dans la deuxième partie, nous décrivons sous un aspect technologique la composition des deux machines et de l'interface ; celui-ci conditionne la nature des programmes de traitement des échanges entre les deux calculatrices, et impose leur mode de fonctionnement.

Dans une troisième partie, nous décrivons plus spécifiquement le langage de programmation propre au couplage. Cet ensemble est géré par un moniteur temps réel et permet le dialogue avec l'opérateur. Il comprend un système de gestion facilitant le traitement des échanges à partir d'un langage évolué.

Enfin, dans une dernière partie, des exemples illustrent diverses applications du système. Les exemples cherchent surtout à montrer les possibilités, et la précision que peut atteindre l'ensemble hybride.

L'aspect technologique n'est pas développé dans ce mémoire, cette partie a été étudiée plus particulièrement par Monsieur TARTE.

La réalisation du système a constamment nécessité des mises au point entre la partie programmation et la partie technologique, aussi est-il difficile de dissocier les travaux de Messieurs TARTE et du BOIS de MEYRIGNAC.

GENERALITES

Devant les problèmes de plus en plus complexes qui se posent, nous nous apercevons de notre inaptitude au maniement rapide des nombres, aussi a-t-on été amené à construire des outils de calcul de plus en plus perfectionnés, généralement appelés: "machines mathématiques".

D'un point de vue général, on peut classer celles-ci en deux catégories selon leur mode de fonctionnement. La première englobe les machines à traitement continu et parallèle des informations : ce sont les machines analogiques. La seconde concerne les machines à traitement séquentiel d'informations discrètes : ce sont les machines numériques.

Les nombreux opérateurs disponibles sur les calculatrices analogiques permettent une très grande souplesse lors de l'étude d'un problème physique. Ainsi ces calculatrices sont-elles particulièrement adaptées à la résolution de problèmes différentiels et integro différentiels. Une analyse de ces machines indique plusieurs caractéristiques essentielles : d'une part la technique du calcul parallèle impose un mode de fonctionnement uniforme à toute la machine ; d'autre part certains éléments non linéaires sont difficiles à réaliser tandis que la mise en mémoire de l'information pose des problèmes délicats.

Ouant aux ordinateurs, ils réalisent à grande vitesse des calculs numériques complexes, les opérations se font en série et les grandeurs mises en oeuvre apparaissent sous forme quantifiée. Ce type de machine impose donc une description mathématique complète du phénomène à étudier et sa réduction aux opérations élémentaires. Parmi les avantages, nous pouvons citer : une grande précision, liée au temps de calcul, une mise en mémoire très facile de l'information, un rôle de gestion puissant commandé par un programme logique. Parmi les inconvénients,figurent la nécessité de discrétiser l'opération d'intégration et une méthode en série.

Finalement l'ordinateur et le calculateur analogique apparaissent essentiellement complémentaires et l'idée de les associer afin de bénéficier des avantages de l'un et de l'autre est toute naturelle.

I PARTIE

DEFINITION DU CALCUL HYBRIDE

On peut définir le calcul hybride comme "l'art et la manière" d'utiliser simultanément un calculateur analogique et un ordinateur liés entre eux par un sous ensemble nommé interface.

Selon les fonctions dévolues au calculateur numérique on distingue le calcul hybride série et parallèle. Par le premier cas les deux calculateurs travaillent l'un après l'autre. Généralement le calculateur analogique résoud le système d'équations, au cours d'un cycle de calcul, alors que le calculateur numérique traite des données ou des résultats afin de préparer un nouveau cycle de calcul analogique.

Dans le calcul hybride parallèle les deux calculateurs travaillent simultanément. Au cours d'un même cycle de calcul, il y a partage des tâches entre les deux calculateurs et donc échanges d'informations (ce mode de fonctionnement parallèle n'excluant évidemment pas le précédent).

Initialement on vit apparaître des ensembles hybrides formés de machines analogiques couplées à des calculateurs numériques de caractéristiques variées, puis des ensembles standarts, étudiés et mis au point par une même firme (EAI 690), si bien qu'il est possible de distinguer trois étapes :

- couplage d'un calculateur analogique et d'un calculateur numérique,

- réalisation d'ensembles hybrides avec software limité,

- ensemble hybride avec software évolué.

Le but du travail que nous nous sommes fixé est d'arriver à ce troisième stade à partir de deux machines apparemment disparates ; les deux calculateurs n'étant pas fabriqués par la même société.

I - STRUCTURE D'UN CALCULATEUR HYBRIDE

Nous décrivons ici les deux machines d'une manière générale, la description des éléments spécifiques au couplage sera détaillée par la suite.

I.1 - Le Calculateur analogique

I.1.1 - Description

Un calculateur analogique peut être divisé en cinq parties :

- les éléments analogiques de calcul parallèle (bloc opérationnel)
- les éléments de logiques (simulateur logique)
- les éléments de liaison analogique - logique (interface logique)
- le système de contrôle et de commande
- les appareils de sortie, lents et rapides

Toutes les entrées et sorties des éléments analogiques et logiques ainsi que les commandes sont rassemblées sur deux panneaux de cablage. D'une part le panneau de cablage analogique comprend les entrées et sorties des éléments analogiques, les entrées analogiques de l'interface logique et les entrées des appareils d'enregistrements.

D'autre part, le panneau de cablage logique rassemble les entrées logiques de l'interface logique et la commande des appareils d'enregistrements.

Les éléments logiques permettent de réaliser soit des circuits combinatoires, soit des circuits séquentiels synchrones. Le système de contrôle et de commande comprend le réglage des potentiomètres, la sélection et la lecture sur voltmètre numérique des valeurs de sortie des blocs opérationnels, la commande des modes analogiques et logiques, la visualisation de l'état de chaque élément logique, la commande de l'horloge de synchronisation, etc... (fig. I.1)

I.1.2 - Caractéristiques fondamentales

Le calculateur analogique est composé de blocs opérateurs élémentaires, assemblés par l'intermédiaire d'un panneau de cablage, afin de simuler un modèle mathématique d'un ensemble physique. Les grandeurs mises en jeu par le calculateur sont des variables dépendant du temps et les opérations possibles sur le calculateur sont l'addition, l'intégration, la multiplication, etc...

Il en résulte que le modèle mathématique doit lui aussi ne dépendre que d'une variable indépendant à support positif. Si cette condition n'est pas remplie, il sera nécessaire de faire des transformations ou des approximations préalables pour s'y ramener quand de telles transformations existent.

Les différentes opérations s'effectuent simultanément et instantanément du fait de son fonctionnement en parallèle. Il en résulte que le temps nécessaire à la résolution d'un problème est indépendant du nombre d'opérations à simuler. Cette qualité n'intervient que sur le nombre de blocs opérateurs à mettre en oeuvre, et caractérise par ailleurs la puissance du calculateur.

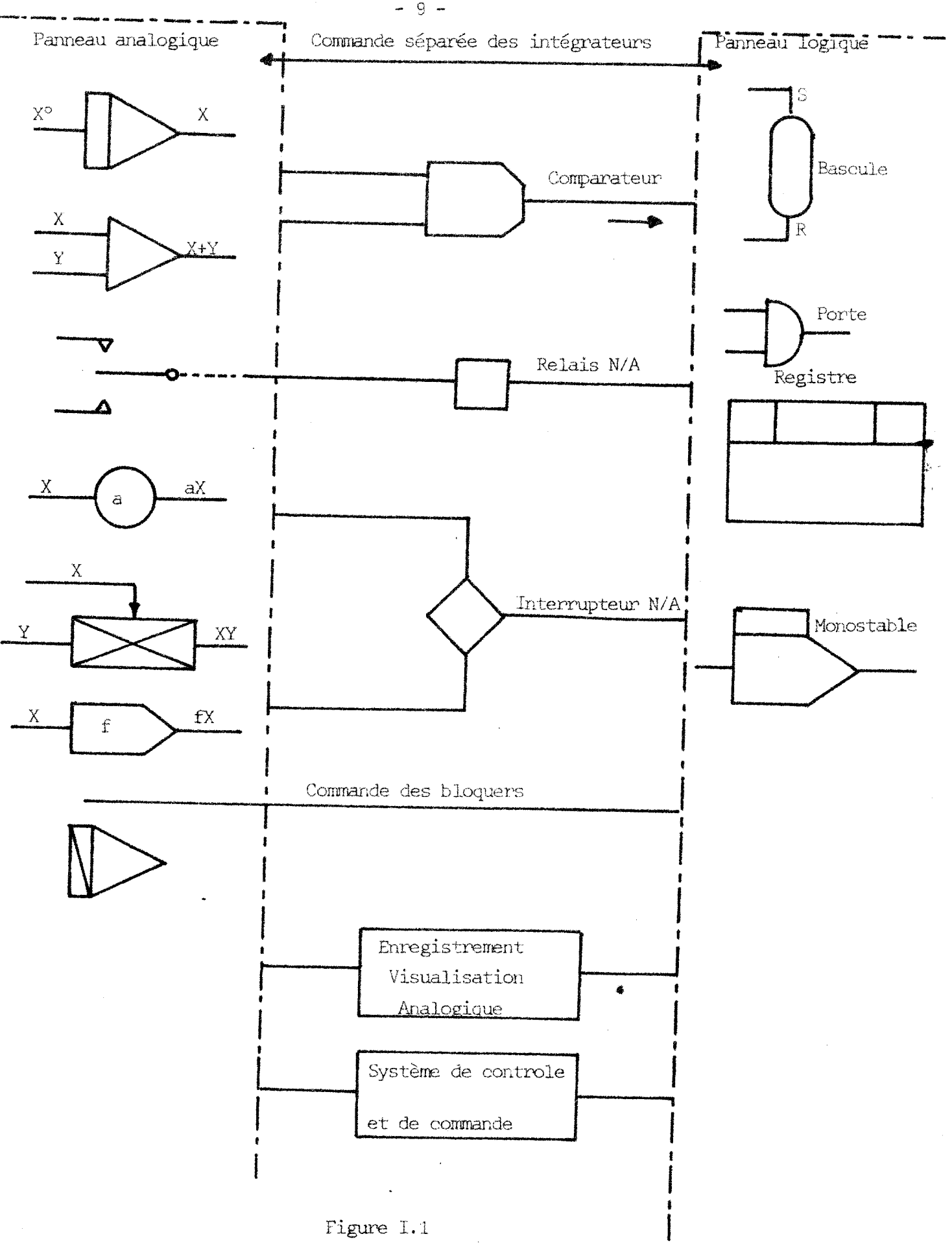


Figure I.1

Organisation générale d'une machine analogique

Nous entendons par temps nécessaire à la résolution du problème, le temps fixé à l'avance, pendant lequel la machine partant des conditions initiales déterminées, fournira l'évolution dynamique des grandeurs physiques ; plus brièvement c'est la durée d'un essai. Nous reviendrons ultérieurement sur cette machine, par comparaison avec le calculateur numérique.

La machine analogique fournit une représentation continue du phénomène étudié, c'est-à-dire qu'au cours d'un essai, chaque tension électrique représentative d'un variable du système peut être notée X (t), le temps t variant de façon continue de 0 à T (fin de l'essai). Cette caractéristique permet de dire que le calcul analogique pur fournit une représentation plus "hette" des phénomènes. Il est en effet possible de définir l'état du système simulé à tout moment, ce que ne permet pas le calcul numérique.

Les blocs opérateurs ne réalisent pas des opérations parfaites. Celles-ci sont limitées par les performances électriques, mécaniques, etc... des éléments.

D'une façon générale, le calcul analogique permet d'aborder l'étude des phénomènes physiques dont le formalisme physique se traduit par un système d'équations. Le calculateur dispose d'organes de commande, de lecture, d'enregistrements qui en font un outil de travail puissant. La conversation avec la machine est relativement aisée grâce à la grande souplesse des organes d'entrée et de sortie d'informations.

I.2 - Le calculateur numérique

I.2.1 - Description sommaire

En vue d'être intégré dans un sensemble hybride, le calculateur numérique doit se caractériser par une grande

vitesse de calcul, un répertoire très complet d'instructions, une largeur de mots permettant une précision suffisante afin de dialoguer avec la machine analogique, une grande souplesse dans les échanges d'entrée-sortie, et être doté d'un système d'interruption efficace.

Nous utilisons pour cela un calculateur de processus, qui permet de par sa structure la gestion d'un grand nombre de périphériques différents (Fig. I.2)

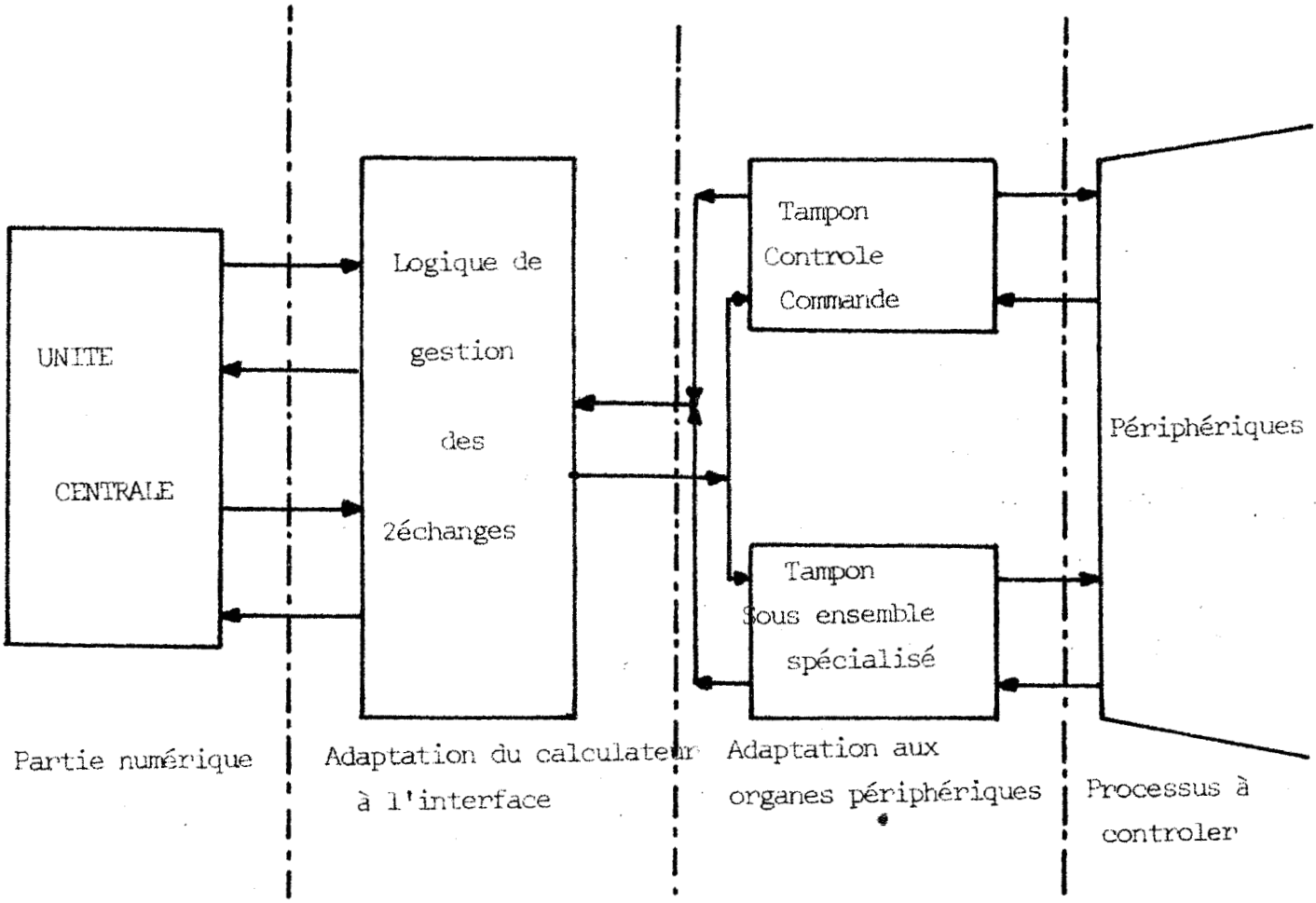


Figure I.2

Les différentes parties d'un ensemble hybride

Quant aux différentes instructions de l'ordinateur, elles permettent d'effectuer les opérations usuelles, aussi bien arithmétiques que logiques. D'autre part, le calculateur dispose d'un canal de transfert automatique qui permet de minimiser le temps d'occupation de l'unité centrale en accroissant la rapidité des échanges. Au niveau de l'adaptation du calculateur avec les différents organes périphériques, il existe une mémorisation des informations données ou reçues par le périphérique, l'adaptation des signaux (format, nature et niveau) est réalisée par l'intermédiaire d'un coupleur. La principale caractéristique de ce dernier est d'être dotée d'une certaine autonomie de fonctionnement, vis à vis de la fonction "adaptation à l'interface unité centrale".

I.2.2 - Caractéristiques

La machine utilisée est un ordinateur de processus, car sa structure permet aussi bien de traiter des problèmes scientifiques que de controler un procédé, en relevant périodiquement l'état du système à l'aide de capteurs. Il permet en outre d'allier une grande précisinn dans les contrôles, a sa rapidité des décisions dans les tests, afin de déterminer les marges de fonctionnement d'un système physique. Afin de diversifier le mode de commande du processus, il est possible de hiérarchiser les interruptions, permettant de connaître l'état actuel du processus (délais, anomalies, changement de consigne...).

Le calculateur est caractérisé par une possibilité de mémorisation importante, que ce soit en utilisant la mémoire rapide, ou en stockant les informations sur les éléments périphériques tels que disques, rubans ou bandes. Cependant, afin d'éviter des pertes de temps, et de compromettre un fonctionnement en temps réel, il est nécessaire de prévoir une mémoire rapide assez importante, permettant

d'éviter un trop grand nombre de transferts de données du calculateur, vers la mémoire lente, ou vice verça.

La précision des calculs dépend en partie de la structure interne de la machine. De nombreux ordinateurs traitent des mots de 16 bits, d'autres, comme l'appareil que nous utilisons utilisent un format de 19 bits, ce qui permet de meilleurs résultats dans les calculs en simple précision.

La machine disposant d'un compilateur Fortran IV, la mise au point des programmes se trouvent simplifiée pour l'opérateur.

En conclusion, nous pouvons dire que le rôle de l'ordinateur dans un système hybride est difficile à définir, car, la plupart du temps les techniques de calcul employées sont fonction du type de problème à résoudre. Toutefois, deux grandes options se dégagent. La première utilise l'énorme pouvoir de gestion des calculateurs numériques, la seconde exploite la facilité de mise en mémoire des informations (table de valeurs numériques par exemple). De plus, la partie numérique pourra effectuer certaines opérations arithmétiques ou contribuera à la réalisation de routines spécifiques à certaines tâches.

I.3 - L'interface

I.3.1 - Constitution

Trois fonctions principales sont confiées à cet organe de liaison :

- d'une part la conversion des tensions analogiques issues du milieu extérieur en données compatibles avec la structure du calculateur numérique, est réalisé à grande vitesse au moyen 'une chaine de mesure comprenant

un multiplexeur analogique reliant séquentiellement un convertisseur analogique numérique aux canaux transmettant les tensions analogiques, sous contrôle du calculateur numérique. Des bloqueurs sont incorporés sur le trajet de l'information analogique, avant le multiplexeur , pour échantillonner au même instant les valeurs analogiques contenues sur chacun des canaux.

- d'autre part, la conversion des informations numériques ou mots délivrées par le calculateur numérique en valeurs analogiques ; cette opération est effectuée au moyen de convertisseurs numériques analogiques.

- enfin la transmission et l'aiguillage de tous les ordre échangés par les machines lors du déroulement des différentes étapes de résolution d'un problème.

C'est au niveau de l'interface que se font tous les tests d'occupation des périphériques nécessaires au couplage, ainsi que la synchronisation entre les deux machines (fig. I.3)

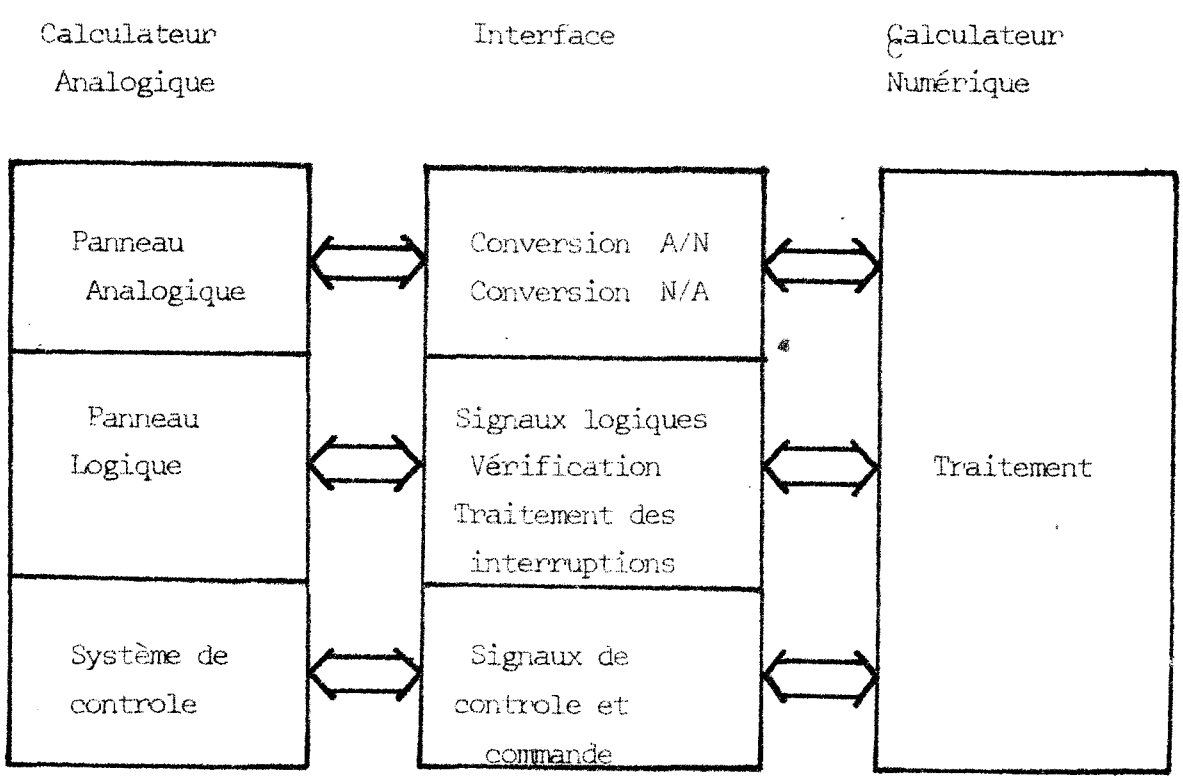


Figure I.3 : Role de l'interface

Un autre rôle de l'interface est de permettre à l'opérateur de contrôler l'ensemble du problème et le modèle réalisé . L'interface doit aussi pouvoir réaliser directement toutes lesfonctions requises par vue d'une autoMatisation de la sélection des organes analogiques, de la lecture des grandeurs de sortie, de l'adressage et de l'affichage des potentiomètres et de la commande du mode opératoire analogique.

L'interface permet aussi de fournir divers états du calcul à la mémoire de l'ordinateur par l'intermédiaire de lignes d'interruptions ou de tests. Dans ce but, la lecture de registres spécialisés permet de connaître l'état du système.

L'interface de commande s'adressant à des unités qui possèdent des temps de réponse différents doit pouvoir être programmée indépendamment de l'ensemble ce qui soulève le problème du "software". En effet un fonctionnement en temps réel nécessite une synghronisation des tâches au niveau du calculateur, et une structure précise dans la modalité des échanges.

I.3.2 - Conclusion

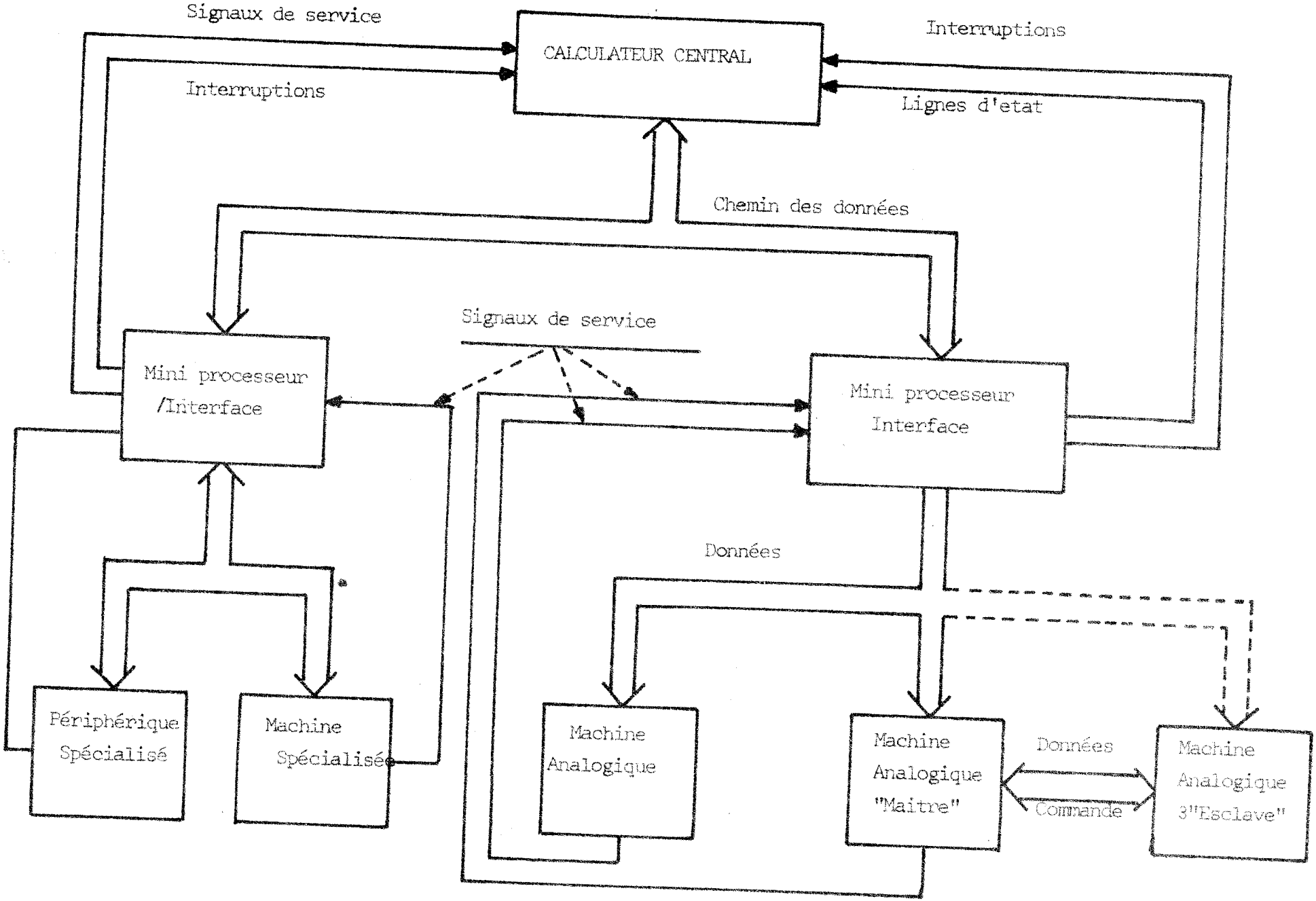
Toutes les possibilités d'un ensemble hybride dependent etroitement de la conception de l'interface qui doit alléger la tâche de l'opérateur mais également lui permettre des interventions rapides et faciles au cours du calcul. A ce niveau se place le problème du langage hybride : dans sa version la plus élaborée, l'interface tend à supprimer le travail de vérification de l'opérateur et du moniteur. La structure des machines utilisées étant fixée une fois pour toute, les vérifications à faire lors des échanges se présentent toujours sous le même format pour les mêmes fonctions ; le système étant alors figé, les qualités d'adaptation par software des programmes d'échanges ne sont

plus utiles, il est alors possible de repousser les vérifications au niveau de l'interface par l'utilisation de la microprogrammation. En allant plus loin, il sera possible de considérer l'interface comme un "minicalculateur" hautement spécialisé, traitant les échanges dont l'ordre et les données initiales seront fournies par le calculateur numérique central. Il sera alors possible à ce dernier, puisque son taux d'occupation diminue, de se consacrer à d'autres travaux qui lui sont plus spécifiques. Du fait que l'ordinateur central ne lance que les échanges, il lui est possible de commander un grand nombre de machines qui lui sont couplées, qu'elles soient analogiques ou de toute autre nature, la fin des échanges étant signalée par une interruption sur un niveau préétabli (fig. I.4). Le calculateur central reconnait alors l'origine de l'interruption, et teste si l'échange s'est bien déroulé pour valider des informations émises ou reçues, vis à vis de l'utilisateur.

II - FONCTION DES ORGANES CONSTITUTIFS D'UN ENSEMBLE HYBRIDE : PARTAGE DES TACHES

II.1 - Le calculateur analogique

La qualité essentielle du calculateur analogique est sa très grande vitesse de résolution des équations différentielles : cette tâche lui sera donc confiée. D'autre part la large bande passante des blocs opérationnels et le faible temps de commutation des relais électroniques offrent la possibilité d'effectuer un calcul répétitif commandé par des éléments logiques. Ainsi peut-on confier au calculateur analogique la résolution, à grande vitesse d'un système d'équations différentielles, alors que la logique parallèle


Signaux de service
Interruptions
CALCULATEUR CENTRAL
Interruptions
Lignes d'etat
Chemin des données
Signaux de service
Mini processeur
/Interface
Mini processeur
Interface
Données
Périphérique
Spécialisé
Machine
Spécialisée
Machine
Analogique
Machine
Analogique
"Maitre"
Données
Commande
Machine
Analogique
3"Esclave"

commande le déroulement des cycles de calcul, la modification des conditions initiales ainsi que les organes d'enregistrement ou de visualisation. Pendant que les opérateurs analogiques effectuent les opérations continues, la logique parallèle peut également simuler les opérations logiques d'un problème, voire même certains traitements d'informations à grande vitesse. En effet pour les opérations simples la logique parallèle est beaucoup plus rapide que le calculateur numérique. D'autre part, pouvant communiquer facilement avec le calculateur analogique et avec l'ordinateur, elle peut être utilisée avec intérêt pour le contrôle de l'exécution d'un problème hybride.

Nous avons remarqué précédamment qu'en principe l'opération de multiplication, et même de division était possible. Cette dernière étant admise sous toute réserve, il reste que les multiplieurs existent sur le panneau analogique, mais présentent rarement des garanties de précision ; il est de loin préférable de passer par le calculateur numérique, quitte à perdre légèrement en précision par les deux conversions successives analogique numérique puis numérique analogique. A ce niveau, le problème important est le temps. En effet, même si nous disposons de convertisseurs extrèmement rapides, les différents échanges peuvent introduire des retards dans le déroulement d'une simulation dans le cas d'une résolution rapide. Il faut alors prévoir une mise temporaire en condition de "gel" de la machine analogique lors de la multiplication numérique. Une autre solution consiste à utiliser les potentiomètres numériques, qui formés à partir de multiplieurs analogiques digitaux très précis, offrent l'avantage de la rapidité, sans nécessiter de passer par le calculateur.

Un autre problème à considérer est la mémorisation analogique. En effet, le plus souvent, si les éléments constitutifs de la machine sont de bonne qualité, il existe néanmoins une dérive, qui peut présenter des inconvénients pour certains problèmes. Toutefois, si l'on se satisfait de la précision ainsi obtenue, la mémorisation de plusieurs données

risque de monopoliser un grand nombre d'éléments, ce qui réduit le champ des possibilités de cablage du modèle.

Sur le calculateur analogique, il est difficile de réaliser certaines non linéarités. Bien qu'existent des générateurs de fonctions (à diodes généralement) leur utilisation demeure délicate et nécessite parfois de longues mises au point afin de générer la fonction désirée.

Enfin il est nécessaire lors de l'élaboration d'un problème de prévoir une mise à l'échelle des variables afin de rester dans les normes de fonctionnement de la machine et éviter la saturation des amplificateurs. Cette mise à l'échelle est faite préalablement sur les équations mathématiques décrivant le phénomène physique, et pourra se faire aisément sur le calculateur numérique en utilisant les programmes en bibliothèques.

Une propriété très importante du calculateur analogique est, une fois la durée de résolution du problème choisie, la possibilité de répéter indéfiniment le résultat du calcul suivant le cycle invariant : repos, conditions initiales, calcul. Il est alors très facile d'étudier l'influence du paramètre et d'obtenir ainsi une solution optimale.

II.2 - Le calculateur numérique

L'ordinateur a pour tâches principales le réglage et la lecture des composants analogiques, l'automatisation des opérations sur le calculateur analogique et le calcul hybride en boucle fermée.

Nous pouvons remarquer que l'automatisation des opérations est une généralisation des possibilités déjà offertes par la logique parallèle du calculateur analogique. Le calculateur numérique peut avoir en mémoire un programme

élaboré comprenant, par exemple, la comparaison d'un résultat de calcul à des données physiques, également en mémoire, soit en vue de déterminer de nouvelles valeurs des coefficients, soit pour choisir un modèle mathématique différent. Ces changements seront alors effectués automatiquement sur ordre du calculateur numérique. Il est possible d'automatiser un calcul par son emploi conjugué avec la logique parallèle.

Le calculateur numérique opère rapidement, mais en séquence, si bien que certaines fonctions présentent des difficultés de réalisation. C'est le cas de l'intégration par exemple, où il est nécessaire de discrétiser les variables et d'intégrer sous forme de développement en série. La précision de la solution dépend alors du temps de calcul. La résolution d'équations aux différences, ou d'équations de récurrence nécessite lors du traitement numérique un temps d'occupation important de l'unité centrale, alors qu'il peut être plus facile parfois de considérer l'équation continue décrivant le phénomène, à l'aide du calculateur analogique, et si un traitement numérique particulier est nécessaire,d'échantillonner à partir de la partie numérique. Suivant la méthode employée, les résultats ne sont pas nécessairement identiques.

Une autre possibilité d'utiliser le calculateur numérique en intégration,est de l'utiliser conjointement avec la machine analogique pour les problèmes résolus en mode de calcul multivitesse : l'intégration lente est effectuée sur l'ordinateur, les intégrations rapides seront calculées analogiquement.

D'autre part, il est possible de commander les intégrateurs séparément à partir du calculateur, ce qui peut simplifier l'élaboration de la logique parallèle.

Une autre possibilité du calculateur numérique est son utilisation en générateur de fonctions, opérateurs souvent absents sur la machine analogique. Il existe d'ailleurs d'autres "creneaux", tel que la mémorisation et la

restitution de fonctions, de variables, la génération de retard dont la durée peut être programmée, le traitement d'informations statistiques, etc...

D'une façon générale on considère le calculateur numérique d'un côté, et le calculateur analogique comme un de ses périphériques plus ou moins particulier. Lors de l'élaboration d'un problème hybride, on s'attache à diminuer le temps d'occupation de l'unité centrale de l'ordinateur et l'on se soucie peu de la partie analogique. Cela est du au fait que la partie analogique calcule en parallèle, donc nettement plus rapidement que la partie numérique ; c'est cette dernière qui a la possibilité de décision et qui commande les différentes étapes du travail. Mais si nous revenons sur la possibilité d'utiliser le calculateur en tant que multiplieur ou comme générateur de fonctions, d'une ou plusieurs variables, l'ensemble numérique est alors utilisé comme un phériphérique, la partie principale du travail se développant analogiquement ; il serait tout aussi légitime lorsqu'on détermine le "côût" de temps d'un travail, de compter le temps imparti au calculateur analogique et au calculateur numérique.

Si nous entendons par périphérique toute machine ne pouvant fonctionner véritablement qu'avec un calculateur, (cas du lecteur de cartes, de l'imprimante, etc...) la calculatrice analogique n'est pas un phériphérique au sens plein du terme. Le fait d'être utilisée avec un calculateur élargit le champ de ses possibilités, mais n'empêche en rien son fonctionnement en local, c'est-à-dire l'utilisation de ses propriétés . En fait nous entendons par périphérique toute machine utilisée par le calculateur sans en faire partie intégrante.

Nous avons défini le calculateur analogique comme étant capable d'intégrer aisément les équations différentielles, à conditions toutefois que la précision globale demandée par le calcul ne dépasse pas 10^{-4} . Au delà le calculateur numérique est

nécesaire ; grâce au traitement en virgule flottante, il supplée efficacement l'analogique pour le traitement des grandeurs à fortes variations dynamiques.

Dans la simulation de la dynamique des systèmes, la dispersion des fréquences des solutions est souvent importante. En effet, pour observer les transitoires rapides, on travaillera analogiquement avec une échelle des temps dilatée : les dérives des amplificateurs perturbent alors les phénomènes lents. Inversement si l'on choisit une échelle contractée pour étudier les phénomènes transitoires lents, les phases rapides sont cachées.

La même difficulté se retrouve lors de la résolution numérique lorsqu'il s'agit de définir le pas d'intégration. Cependant, les parties hautes et basses fréquences sont souvent assez découplées pour qu'on puisse les traiter séparément. Les premières sont câblées en analogique, les secondes programmées en numérique.

Un programme de simulation numérique doit être exécuté d'une manière quasi parallèle s'il veut conserver la notion du temps réel. Par cela le pas d'intégration minimal est égal au temps de calcul élémentaire, c'est-à-dire à la durée totale d'exécution de toutes les opérations qui sont incluses dans ce pas. Si nous voulons simuler un problème nécessitant dans le cas d'une résolution analogique l'emploi d'une centaine d'amplificateurs, nous sommes obligés de le traiter numériquement.

Si nous supposons un temps de résolution de l'ordre de grandeur de 4 000 à 5 000 cycles de bases du calculateur, nous arrivons, suivant la méthode que nous employons à une résolution d'une durée de 5 ms, pour une précision de l'ordre de 10^{-4}

Dans un cas tel due celui-ci, la simulation en temps réel purement numérique n'est alors pas possible, et la méthode purement analogique non plus. Le calcul hybride se présente alors comme la seule solution à condition de

répartir convenablement les tâches. Cette situation se reproduira lorsqu'un système présente un grand nombre de cellules semblables ; bien que le modèle mathématique relatif à chaque cellule soit le plus souvent bien adapté à la résolution analogique, cet avantage peut se transformer en un inconvénient majeur. Il est en effet nécessaire de disposer d'autant d'éléments de calcul que de cellules, ce qui peut devenir rapidement prohibitif. On peut alors simuler le fonctionnement d'un nombre restreint de cellules et appliquer une méthode voisine du calcul répétitif ou itératif en calculant successivement la propagation des variations des éléments le long des cellules.

III - CONCLUSION

Il existe de nombreux domaines d'application du calcul hybride, n'offrant pas seulement une amélioration des possibilités déjà offertes par des méthodes purement analogiques ou numériques, mais aussi, comme nous l'avons vu précédemment, présentant une méthode originale de traitement de problèmes impossibles à résoudre par ailleurs.

Pour tout système géré par ordinateur tel qu'une usine par exemple, il est nécessaire de tester par avance le comportement de l'ensemble géré par programmes. Une simulation hybride permet d'atteindre cet objectif dans lequel le système lui-même est représenté par une structure analogique ou hybride et l'ordinateur de commande par une partie de l'unité numérique du calculateur hybride.

D'autres possibilités se sont généralisées, comme l'identification des problèmes, l'optimalisation, la physique nucléaire, etc...

On ne peut évidemment pas conclure que le calcul hybride peut tout faire parce qu'il allie les avantages des méthodes numériques et analogiques. Il en revoit aussi

certains inconvénients et présente par là même des limitations; de nombreux efforts doivent encore être réalisés tant dans le domaine des méthodes que dans celui des langages. A notre niveau nous décrivons le système de base de l'interface réalisée, et mettons en évidence sur des exemples, les possibilités de résolution de quelques problèmes, les comparant avec ce qui avait pu être obtenu par une solution uniquement analogique ou numérique.

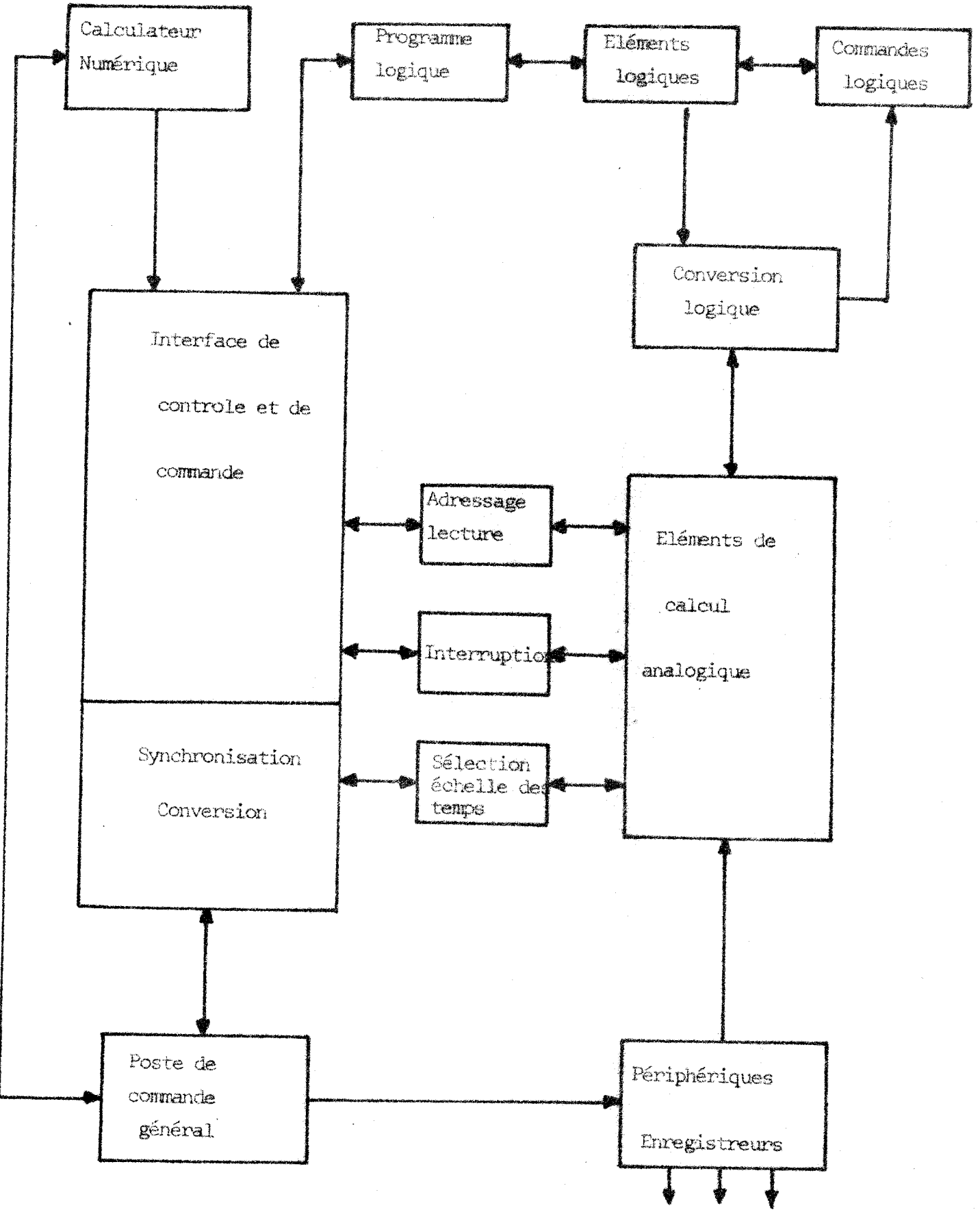


Figure I.5

Ensemble Hybride

2ème PARTIE

LE COUPLAGE

Cette partie est un résumé d'un exposé plus détaillé présenté dans le mémoire de thèse de Monsieur TARTE. Notre but est à la fois de donner une vue d'ensemble du travail commun effectué sur le couplage et de présenter les éléments essentiels que nous avons à manipuler dans la partie qui nous est plus spécifique et qui traite de la programmatinn hybride.

LE COUPLAGE

1 - Les calculateurs

Le couplage envisagé dans ce travail a été effectué en utilisant un calculateur de processus "Télémécanique T 2000" et une calculatrice hybride à courant continu "EAI 580" ; notons toutefois que cette étude reste valable pour n'importe quel type d'appareil et que les performances du système global dépendront surtout du matériel utilisé.

1.1 - L'ordinateur de processus

Le calculateur T 2000 est conçu pour prendre en compte des informations de toute forme, grâce à une gamme complète d'organes élémentaires réalisant des fonctions caractéristiques et ayant des performances variables.

1.1.1 - Organisation générale

L'organe central, a un cycle de base de 1,5 µs et les mots traités sont de 19 bits.

Le système d'interruptions comporte 16 niveaux hiérarchisés. Il permet l'acquisition et la mémorisation des appels, le tri suivant la hiérarchie, l'exécution de l'interruption et la libération du niveau.

En plus des périphériques classiques, existe une gamme de coupleurs spécialisés. On distingue :

- les périphériques logiques : il s'agit en particulier du coupleur universel, dont l'utilisation est très importante au niveau du couplage. Il assure la gestion de mots logiques TTL, en entrée et en sortie.
- les périphériques analogiques.

En particulier, nous notons la présence d'une chaine d'acquisition de valeurs analogiques. La mesure est convertie sur 14 bits + signe en un temps de 10 µs. Sa vitesse atteint 8 000 voies par seconde.

Les sorties analogiques, délivrent une tension proportionnelle à la valeur digitale affichée par programme. La résolution se fait en ± 512 points avec une vitesse de 40 µs (dont 30 µs pour la stabilisation du système).

1.1.2 - L'unité de traitement

Le code d'ordre du calculateur est de 56 instructions. Le calculateur possède 2 accumulateurs et un index. Les circuits de calculs sont parallèles et la multiplication et la division sont des opérations cablées.

Le dialogue avec les périphériques s'effectue suivant trois modes :

En programme simple, chaque opération d'entrée ou de sortie se fait par l'intermédiaire de l'accumulateur et il convient de tester si le périphérique sur lequel on effectue un échange est libre.

En mode programmé prioritaire, le périphérique indique à l'unité de traitement par le système d'interruptions qu'il est prêt à prendre en compte ou qu'il dispose de l'information.

Le mode canal, réduit la programmation et libère l'unité centrale des opérations de dialogue avec le périphérique.

1.2 - Le calculateur hybride

La calculatrice hybride est une machine à courant continu. Elle permet la réalisation simple d'équations différentielles linéaires ou non linéaires, ainsi que le fonctionnement simultané de plusieurs boucles de calcul à des vitesses d'intégration différentes.

Cette propriété est le travail le plus fondamental de ce calculateur et est à l'origine des méthodes de sous-routines et de calcul multivitesse. Ceci n'est rendu possible que par l'adjonction à la partie analogique classique, d'une partie logique synchrone, capable de commander séparément chaque intégrateur.

1.2.1 - Les fonctions analogiques

L'amplificateur opérationnel est l'élément majeur de cet ensemble. on distingue :

- l'intégrateur, commandable par une logique parallèle,
- le sommateur et l'inverseur,
- le comparateur...

1.2.2 - Les fonctions logiques

La logique de l'EAI est une logique positive 0 volt , 5 volts dont les fonctions de base sont :

- les portes "ET"
- les compteurs
- les bascules synchromes...

1.2.3 - Le pupitre de commande

Un pupitre de commande contrôle l'ensemble et permet le choix d'un certain nombre de fonctions analogiques et logiques.

L'utilisateur choisit la constante d'intégration des amplificateurs et la fréquence de l'horloge.

Le pupitre exécute la sélection des potentiomètres et l'affichage de la valeur analogique.

Notre problème consiste donc à effectuer une simulation numérique du pupitre, afin de réaliser les mêmes opérations par le calculateur de processus.

1.3 - Conclusions

La description des deux calculatrices, nous amène maintenant à aborder l'étude du couplage entre ces deux machines. Les fonctions du pupitre sont facilement réalisées, les références logiques des deux machines étant les mêmes.

L'affichage des potentiomètres, par contre risque d'être plus délicat à concevoir. Il entraine l'utilisation d'un changement de logique (0 volt , 5 volts) en (0 volt, 20volts)

Le problème est surtout d'assurer un grand nombre de commandes avec le minimum d'informations numériques. L'étude a été envisagée pour le couplage de deux EAI et de ce fait, nous avons adopté une configuration qui n'est pas forcément celle convenant au couplage avec une seule machine.

2 - L'interface

La première étape dans la réalisation d'un ensemble hybride par couplage d'un ordinateur de processus et d'une calculatrice hybride est d'effectuer par programmation toutes les opérations manuelles réalisables au pupitre de la calculatrice.

Nous devons noter que les regroupements d'informations effectués ont été choisis suivant plusieurs critères :

- minimiser le nombre de signaux de commande,
- envisager un décodage simple et des vérifications au niveau de l'interface,
- assurer des facilités de codage au niveau de la programmation
- tenir compte de la technologie utilisée.

Dans toute notre réalisation, nous nous sommes intéressés aux liaisons "Hardware" - "Software", afin d'assurer pour chaque partie un travail minimum, avec un maximum d'efficacité et de rapidité.

2.1 - Commande du pupitre

2.1.1 - Etude des signaux EAI

L'analyse du système de commande des différentes fonctions est indispensable avant l'exposé de notre réalisation.

Les signaux analogiques, logiques, temps d'intégration, fréquence d'horloge, sont mémorisés par des bascules.

Dans chacun des groupes précédents, la présence d'une commande élimine toutes les autres.

Pour les fonctions analogiques, nous avons les commandes :

SP : "SET POT", arrêt des calculs
IC : "Initial Condition", conditions initiales
OP : "Operate", calcul, résolution
HD : "Hold", gel des tensions
PP : "Program Panel", répétitif

Les commandes logiques sont :

C : "Clear", remise à zéro des bascules et compteurs
S : "STOP", arrêt des horloges
R : "RUN", exécution du programme logique
PP : "PATCH PANEL", répétitif de la logique

Il existe deux temps d'intégration : 1 s ou 2 ms.

Les horloges sont au nombre de trois : 10^6, 10^5 ou 10^1 impulsions par seconde, et pilotent toutes la logique des calculatrices.

2.1.2 - But à atteindre

Plusieurs réalisations sont possibles ; mais il faut noter qu'une EAI peut en piloter une autre en transmettant tous les signaux de commande. Un bouton "RMT" (REMOTE) permet de mettre en relation directe toutes les bascules du système avec les informations extérieures venant de l'organe maître.

Afin d'assurer une commande simultanée des deux calculatrices nous avons considéré un système de bascules au niveau de l'interface qui mémorise les informations, et ne prend en compte ces dernières qu'à l'instant où toutes les sorties ont été effectuées.

2.1.3 - Réalisation

La commande de toutes ces fonctions nécessite huit informations binaires que nous avons codées de cette façon :

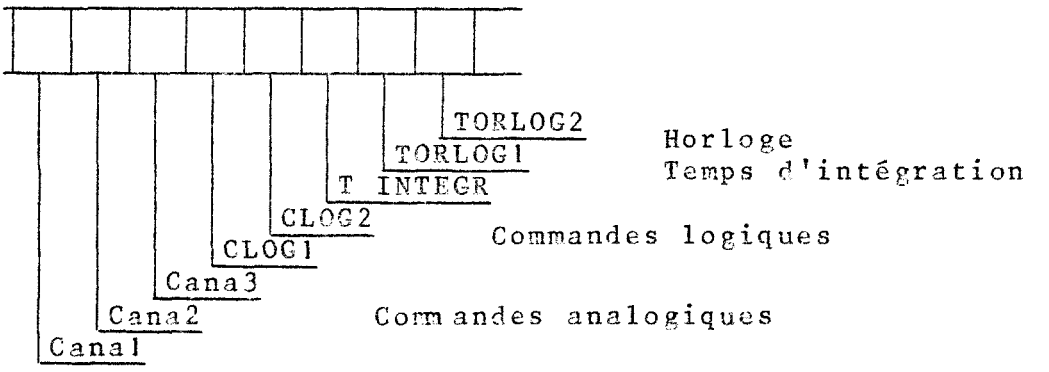


Figure 2.1
Le mot de commande

2.1.4 - Le mot d'état

Les informations de commande de l'EAI, mémorisées dans des bascules, forment un registre fictif que nous appelons : "Registre d'état du système". Il nous permet de connaître les différentes fonctions en cours.

Nous détectons, par l'analyse de ce mot d'état, des erreurs dues à des modules défectueux.

Le mot d'état comporte 13 informations ; la présence d'un 1 logique indique la réalisation de la fonction correspondante sauf le bit n° 9 qui a deux états significatifs (figure 2.2).

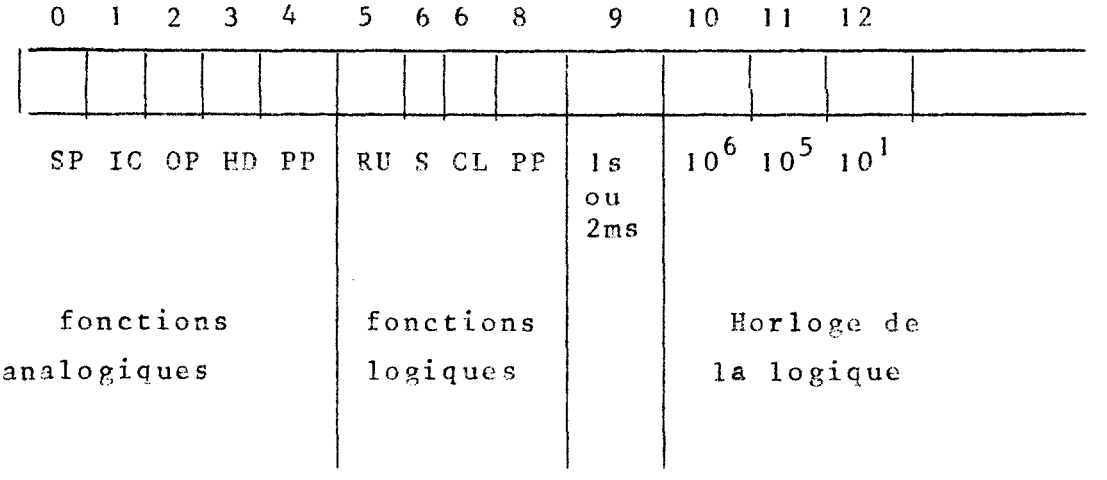


Figure 2.2

2.1.5 - Conclusions

La réalisation de ce système entraine un certain nombre de contraintes. En effet, lors d'une utilisation des EAI en local, c'est-à-dire, indépendamment du calculateur, nous ne devons pas être influencé par l'interface. Il est indispensable de débrancher certaines lignes afin d'éviter d'éventuels ennuis.

Cette modification a été réalisée à l'aide de relais qui sont commutés une fois pour toute en début d'utilisation grâce à une information venant du calculateur. Cette information notée "SELECT", a aussi un autre but, celui de réaliser la sélection de la calculatrice EAI.

Les différentes lignes coupées sont :

- les lignes des diverses fréquences d'horloge,
- le temps d'intégration,
- la durée du répétitif.

Le but de notre étude étant la commande intégrale des machines hybrides, il convient de pouvoir piloter individuellement chaque intégrateur.

2.2 - Commande individuelle des intégrateurs

Il est possible de commander les intégrateurs par la logique parallèle de l'EAI.

En mode normal, un cavalier de bouclage sur le panneau analogique, fournit à l'intégrateur les signaux (OP) bus et (IC) bus qui déterminent le mode de fonctionnement.

Ces signaux sont donnés par la table de verité figure 2.3.

Si l'on désire commander l'intégrateur par la logique parallèle, les signaux logiques sont envoyés directement sur le panneau avant et remplacent ceux fournis par le cavalier de bouclage.

IC Bus	OP Bus	Commande
0	0	Gel des calculs
0	1	Calculs
1	1	Cond. Init.
1	0	" "

Figure 2.3

Table de vérité de la commande individuelle des intégrateurs

2.2.1 - Etude des signaux

Nous ne pouvons pas traiter directement les signaux fournis par les bus (IC) et (OP), car cela entraine une commande identique.

Ces signaux étant délivrés à chaque module, il est possible de les remplacer par deux informations élaborées au niveau de l'interface.

2.2.2 - But à atteindre

La commande doit pouvoir s'effectuer soit par le calculateur, soit par le pupitre, et ceci pour chaque intégrateur.

Une information nous définie quel est l'organe assurant la commande.

L'information RAFFICH, précédente, assure la mémorisation des informations, mais aussi la commande simultanée des fonctions.

2.2.3 - Réalisation pratique

Soit ENTOP et ENTIC, les informations fournies

par les bus (OP) et (IC), INFOP et INFIC, sont deux informations qui jouent le même rôle que ENTOP et ENTIC. Si T nous définit le mode de commande (T = 1, commande par le pupitre), nous pouvons écrire :

$$SORIC = T \cdot ENTIC + \overline{T} \cdot INFIC$$

$$SOROP = T \cdot ENTOP + \overline{T} \cdot INFOP$$

Appelons alors BITIC et BITOP, les informations venant du calculateur. En codant ces informations, nous avons

$$T = \overline{BITIC} \cdot \overline{BITOP}$$
$$INFIC = \overline{BITOP}$$
$$INFOP = \overline{BITIC}$$

d'où

$$SORIC = \overline{BITIC} \cdot ENTIC + BITIC \cdot \overline{BITOP}$$
$$SOROP = \overline{BITOP} \cdot ENTOP + \overline{BITIC} \cdot BITOP$$

En sortie de la bascule, nous avons les informations SORIC et SOROP mémorisées et appelées MEMOIC et MEMOOP.

L'utilisation des calculatrices en local entraine une restitution des signaux (IC) bus et (OP) bus du pupitre.

Un bouton poussoir au niveau de l'interface va permettre de restituer les informations initiales aux calculatrices.

Soit "CHOISI", le nom de ce signal.

Nous avons :

$$ICDEOR = CHOISI \cdot MEMOIC + \overline{CHOISI} \cdot ENTIC$$
$$OPDEOR = CHOISI \cdot MEMOOP + \overline{CHOISI} \cdot ENTOP$$

Le nombre de modules intégrateur est de 8 ; 16 informations venant du T 2000 sont nécessaires à la commande.

2.2.4 - Conclusions

La possibilité de commander chaque intégrateur indépendamment du pupitre donne au système une puissance de calcul assez importante ; cette possibilité réduit le volume de commande par la logique parallèle, les décisions de mises en conditions initiales de calcul sont fournies par le calculateur.

2.3 - Affichage des potentiomètres asservis

Les problèmes de recherches paramètriques, ou de modèles nécessitent la possibilité de changer les conditions initiales et les paramètres de bouclage. Pour cela il faut assurer la sélection du potentiomètre, le décodage de la valeur à afficher, puis la transmission de cette valeur.

2.3.1 - Réalisation au niveau EAI

Un potentiomètre ne peut être affiché qu'en mode arrêt des calculs. Cette restriction entraine un certain nombre de contraintes au niveau de l'utilisatinn de la calculatrice dans le couplage, car le temps de sélection de l'affichage n'est pas négligeable.

L'affichage d'un potentiomètre s'effectue en plusieurs étapes ; nous devons d'abord sélectionner l'adresse du potentiomètre (10 à 79), décoder la valeur sur 4 digits décimaux, puis afficher cette valeur grâce au bouton poussoir "SET".

Une matrice de relais sélectionne l'adresse du potentiomètre. La tension d'alimentation de ces relais est de - 20 volts, l'affichage de la valeur se fait pour 4 digits décimaux. Un réseau de résistances pondérées délivre un courant proportionnel aux valeurs affichées. Un amplificateur opérationnel transforme ce courant en tension directement

utilisable. Cette tension est connectée à la ligne "SERVO IN" et sert de comparaison pour le positionnement du potentiomètre commandé par le bouton "SET".

Durant toutes la durée de l'affichage, sur une ligne : "NULL" apparaît une tension de +15 volts.

2.3.2 - But à atteindre et réalisation

Nous devons effectuer toutes ces opérations de la manière la plus rapide en assurant la précision la plus grande.

Il est nécessaire,au niveau du coupleur d'effectuer un changement de logique qui au niveau 5 volts fasse correspondre - 20 volts.

L'adresse du potentiomètre est sélectionnée par deux informations codées en BCD.

La valeur analogique du potentiomètre est fournie par un convertisseur digital-analogique à 9bits plus signe. Le signal logique que nous appelons "POTSET" assure l'affichage du potentiomètre.

La retombée du signal "NULL" indique la fin de l'opération.

2.3.3 - Performances

L'enchainement de programmation s'effectue de manière à assurer la plus grande rapidité, mais cette durée est uniquement fonction du moteur d'entrainement du potentiomètre. La précision du système est celle du convertisseur soit 2.10^{-3}. Afin d'éviter d'éventuels ennuis, chaque ligne de sélection du potentiomètre a été coupée par un relais commandé par le bit "SELECT".

A chaque affichage, la valeur est relue par la chaine d'acquisition afin de controler si l'opération s'est déroulée correctement.

L'affichage de ces potentiomètres présente une contrainte d'utilisation car la commande ne peut s'effectuer qu'en arrêt des calculs.

Notre intention est donc de réaliser des potentiomètres numériques qui peuvent changer de valeur en cours de calcul, grâce à leur temps de conversion faible.

2.4 - Les potentiomètres numériques

Le temps de conversion doit être faible et la précision de l'ordre de 10^{-3}.

Ils sont destinés à faire le produit d'une tension analogique par une valeur digitale.

2.4.1 - Position du problème

Un certain nombre de contraintes sont à respecter.

Il faut prévoir une matrice d'adressage du potentiomètre et une sauvegarde du paramètre α, valeur de l'affichage.

Ces potentiomètres doivent être commandables également en mode local.

2.4.2 - Solution envisagée

La solution qui a été retenue pour notre système consiste en un convertisseur digital-analogique multiplieur 10 bits (Hybrid System DAC 315-10).

La vitesse d'évolution de ce système est de 5 volts par μs. La bande passante de 100KHz à 3 db et l'impédance d'entrée de 10 KΩ. Enfin de rétablir le signe en sortie, il faut ajouter au système un amplificateur opérationnel.

2.4.3 - Mode de sélection

Nous envisageons l'utilisation du potentiomètre à la fois dans le couplage et en local.

Chaque potentiomètre est défini par une adresse qui est celle de son emplacement dans le bac interface. Le mode de sélection est effectué par un bouton poussoir "MODE":

MODE = 1 Sélection par le T 2000
MODE = 0 Sélection par le clavier de l'interface

L'adresse fournie par le T 2000, codée en BCD, afin de minimiser le nombre d'informations, est transformée en décimal au niveau de l'interface.

Sur un emplacement de cartes arrivent quatre lignes qui sont les dizaines et les unités de l'adresse.

L'adresse du potentiomètre s'écrie :

$$ADRPOT = (LOCUNIT \cdot LOCDIZ) \cdot \overline{MODE} + (CALUNIT \cdot CALDIZ)\ MODE$$

2.4.4 - Sélection de la valeur

Soit les signaux suivants :

- CALVAL : valeur sur 10 bits venant du calculateur
- LOCVAL : valeur sur 10 bits venant du pupitre
- PASAPA : bouton de sélection du mode pas à pas
- INCRE : bouton d'affichage d'un registre de la valeur de l'incrément
- CALSET : information d'affichage venant du T 2000
- LOCSET : information d'affichage venant du pupitre
- PASVAL : valeur d'incrémentation du pas à pas
- VALSET : signal d'horloge du registre tampon du potentiomètre
- POTVAL : valeur fournie au registre tampon du potentiomètre

Nous avons les réalisations suivantes :

$$COMVAL = LOCVAL \cdot \overline{MODE} + CALVAL \cdot MODE$$

$$POTVAL = COMVAL \cdot \overline{PASAPA} + ADIVAL \cdot PASAPA$$

$$ADIVAL = (PREVAL) + (PASVAL)$$ (addition de 2 valeurs)

PASVAL = LOCVAL initialisé par INCRE
INIVAL = LOCVAL . $\overline{\text{PASAPA}}$ + ADDREG . PASAPA
"ADDREG" prend la valeur "ADIVAL", lors de l'affichage du potentiomètre
"PREVAL" prend la valeur "INIVAL", grâce au signal "LOCSET" retardé du temps nécessaire à l'affichage du potentiomètre

On peut remarquer que pour simplifier la logique, nous utilisons des multiplexeurs 2 bits, réalisant les signaux "INIVAL","COMVAL", "POTVAL".

2.4.5 - Affichage de la valeur

Nous disposons maintenant de la valeur du potentiomètre "POTVAL".

Cette valeur doit être mémorisée lorsque le potentiomètre est sélectionné.

L'affichage est effectué soit par "CALSET", soit par "LOCSET".

VALSET = (LOCSET + CALSET) . ADRPOT.

2.4.6 - Conclusions

Le système élaboré, fournit un outil puissant, vue la rapidité de travail et de conversion. L'erreur apportée, intervient uniquement dans le choix du convertisseur numérique analogique. Le temps de réponse du potentiomètre pour une variation de 0 à 10 volts est de 7 µs, ce qui pratiquement, en tenant compte des différentes opérations de sélection, donne une fréquence d'affichage de 100 KHz, donc une fréquence semblable à celle de la chaine d'acquisition.

2.5 - Acquisition de données

Nous disposons d'une chaine analogique numérique

possédant huit voies. La vitesse de la chaine est limitée par le système d'acquisition et n'atteint que 8 000 voies par seconde.

Le T 2000 par l'intermédiaire du coupleur envoie et reçoit des informations du controleur haut niveau.

La rapidité du système étant grande, le calculateur travaille en mode canal. Il suffit d'initialiser l'échange, et le canal avertit l'unité centrale de la fin de travail.

La précision que l'on peut attendre de la chaine est de 1 à 2.10^{-4}.

2.6 - La saturation des amplificateurs

Lors de l'étude d'un problème, la saturation d'un amplificateur occasionne des erreurs de calcul, même si cette saturation n'est que momentanée.

2.6.1 - Etude des signaux

Un amplificateur est considéré comme saturé, lorsque sa tension de sortie devient supérieure à 11,5 volts.

Sur une ligne de l'amplificateur, apparaît un signal périodique nommé "SATUR".

En même temps, sur une ligne commune appelée "$\overline{\text{OVLD}}$" à tous les amplificateurs, le système élabore un signal qui passe à zéro lors d'une saturation.

2.6.2 - But à atteindre et réalisation

Nous devons être averti le plus rapidement de la saturation d'un amplificateur, afin de bloquer les calculs pour vérifier le montage.

Le signal "OVLD" fournit au calculateur, l'indication d'une saturation en créant un appel prioritaire au

niveau de l'unité centrale. Il devient alors possible, après mise en forme des signaux "SATUR", de repérer les amplificateurs qui ont déclenché l'appel. Afin de faciliter cette lecture, nous utiliserons des multiplexeurs pour limiter le nombre d'informations à lire à chaque fois. L'EAI est remise en arrêt des calculs lorsque toutes les sorties des multiplexeurs ont été lues.

2.7 - Conclusions

Les contraintes de rapidité et de précision que nous avons envisagées pour notre dispositif, ne peuvent être valablement appréciées que sur des exemples. Cela fera l'objet de la dernière partie de ce mémoire. Mais la qualité de l'ensemble hybride ne peut être assurée que si chacune des parties est en parfait état de marche ; nous allons donc considérer dans le paragraphe suivant le problème des tests de notre système.

3 - Les tests du système

Pour permettre au système d'assurer complètement son rôle, il est intéressant d'effectuer un certain nombre de tests, tendant à détecter et à localiser les défauts des divers modules.

Cet ensemble de tests se décompose en trois parties :

- test du calculateur numérique
- test de l'interface
- test des calculatrices hybrides

3.1 - Test du calculateur

La Société Télémécanique fournit différentes bandes de tests du calculateur et de ses organes d'entrées-sorties.

Ces tests doivent être effectués régulièrement afin d'assurer la maintenance du matériel.

3.2 - Test de l'interface

La deuxième étape à réaliser est le test de l'interface, c'est-à-dire la vérification de tous les circuits de commande. Pour réduire le nombre d'informations à lire, ou à transmettre, les signaux sont regroupés sur des multiplexeurs.

3.2.1 - Présentation du problème

Ouatre modèles différents de plaques sont à tester :

- cartes de commande pupitre
- cartes de changement de logique
- cartes de commande individuelle des intégrateurs
- cartes de saturation.

Nous avons constaté que le multiplexage des informations n'est nécessaire que pour les signaux de sortie.

3.2.2 - Réalisation pratique

Nous avons cherché à déterminer pour chacune des plaques de circuit, les séquences optimales de test à effectuer pour détecter un défaut.

L'analyse du mot de sortie nous permet de générer, lorsqu'il existe un défaut, diverses séquences , afin de localiser les éléments défectueux. Lorsqu'il n'est pas possible de mettre en cause un élément, le diagnostic se limite à une classe de modules.

Pour effectuer cette série de tests, il faut être certain que les plaques de multiplexage n'apporte aucune erreur.

Un emplacement dans le bac de test, leur est réservé afin de déterminer les opérateurs hors de fonctionnement.

3.3 - Test des calculatrices hybrides

Le principe de ce test est simple. Nous analysons tous les amplificateurs afin de s'assurer de leur bon fonctionnement, tant au point de vue analogique que logique.

Les tests effectués sont fonction de la nature du module.

Nous vérifions que pour les inverseurs, la tension de sortie est l'opposé de la tension d'entrée.

La somme de deux signaux teste les sommateurs.

Pour les intégrateurs, nous vérifions le système de conditions initiales, l'intégration, et la dérive des amplificateurs.

Les éléments non linéaires sont aussi testés par des montages plus particuliers, qui nous permettent de balayer tout leur domaine de variation.

3.4 - Conclusion

Cette série de test s'effectue de manière continue grâce à un panneau pré-cablé. Pour éliminer les risques d'erreurs, dus aux affichages de potentiomètres, nous ne tenons compte dans cette série de tests que des informations lues par la chaine d'acquisition. Les éléments défectueux sont signalés par un message sur l'imprimante.

CONCLUSION

Après la description de l'interface, notre but est de définir le langage hybride adapté à notre système.

Ce langage a été élaboré afin de permettre à l'utilisateur une mise en oeuvre simple sans l'obligation de connaître la structure du couplage.

IIIème PARTIE

PROGRAMMATION

Nous traitons dans cette partie les méthodes de programmation propres à un ensemble hybride. A partir des données relatives à une liaison calculateur-périphériques, nous mettons en évidence les modules programmés nécessaires à la gestion des échanges, nous permettant d'élaborer le système de base. D'autre part, il convient ensuite d'ajouter les éléments concernant notre couplage particulier et qui découlait des possibilités techniques offertes pour une exploitation aisée de l'ensemble hybride.

Par la suite, nous ne pourrons éviter l'emploi des termes anglo-Saxons de "hardware" et "software". Le premier comprend toutes les fonctions cablées et fixées une fois pour toute, par le second nous entendons tout ce qui est programmé, c'est à dire la liaison entre l'interface et l'opérateur.

LANGAGE ET PROGRAMMATION

1 - INTRODUCTION

L'absence de langage permettant de faire fonctionner simultanément la machine numérique et la machine analogique a été initialement un frein au développement du calcul hybride.

Actuellement un software hybride existe permettant d'utiliser avec efficacité et rapidité l'ensemble des deux calculateurs. Cela présente toutefois l'inconvénient d'éloigner l'analyste et le programmeur de la machine et, par la même, de ne profiter d'avantages liés à certaines caractéristiques propres aux calculateurs. En résumé, le software hybride peut être comparé à un organe de liaison programmé facilitant la communication entre l'utilisateur et l'ensemble hybride.

Les traits caractéristiques du software hybride répondent aux exigences propres au calcul hybride. Les principales sont les suivantes.

1.1 - Exigences liées à la nature du problème

Dans de nombreux problèmes, s'exerce une contrainte de temps minimal provenant du besoin de communiquer avec un système externe au calculateur numérique qui, lui, travaille sur une base de temps réelle. Le programme hybride doit donc être capable d'utiliser des informations continues ou discrètes provenant de supports externes, de les traiter et de répondre à la fonction demandée dans un intervalle de temps prédéterminé. Il travaille en temps réel.

1.2 - Exigences liées à la préparation d'un programme

Il est d'une grande importance que, quel que soit le langage de programmation utilisé, le programme hybride fonctionne comme une entité ; Sa fonction est d'adapter le mode d'opération séquentiel du calculateur numérique aux propriétés parallèles du calculateur analogique.

Il doit donc réaliser la synchronisation des opérations analogiques et numériques, l'optimisation du temps de calcul nécessaire, la communication de données et des informations de commande entre les programmes analogique et numérique.

1.3 - Exigences liées aux procédés opératoires

Le calculateur hybride, d'après sa conception, offre des possibilités de dialogue avec l'opérateur. Le software hybride doit donc tenir compte de la possibilité d'intervention de celui-ci.

Enfin une caractéristique importante est de posséder des programmes de diagnostics permettant de vérifier les opérations de calcul de l'ensemble et plus spécialement les opérations analogiques.

On peut donc décomposer le système hybride en plusieurs classes : les langages de programmation, les systèmes de commande, les programmes de bibliothèque, les programmes utilitaires, les programmes de mise au point et de vérification.

En fait, le software hybride a beaucoup de points communs avec un software numérique classique. Le langage de programmation, les programmes bibliothèque et une partie du système moniteur peuvent ne pas être spécifiques à l'ensemble hybride.

2 - SPECIFICATION DE L'ENSEMBLE HYBRIDE

Nous étudions dans ce paragraphe la philosophie du software hybride afin de mettre en évidence les différentes fonctions requises par l'utilisateur. Nous déterminons les conditions d'utilisation du système et dégageons sa structure première, au fur et à mesure de l'avancement de l'exposé.

Nous nous interessons plus particulièrement ici au software mais étant donné la dépendance des deux parties certains aspects du hardware doivent être discutés.

Nous entreprenons cette approche en nous plaçant de deux points de vue légèrement différents. Le premier tient compte des spécifications préliminaires découlant d'un système hypothétique idéal. Les contraintes sont alors très précises : longueur de mots, nombre de registres, sous-programmes du système, rapidité des processeurs etc... Le deuxième point de vue est celui de l'utilisateur qui ne s'intéresse qu'aux résultats et pour qui le mode opératoire doit être le plus simple possible.

2.1 - Utilisation du Fortran

Le couplage étant conçu pour être utilisé par des chercheurs de laboratoire il semble intéressant d'utiliser la langage Fortran, capable de résoudre de nombreux problèmes scientifiques, facile à utiliser et de plus bien connu. Il n'est en effet pas concevable d'avoir à apprendre un langage assembleur pour utiliser l'ensemble hybride. Néanmoins, il se peut que l'utilisation du Fortran pose des problèmes de rapidité dans des cas particuliers. Pour être tout à fait objectif, il faut remarquer que le compilateur Fortran ne fournit pas un code objet aussi efficace que celui écrit directement en langage assembleur ; ce dernier est très souple mais nécessite du programmeur une bonne connaissance du calculateur. Il est donc nécessaire de prévoir les deux possibilités.

2.2 - Les opérations sur machine

Chaque utilisateur étant responsable de son programme doit assurer à l'aide des directives du superviseur son implantation en mémoire, sa compilation etc... Ces procédures d'utilisation sont simplifiées au maximum, afin de ne définir que les ressources (fichiers par exemple) strictement nécessaires à la résolution du problème.

.2.3 - Les interruptions

Nous avons à notre disposition deux consoles analogiques ; il est donc nécessaire de prévoir leur fonctionnement en mode esclave ou en mode indépendant. Le calculateur numérique traite les deux simulations simultanément, ce qui suppose le branchement aux routines de traitement des interruptions. C'est à ce niveau que l'on trouve des problèmes de délais et de réentrance de routines lors d'un traitement en temps réel.

2.4 - Aménagement des fichiers

Pour faciliter la réutilisation d'un programme, il est souhaitable de prévoir la sauvegarde du code objet résultant d'une compilation ou d'un assemblage. Le rôle du moniteur est de restituer le contenu des fichiers et d'interdire les interférences possibles entre utilisateurs en les verrouillant à l'aide d'un code. Une fonction moniteur permet en outre le déchargement en mémoire d'un programme et sa mise en exécution automatique.

2.5 - La bibliothèque du système

La bibliothèque du système comprend les programmes d'utilisation courante et d'intérêt général ; sa gestion inclut la suppression des routines qui n'ont plus cours ou qui ont été modifiées.

Les programmes de traitement des interruptions doivent être réentrants dans la mesure du possible pour éviter lors de traitement simultané la présence de copies multiples en mémoire. La non - réentrance permet cependant un gain de temps mais perd de la place en mémoire vive.

2.6 - Exécution au niveau des machines hybrides

L'exécution se fait à partir de routines de contrôle qui se greffent sur le moniteur. Etant élaborées par le laboratoire, elles résultent d'une bonne connaissance du couplage et correspondent à une juste adaptation aux capacités de l'ensemble hybride.

3 - LIAISON ENTRE LES DIVERS NIVEAUX DU SYSTEME HYBRIDE

Il est possible de diviser les fonctions hardware en trois parties :

- les signaux de service
- le transfert des données
- le transfert des signaux.

La nature d'un interface dépend des machines à coupler, notamment par la synchronisation des signaux, leurs formes, le format des échanges etc...

Le software nécessaire à une telle liaison contient donc au minimum les procédures d'accès permettant l'exécution dans le calculateur central des programmes d'application et par leur intermédiaire l'utilisation des terminaux.

En outre une partie du software orientée principalement vers l'exploitation optimale des machines veille à l'exécution des tâches qui leur sont spécifiques. Elle consiste en moniteurs spécialisés, traducteurs, routines de tests et de diagnostics. De plus il est souhaitable d'utiliser un système en temps partagé par l'intermédiaire d'un moniteur temps réel, tout en permettant le mode conversationnel et un grand nombre d'échanges. Enfin, les différentes machines doivent pouvoir travailler de manière indépendante du calculateur numérique.

3.1 - Les liaisons hardware

Il existe un grand nombre de manières de fixer les fonctions spécifiques à un ou plusieurs interfaces ; néanmoins on y trouvera nécessairement des fonctions telles que le contrôle (interruption, temporisation, multiplexage,

sélection des périphériques, horloges ....) le transfert des données (codage et mise en forme) et la transmission des signaux.

1.a - Les signaux de contrôle

Ils sont générés par le calculateur central et par l'unité terminale et interprétés par programmation. Certains tests peuvent être cablés entrainant des décisions automatiques. Les programmes correspondants forment ce qu'on appelle les méthodes d'accès et seront définis plus précisement par la suite.

Ces signaux sont élaborés, soit sous forme d'impulsions sur des lignes sélectionnées, soit par trains d'impulsions de différents formats envoyés en parallèle ou en série sur des lignes indépendantes, soit en parallèle avec des données. Par exemple, la sélection d'une unité physique se fera par des impulsions envoyées en parallèle suivant un format prédéfini, alors que la mise en occupation correspondante peut n'être représentée que par un niveau sur une ligne spécialisée. Enfin, un signal de contrôle peut être envoyé en même temps que les données pour permettre la détection des erreurs de transmissions (contrôle de parité).

Les signaux de contrôle donnent lieu aussi à des séquences logiques : dans le cas le plus simple, seul l'ordre de la séquence importera, mais il est possible de définir plusieurs comportements suivant l'espacement des signaux dans le temps, particulièrement pour les systèmes synchrones. Cependant, même pour les systèmes asynchrones, il est nécessaire de définir un temps maximum autorisé, afin d'éviter, en cas de mauvais fonctionnement de l'unité périphérique, unetrop large utilisation de l'unité centrale.

On peut distinguer dans les fonctions de contrôle quatre étapes successives : la demande de l'échange, l'initialisation, son contrôle et l'achèvement.

- La demande d'échange :

Afin d'entamer un échange, il est nécessaire d'adresser le périphérique désiré. Il faut alors attendre que l'unité concernée soit libre. Une meilleure solution consiste à constituer une file des demandes non satisfaites, complétée par une scrutation programmée ou non. Le calculateur central traite les demandes et met en pile suivant ses propres critères, sans échanges préalables avec l'unité périphérique. Une forme plus impérative de demande utilise le système d'interruptions prioritaires : l'unité centrale prend l'initiative de l'échange et est avertie par le périphérique lors de son achèvement. Le service dans la file d'attente est du type "premier arrivé, premier servi" : il donne lieu à une gestion périlleuse, nécessitant l'institution de priorités.

- L'initialisation

Une fois la demande satisfaite, cette phase permet de remettre à jour les registres du périphérique afin d'isoler cette nouvelle opération de la précédente, de sélectionner le mode opératoire et de fournir les paramètres nécessaires. Nous arrivons alors au début de l'échange ainsi qu'à la séquence de contrôle appropriée.

- Le contrôle

L'initialisation du transfert étant effectuée, les paramètres cablés conduisent cet échange jusqu'à la fin ou l'interruption de cette phase. Les anomalies sont détectées simultanément par hardware ou par un programme très réduit. Il est intéressant d'effectuer les contrôles au maximum par un système précablé, les opérations pouvant alors se faire en parallèle plûtôt que par software, le traitement séquentiel pouvant présenter des inconvénients lors d'une utilisation en

temps réel. Pour un ensemble plus performant, il est utile de prévoir lorsqu'une erreur est reconnue, non plus la suppression et l'arrêt total du travail responsable, mais le traitement de l'erreur, soit à partir de la console de l'opérateur soit par programme.

- La finition

L'échange est achevé, c'est àdire qu'on peut tester la manière dont il s'est déroulé grâce à certains paramètres (mots d'état).

Le contrôle software intervient dans la sélection des programmes propres à l'échange, dans la définition des options demandées par l'utilisateur et dans l'ordonnancement des échanges avec le superviseur.

1.b - Le transfert des données

Le transfert des données nécessite la définition du codage, du format et du sens du transfert. Le premier se rapporte à la forme du signal, la définition du caractère, des nombres etc... Le format se refère à l'ordre dans lequel arrivent les signaux, tel que la longueur d'un mot lorsque la transmission se fait en série, le nombre de mots ou d'informations pour permettre certains contrôles. C'est aussi à ce niveau qu'il faut décider des différents modes possibles : les transferts peuvent se faire en simplex (transfert selon une direction fixe), en semi-duplex (transmission suivant plusieurs directions, mais une seule possibilité à la fois) ou duplex complet permettant des transferts dans plusieurs directions simultanément. Ceci nécessite un grand nombre de lignes séparées pour chaque direction, ou l'utilisation d'un multiplexage (division en fréquence) indépendante du temps, chaque solution représentant une amélioration mais se traduisant par un coût plus élevé.

1.c - La transmission des signaux

Cette partie concerne surtout les moyens de transmission tels que la modulation - démodulation, les transmetteurs et les capteurs, le bruit, les bandes passantes les mises en forme des signaux, les niveaux de conversions... Elle dépend surtout du type des machines utilisées et varie d'une configuration à l'autre. Nous ne nous attarderons donc pas sur ce sujet, traité dans la deuxième partie.

3.2 - Le Software

Il est maintenant possible de définir la structure du système d'exploitation de plusieurs machines. Nous précisons ici les programmes de base à inclure pour rendre les échanges faciles et diminuer le nombre d'opérations à exécuter.

Ces programmes doivent assurer la liaison des différentes unités physiques entre elles par l'intermédiaire du calculateur central. Ils comprennent les méthodes d'accès, les moniteurs, les programmes de test, les traducteurs et permettent le traitement de terminaux et le mode conversationnel ; Il sera utile de les compléter par des moniteurs temps réel, des superviseurs contrôlant les transferts, et traitant les interruptions externes. Cet ensemble est schématisé par la figure 1.

2.a - Les méthodes d'accès

Elles correspondent aux types de programmes les plus importants pour une liaison d'un calculateur à un ensemble de périphériques et forment en quelque sorte la

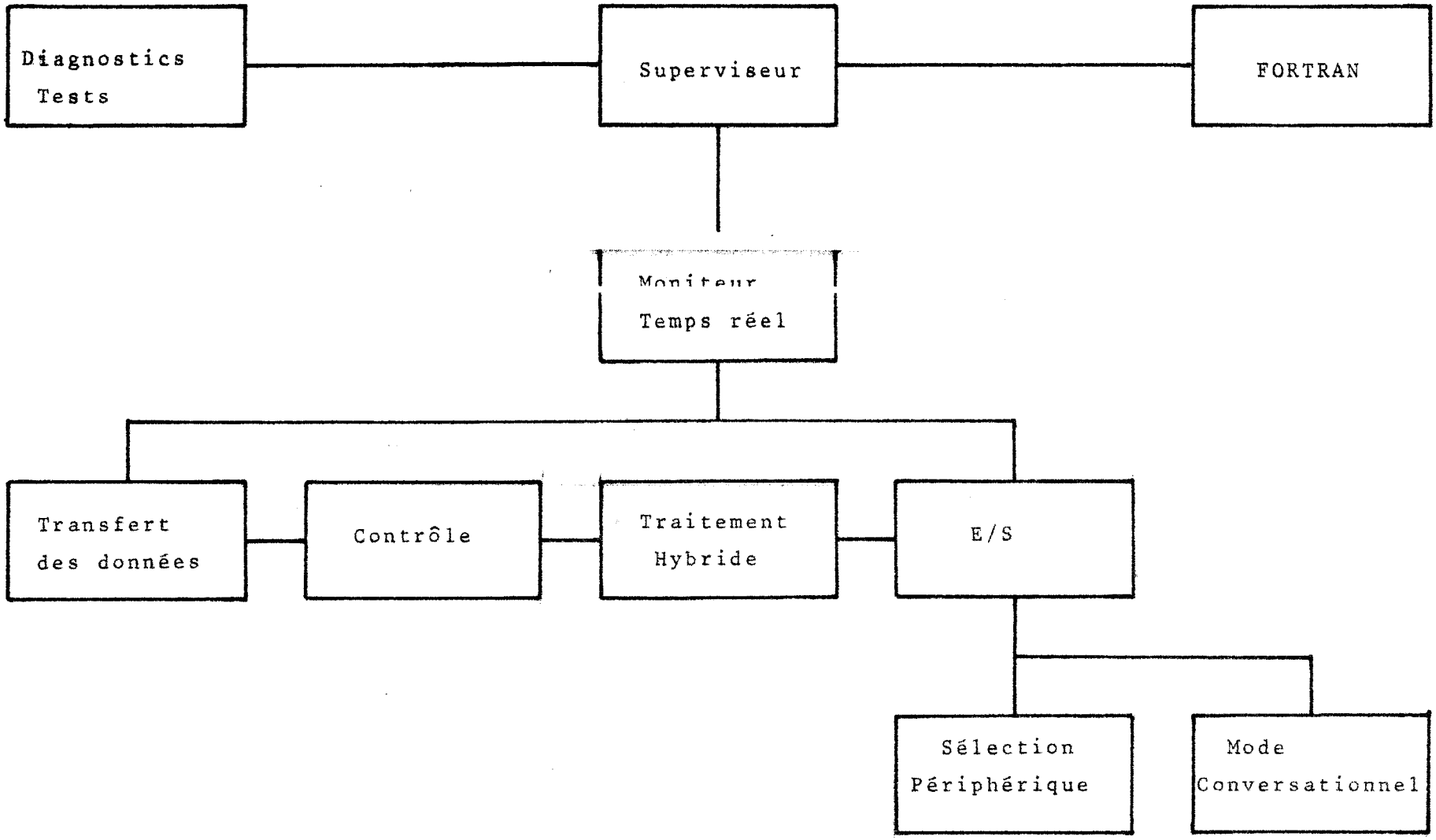


FIG. 1 - Hiérarchisation des programmes

contrepartie programmée du contrôleur cablé de l'interface. Cet ensemble de fonctions gère les possibilités d'entrées - sorties du calculateur sur une unité périphérique donnée et d'une manière plus importante, le traitement des erreurs avec une édition de messages.

Pour ce faire, les routines correspondant aux méthodes d'accès suivent lors d'un échange les quatre étapes définies précédemment. Les programmes de contrôle des erreurs interprêtent tous les paramètres cablés ainsi que les tests logiques et les erreurs de formats.

Lors de l'apparition d'une erreur pour une lecture par exemple, il est préférable de ne pas abandonner la tâche en cours en tentant une deuxième lecture ; dans le cas d'une nouvelle erreur, un message est édité pour en avertir l'opérateur et le travail est mis en attente de sa décision.

Pour une utilisation en temps réel, il est nécessaire que les méthodes d'accès traitent rapidement les différents problèmes puisqu'elles sont en série avec le programme utilisateur et le système cablé. Le programme de base devra donc être à ce niveau simple et ne traiter qu'un nombre restreint d'erreurs.

En général il y a un programme de méthode d'accès propre à chaque périphérique et l'ensemble des modules ainsi définis forme les méthodes d'accès proposées à l'utilisateur.

2.b - Les moniteurs

Chaque moniteur a un double but : d'une part fournir à l'utilisateur les programmes d'exploitation du système sans nécessiter une connaissance précise de l'interface d'autre part, régler l'enchaînement des programmes sans avoir à en connaître leur contenu.

En outre, on peut resserrer la liaison entre le hardware et le software en contrôlant précisément les erreurs de programmation, pour protéger le moniteur lui-même, aider à la programmation, isoler les programmes les uns des autres (cas du temps partagé) et enfin prévenir l'opérateur de l'état interne du système.

Nous pouvons représenter l'ensemble du système (Fig. 2) par trois zones : les programmes annexes (messages, lecteur, interprèteurs ...) les moniteurs et les programmes de traitement.

De plus, indépendamment de la manière dont se déroulent les opérations, les fonctions du système peuvent se classer en plusieurs catégories :

- traitement des travaux
- traitement des données
- traitement des tâches.

Le moniteur temps réel vient en aide au superviseur qui traite les liens entre les programmes utilisateurs. Il traitera plus spécialement les périphériques rapides et pourra s'intégrer dans le système comme programme séparé, c'est à dire résident faisant partie du système ou comme tâche particulière, introduit dans le flot des travaux. Son rôle est de traiter toutes les interruptions mais lorsqu'il n'est pas momentanément activé, les travaux se dérouleront normalement sous le contrôle du superviseur principal. On peut même supposer des travaux nécessitant une exécution en temps réel faisant appel à des tâches se déroulant hors de son contrôle (désactivation passagère).

Il permet de déterminer les priorités, l'initialisation et l'achèvement des travaux, et traite aussi les mises en attente, les temporisations, les conditions etc....

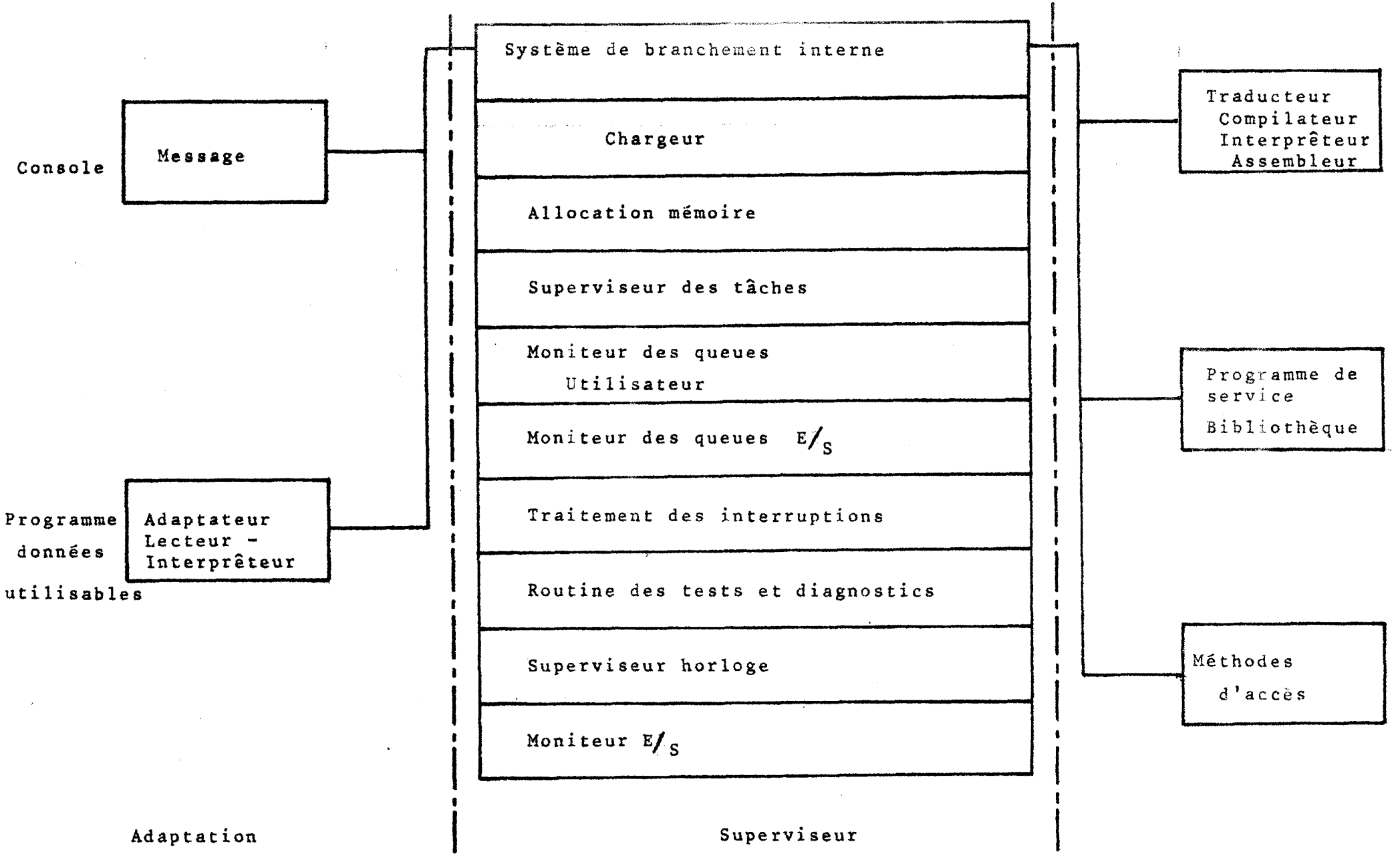


FIG. 2 - Le noyau du système

2.c - Programmes spécifiques aux unités physiques

Ces programmes étroitement liés aux méthodes d'accès apparaissent dans certains cas comme des moniteurs particuliers. Ce sont généralement des moniteurs non résidents et qui sont appelés du disque par le superviseur principal. On peut citer comme exemple le programme de traitement du Fortran qui ne génère pas un code directement exécutable mais fait appel à chaque étape à des modules spécialisés (compilateur, éditeur de lien, etc...).

Initialement, cet ensemble est le prolongement des méthodes d'accès : pour quelques périphériques particuliers, les programmes présents ne se réduisent qu'à des fonctions de lecture ou d'écriture, et aux quatre fonctions des méthodes d'accès. La liaison avec le programme utilisateur se fait alors à un niveau plus élevé que celui de l'assemblage.

2.d - Programmes de test

Ces programmes peuvent être utilisés pour l'entretien, et dans un contexte hybride, pour contrôler automatiquement la calculatrice analogique. Cet ensemble a pour but la mise en évidence de trois types d'opérations : simuler le sytème, détecter les erreurs et détecter la cause des erreurs.

Connaissant la composition exacte de l'interface, il est possible de déterminer les séquences de traitement permettant de détecter le ou les composants en défaut. Chaque élément peut être repéré par sa position physique représentée en mémoire par une matrice à trois dimensions. Le programme de test indique alors les numéros des circuits défectueux qu'il est aisé de resituer avec une grille.

2.e - Les traducteurs

Ils permettent de passer d'un langage haut niveau à l'assembleur ou le langage machine. Ils peuvent être utilisés pour la traduction instruction par instruction avec exécution simultanée ou non, soit en mode conversationnel.

On peut utiliser les traducteurs dans le cadre d'un système hybride, lorsqu'on opère en fortran et que les routines sont élaborées dans ce langage, notamment pour des fonctions telles que la sélection du mode analogique, la lecture du voltmètre digital, ou l'affichage des potentiomètres. Le traducteur interprète alors les instructions, une seule étant sous le contrôle de l'opérateur, ou traite la compilation du programme source en un seul tenant.

4 - METHODE DE PROGRAMMATION

Les calculateurs analogique et hybride sont couramment utilisés pour la simulation des systèmes continus. Chaque type de machine a ses qualités propres, la partie numérique ayant l'inconvenient d'opérer en séquence comme nous l'avons déjà souligné, et de permettre difficilement le dialogue homme machine, la machine analogique étant limitée en précision uniquement par la technologie employée. L'ensemble hybride essaye de combiner les avantages des deux machines. Nous pouvons considérer la partie analogique comme un périphérique du calculateur numérique, puisqu'on cherche à commander l'un par l'autre à partir de routines mémorisées dans la partie digitale, on le considère comme partie intégrante du système global. Nous essayerons par la suite de nous placer dans ce dernier cas.

4.1 - Programmation hybride

Une simulation par calculateur hybride peut se dérouler en cinq phases schématisées à l'aide de la table de la figure 3. La première étape est valable pour tous les types de simulation et quel que soit le système de calcul. La deuxième donne quelques indications sur la variété des décisions à prendre et la complexité qui peut en résulter dans le contexte d'un ensemble hybride, mettant à l'épreuve l'esprit de décision de l'utilisateur.

En effet, à ce niveau de la réalisation de la simulation, le programmeur aura à surmonter un certain nombre de difficultés :

a) il doit choisir sa solution parmi un grand nombre de possibilités et de combinaisons dues aux deux types de calculateurs. Autrement dit, il doit tenir compte des contraintes de précision nécessitées par sa propre étude et imposées par le calculateur et son interface.

b) il doit comprendre et traiter ses problèmes de synchronisation et de temporisation dans le cadre d'un fonctionnement en temps réel.

c) Il doit traiter les procédures nécessaires à la préparation de son travail d'une façon plus importante que s'il travaillait sur le calculateur numérique seul.

C'est ce troisième volet qu'il importera de réduire au maximum dans les routines d'aide à la programmation.

4.2 - Problème du coût

Les difficultés rencontrées lors de problèmes hybrides peuvent être examinées en terme d'efficacité en établissant un coût pour chaque phase du tableau I (fig. 3). C'est d'abord un coût en temps, celui de la machine (unité centrale) et celui passé par l'utilisateur pendant la phase de préparation. On peut donc chiffrer le prix global d'une étude à l'aide d'une simulation de la manière suivante :

$$C = \sum_{C=1}^{5} (P_{t_{pi}} + C_{t_{ci}}) \quad \text{pour les cinq phases,}$$

où P et C représentent le coût par unité de temps pour l'analyste et le calculateur et t_{pi} et t_{ci} représentent le temps passé sur le calculateur (t_{ci}) par la personne (t_{pi}) lors de la phase i.

Une telle évaluation peut permettre de déterminer l'efficacité de n'importe quel type de système, mais définir la précision du calcul peut présenter certaines difficultés. L'expérience a cependant montré que c'est le coût correspondant aux phases deux (préparation du problème) et trois (résolution - aide à la programmation...) qui fixe le type de solution à employer. De plus, il arrive souvent que la mise en route et le traitement des erreurs prennent plus de temps que la résolution du problème lui-même, en particulier dans le cas d'un grand nombre de travaux, chacun n'effectuant qu'une petite simulation.

Nous pouvons alors essayer de définir les caractéristiques générales d'un "bon système" et les influences sur le coût global (fig. 4).

Nous devons éviter, dans la mesure du possible, de ne traiter qu'une classe de problème, de ne proposer qu'un ensemble incomplet des routines nécessaires à un système efficace, ou de limiter la liberté du programmeur pour développer d'autres configurations, de demander à l'utilisateur plus d'efforts qu'il n'est nécessaire.

Ce sont des caractéristiques qu'il est souhaitable d'obtenir d'un système, mais que l'on ne rencontre toutes réunies dans aucun ensemble. Nous ne pensons pas faire mieux que d'autres, et notre ensemble hybride sera nécessairement limité, mais notre souci majeur est de permettre une solution aisée de problèmes de type courant, et d'autoriser une évolution au fur et à mesure que se développeront ses possibilités.

5 - CONCLUSION

Aucun système hybride existant ne présente réunies toutes les qualités décrites précédemment et, en ce sens, n'est un "bon système". Cependant, l'évolution générale laisse prévoir des langages spécialisés de plus en plus performant ainsi qu'une certaine normalisation.

Les différents systèmes s'étoffent sensiblement, la partie software devenant plus importante avec l'accroissement du nombre de possibilités offertes ; certains fonctions s'exécutant autrefois par software se font actuellement au niveau du hardware, ce qui rend plus important le rôle de l'interface.

Après la description des caractéristiques essentielles d'un système hybride, il est possible de déterminer le système de base de notre ensemble. Celui-ci, dans son état actuel, ne présente évidemment pas toutes les caractéristiques souhaitables mais nous nous sommes efforcés, lors de son élaboration de permettre un développement ultérieur. Dans les chapitres qui suivent, nous décrivons ce système et son application à différents exemples.

TABLE I

Différentes phases de résolution d'un problème à partir d'un système hybride.

Ceci ne représente qu'un essai de classification et la structure des différentes phases n'a rien d'impératif.

1 - DEFINITION DU PROBLEME
- a) Equations du système
- b) Paramètres variant lors de la simulation
- c) Procédure correspondant à la solution générale
- d) Répartition des tâches analogiques et digitales.

2 - PREPARATION HORS LIGNE
- a) Mise à l'échelle des équations
- b) Détermination des valeurs par le contrôle statique
- c) Allocation des éléments sur le panneau analogique
- d) Détermination des liaisons avec l'interface
- e) Développement des programmes numériques.

3 - PREPARATION EN LIGNE
- a) Cablage
- b) Chargement des programmes sur l'ordinateur
- c) Contrôle statique et dynamique des composants de l'interface
- d) Entrée des paramètres pour l'établissement des valeurs de potentiomètres.

4 - DEROULEMENT DE LA SIMULATION
- a) Evolution des paramètres
- b) Diagnostics

5 - RESULTATS
- a) Evaluation du modèle
- b) Détermination du procédé de calcul
- c) Détermination du modèle ou retour à la phase 2 ou 3 pour évaluation plus fine.

FIG. 3

TABLE 2

Efficacité du langage	: possibilité de représenter un problème hybride dans un langage évolué ou assembleur simultanément ou indépendamment.
Aptitude	: possibilité de générer des macro operations ou des macro-instructions.
Diagnostic	: ensemble de détection des erreurs et leur édition que ce soit au niveau numérique ou analogique
Documentation	: liste des programmes d'utilisation générale
Simulation digitale	: possibilité de faire numériquement une simulation de certaines parties du problème résolu analogiquement.
Allocation mémoire	: possibilité de transfert de programme en mémoire et plus généralement possibilité d'allouer dynamiquement les ressources des calculateurs.
Mise à l'échelle	: Mise à l'échelle des variables analogiques et temporelles.

Quelques caractéristiques souhaitables d'un système hybride

FIG. 4

SYSTEME DE COMMANDE

Le système des commandes est l'ensemble des programmes qui permet l'exploitation de l'ensemble hybride. Il comprend donc le moniteur qui dirige l'exécution des différents travaux présents en mémoire à un instant donné et les programmes relatifs au sous ensemble de liaison analogique logique.

Nous avons tout d'abord mis au point un moniteur temps réel en assembleur qui gère des travaux directement exécutables. En d'autres termes, les travaux sont entrés directement en mémoire, à partir d'un support externe, ruban, disque etc..., sous forme binaire. Dans une deuxième étape, le moniteur est considéré comme une tâche particulière vis à vis du superviseur et ne traite que ce qui concerne les entrées - sorties spécifiques au couplage, ainsi que certaines fonctions spécialisées, tout en permettant l'écriture des programmes en Fortran.

1 - LE MONITEUR

1.1. - Rôle du Moniteur

Le Moniteur doit pouvoir assurer les fonctions suivantes :

1.a - Chargement du système

Le noyau du système étant en mémoire charge lui-même les modules dont il a besoin pour son initialisation, et au fur et à mesure des demandes de l'opérateur, les fonctions nécessaires au déroulement des différentes étapes des travaux. Il doit en outre gérer le traitement des divers fichiers sources.

1.b - Sauvegarde des programmes

L'utilisateur peut demander à tout moment de préserver le code objet de son programme en le mettant en mémoire de masse, ou en demander la restitution à partir de fichiers déterminés.

1.c - Gestion de la bibliothèque

Une fois un programme mis au point, l'opérateur a la possibilité de le sauvegarder en bibliothèque. Ce traitement ne se fait plus sur un fichier commun à tous les utilisateurs, donc susceptible d'être effacé après utilisation mais sur un fichier réservé au système. Autrement dit, un programme stocké en bibliothèque est réutilisable en permanence, alors que le même programme stocké différemment ne sera conservé que dans la mesure où l'on est sous le contrôle du moniteur. La structure de l'ensemble est définie par le schéma suivant (fig. 1).

1.d - Modification en mémoire

L'utilisation de l'assembleur suppose la connaissance à tout moment des adresses absolues du programme ; dans le cas où celui-ci a été translaté à la suite d'une compression

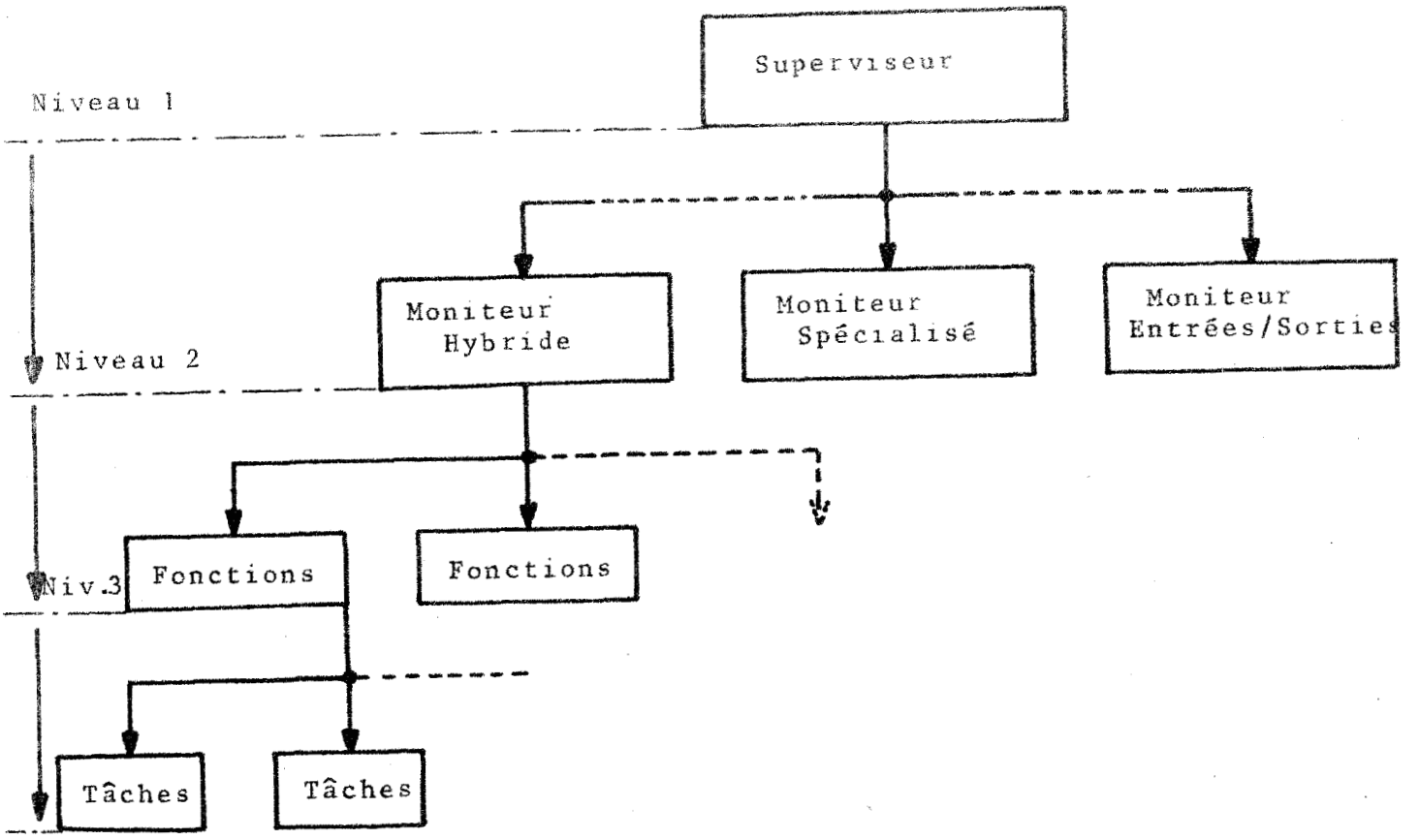


Fig. 1

Les différents niveaux du système

due à la suppression d'un travail, il est possible, à partir de la console d'obtenir l'état des travaux, (un travail "actif" ne peut être translaté) et la localisation de ceux-ci.

I.e - Gestion de la mémoire

Le moniteur a pour rôle de contrôler les limites de la mémoire utilisée par les différentes routines de traitement, afin de permettre à un programme en temps réel ou un programme de mise au point de coexister en mémoire avec d'autres.

Il n'est en effet pas concevable que, par inadvertance, on puisse charger un programme objet issu d'une compilation ou d'un assemblage sur les emplacements d'un autre programme.

1.f - Gestion des travaux

Sans entrer dès maintenant dans les détails de la gestion des travaux, nous pouvons dire que le moniteur, sur l'ordre de l'opérateur, initialise, exécute ou interrompt un travail, quelque soit son état.

1.g - Mode conversationnel

L'opérateur peut demander en permanence l'état des travaux pour modifier le cas échéant les priorités, supprimer certaines tâches, ou en activer d'autres. Il doit d'autre part être tenu au courant des erreurs d'exécution, des alarmes, grâce à l'émission de diagnostics indiquant le type d'erreurs, le numéro du travail l'ayant provoqué et la cause de l'erreur, par localisation de l'instruction qui en est l'origine.

Certaines erreurs entraînent l'arrêt définitif du travail, d'autres moins graves sont "rattrapées" par le moniteur sans notifications particulières.

1.2 - Description du moniteur

Le moniteur gère l'exécution de 18 travaux se déroulant en parallèle. A chaque travail sont associées une priorité et une zone d'implantation mémoire. Le moniteur utilise les temps morts lors des entrées - sorties avec les

périphériques lents pour activer les différentes tâches. D'autre part, certaines fonctions spécifiques au moniteur sont traitées comme des programmes ordinaires,mais ont une priorité fixée une fois pour toute, généralement haute, leur permettant de modifier l'état du système.

2.a - Notion de travaux

Un travail est caractérisé par quatre états à priori différents : actif, inactif, inhibé, en attente (d'un périphérique, d'une fin d'échange, d'une condition etc...).

- Travail actif

A un instant donné, un travail et un seul est actif, c'est à dire qu'il occupe alors l'unité centrale en développant une série d'instructions (calculs, tabulation etc...) qui ne nécessitent aucun périphérique. Ce travail peut d'ailleurs être un des modules spécifiques au moniteur.

- Travail inactif

Un travail inactif occupe de la place en mémoire et peut être activé à tout moment sur demande de l'opérateur par exemple, ou d'un autre travail. Sa présence est connue du moniteur en ce sens que ses caractéristiques sont définies (priorité, emplacement mémoire.....) et qu'il lui est donc impossible d'implanter un autre programme sur sa zone mémoire, qui est considérée comme "gelée".

- Travail inhibé

Tout travail peut être inhibé quelque soit son état. Dans ce cas, toute demande ou tout appel à ce travail ne sera pas satisfait et provoquera l'inhibition du travail demandeur, le déroulement normal de celui-ci étant perturbé. Il est impossible d'inhiber une tâche du moniteur. Un travail

inhibé est gelé dans l'état précédent l'inhibition et sa validation permettra son déroulement à partir de cet état.

- Travail en attente

Lors d'un échange avec un périphérique, il existe nécessairement un temps de latence mis à profit par le calculateur pour exécuter d'autres tâches ou lancer des échanges sur d'autres périphériques. Une fois l'échange satisfait, le travail repart de son point d'interruption. Tout se passe comme si le programme avait été momentanément inhibé, la fin de l'échange sur le périphérique permettant sa validation.

- Abandon

Un travail peut prendre un autre état momentané. Lorsqu'une tâche provoque une alarme, elle est abandonnée, elle passe à l'état inactif puis est supprimée. Le moniteur libère alors les périphériques qu'elle utilise, permettant une remise à jour des affectations. La zone mémoire correspondante pouvant être réalouée, le moniteur ne tient plus compte de sa présence et ne la sauvegardera pas, et le travail en dépendant devient alors inexistant.

Les différents états que peut prendre un travail sont représentés par le schéma de la figure 2. Le travail le plus prioritaire est le travail de priorité 1 (partie du moniteur), le moins important a la priorité 18 (travail de fond).

A un instant donné, il existe toujours un travail activé. Si plusieurs travaux sont en attente et que l'unité centrale est libre, le moniteur provoque l'exécution d'un travail de fond, qui peut être une boucle sans fin. Ce travail ne peut être inhibé et a pour but d'éviter l'arrêt total du calculateur.

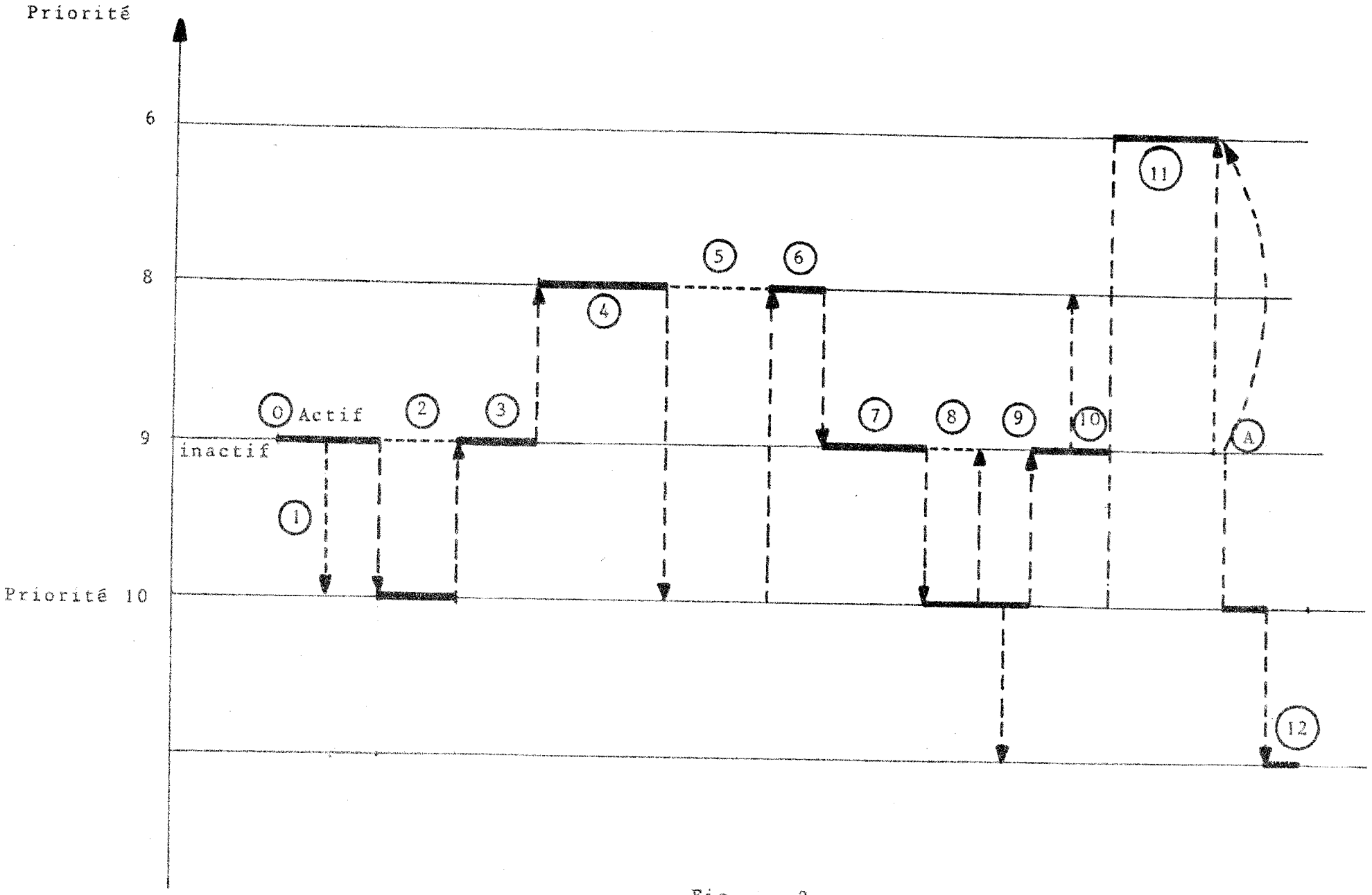


Fig. 2

Les différents états des travaux

Légende de la Figure 2

Exemple de fonctionnement.

0 Point de départ activation du travail 9.

1 Le travail 9 demande le travail 10.

2 Attente de 9 (Entrée - sortie) et déroulement de 10 pendant ce temps.

3 Interruption de fin d'échange - reprise de 9, inhibition de 10 par le moniteur.

4 9 demande 8 avec condition de retour - attente de 9

5 8 en attente d'Entrée - sortie. Comme 9 est en attente de 8 on a l'exécution de 10.

6 Interruption E/S - Exécution de 8.

7 La condition qu'attend 9 est réalisée. 8 est terminé donc 9 reprend son exécution (validation).

8 9 est en attente d'E/S. Pendant cette attente 10 se déroule et inhibe 9.

9 L'interruption de fin d'échange de 9 est apparue mais inhibé, il ne peut s'exécuter.

10 Validation par 10 de 9. Demande de 8.

11 8 demande 6 qui travaille anormalement. Abandon de 6 et de ses satellites. En cascade, on abandonne les travaux demandeurs, 8 et 9.

12 Il ne reste que 10 en cours.

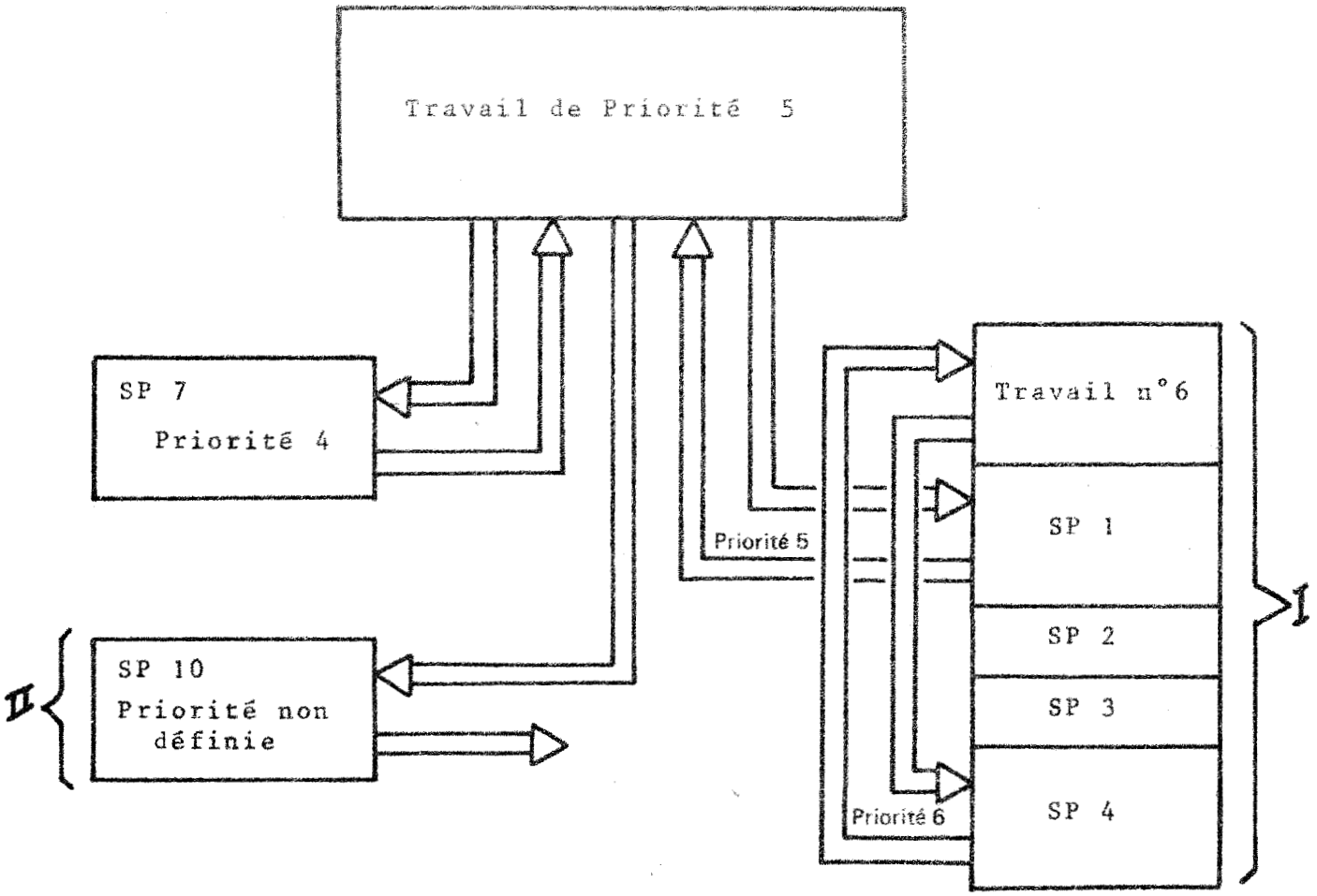


Fig. 3

Relation priorité - programmes

Cas I : SP 1 est "banalisé"
Il a la priorité du travail appelant (n° 5)

S'il fait des entrée - sortie, celles-ci seront affectées au travail 5 : il est donc impossible dans ce cas de faire appel à SP1 à partir d'un travail de priorité différente de 5.

Le cas est le même avec SP4 et le travail 6.

Cas II : SP 7 est "personnalisé" par sa priorité 4.

S'il fait des entrées - sorties, les périphériques lui seront affectés, donc il est utilisable pour tous les travaux (à condition de réentrance).

Il faut prévoir probablement des synchronisateurs programmés dans le cas où pendant un échange de SP 7 le moniteur reprendrait l'exécution du programme appelant.

SP 10 est inactif et n'ayant pas de priorité définie (travail abandonné ou terminé), il peut être supprimé de la mémoire.

2.b - Organisation dans la gestion des travaux

Chaque travail est repéré par son niveau de priorité, aucune ne pouvant être attribuée simultanément à plusieurs travaux.

Ceci peut présenter un inconvénient lors d'un changement de priorité puisqu'alors le travail change de nom, mais cette opération n'étant réservée qu'à l'opérateur, celui-ci peut très bien suivre l'évolution des travaux.

- Gestions des priorités

La priorité porte sur un travail consistant en un certain nombre de sous programmes spécialisés (programmes de calculs, opérations particulières, etc...). Ces tâches peuvent ne pas recevoir de priorité en elle-même, à condition d'avoir été répertoriées sous un nom de travail, afin d'éviter leur effacement. Des cas particuliers sont envisagés dans la figure 3.

- Gestion des périphériques

La succession des entrées - sorties sur les différents périphériques nécessite une gestion afin d'optimiser les échanges. D'autre part, afin d'éviter les interférences entre travaux de priorités différentes, le moniteur doit gérer le fonctionnement de tous les périphériques.

Afin d'augmenter la rapidité, il importe que tous les échanges se fassent en mode prioritaire.

Une fois activé, tout programme peut faire une demande de périphérique au moniteur. Celui-ci décide alors s'il peut le lui accorder suivant l'état d'occupation de l'unité. Si ce périphérique est occupé, la demande est rejetée, le travail est mis alors dans la file d'attente du péri-

phérique. Il est inhibé momentanément jusqu'à sa validation qui n'interviendra qu'à la libération du périphérique et suivant la priorité du travail. En effet, si un périphérique est affecté à un travail, il est considéré comme occupé pour toute autre demande, et ne sera libéré qu'à la fin du travail l'utilisant ou sur abandon.

Nous avons alors l'organigramme de la figure 4 :

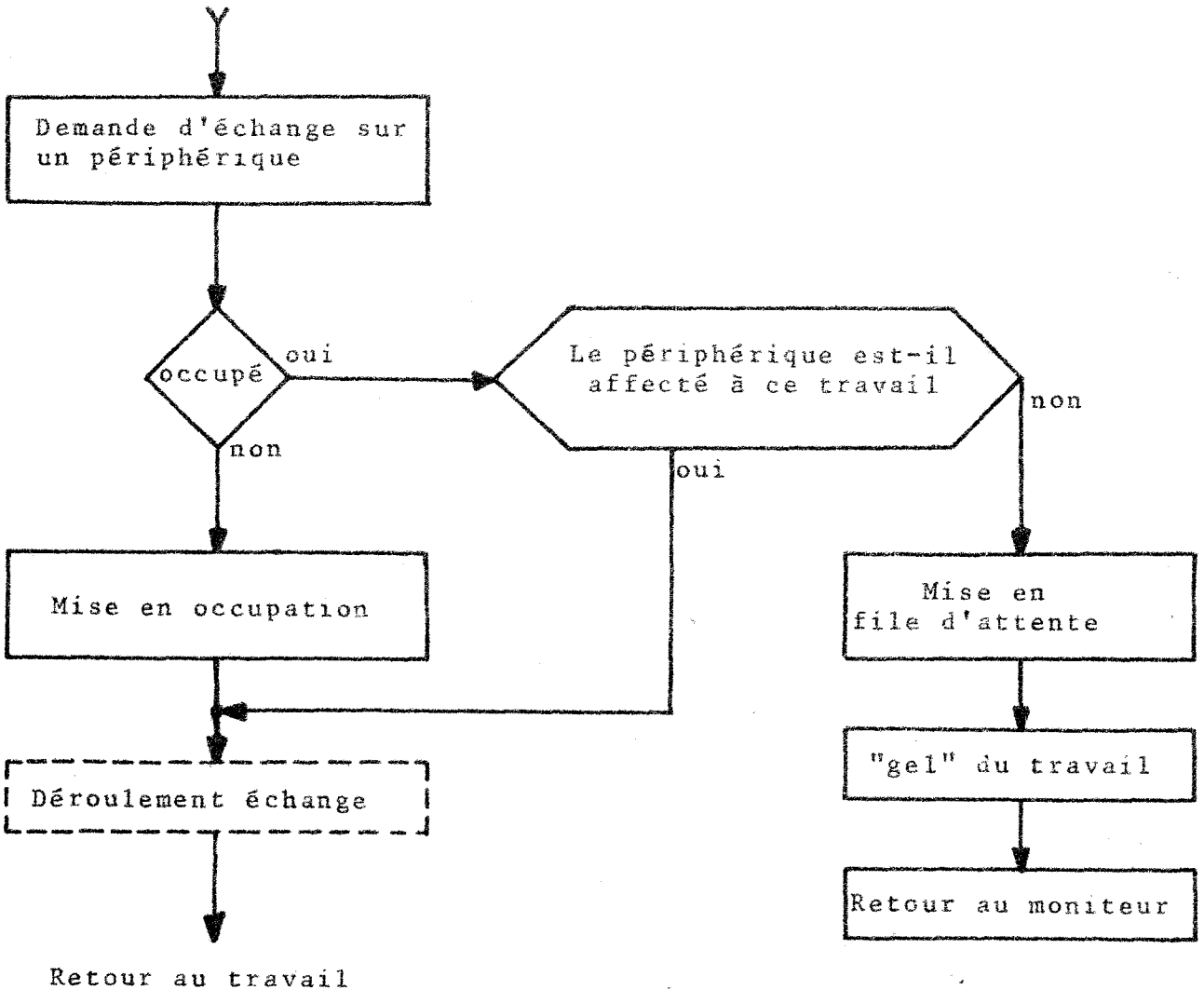


Fig. 4

Demande d'échange sur un périphérique

Pour permettre une amélioration des performances du système, il est possible de se "défaire" d'un périphérique avant la fin normale du travail. En étudiant la file d'attente du périphérique, le moniteur remet en circuit une tâche suffisamment prioritaire , l'opération se déroulant suivant la figure 5 :

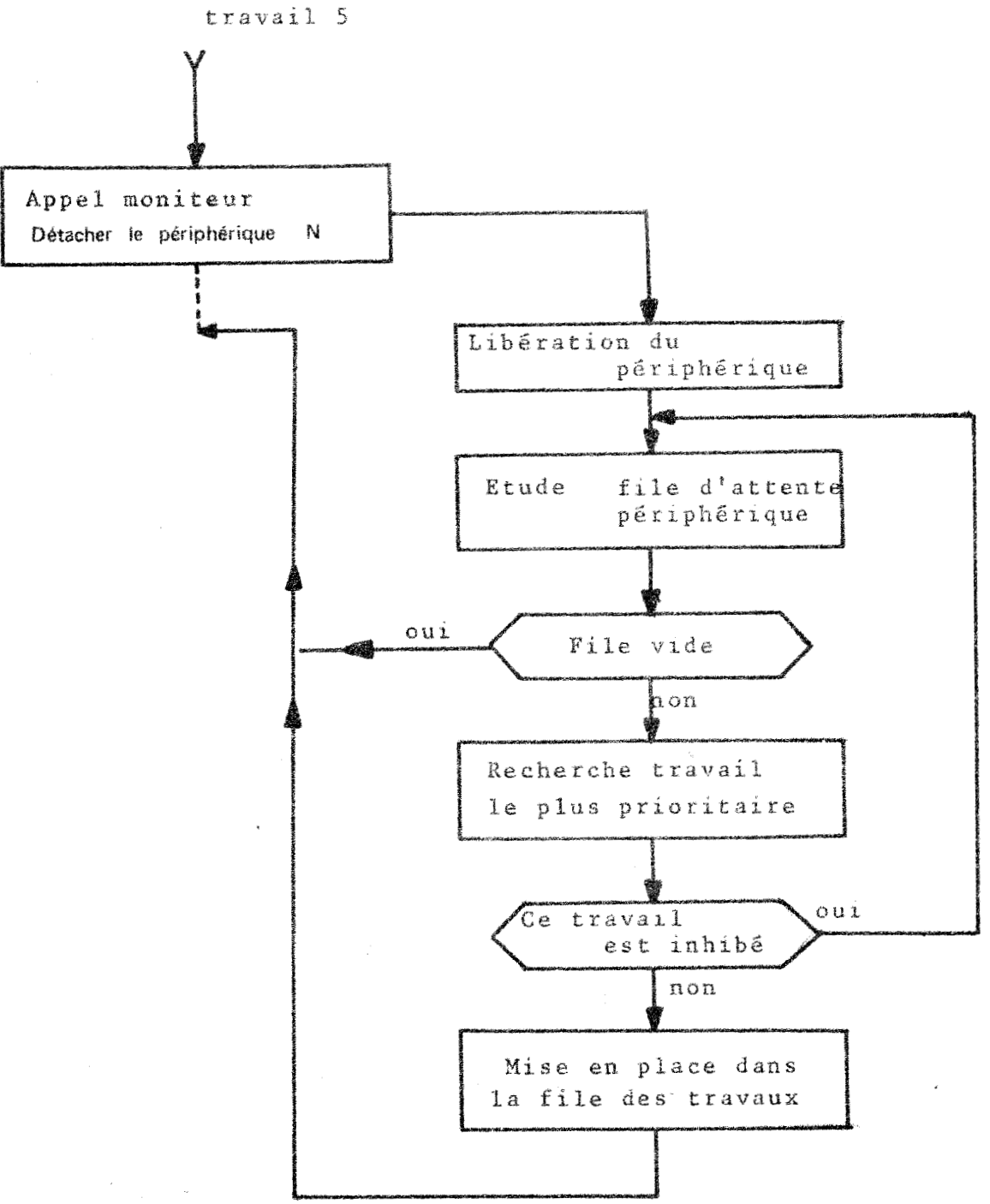


Fig. 5

Libération d'un périphérique

Comme nous pouvons le voir sur cet exemple, il existe des files d'attente à plusieurs degrés. D'une part au niveau du périphérique, et d'autre part au niveau des travaux (file d'attente générale). Dans cette dernière ne figurent que les travaux normaux, c'est à dire non inhibés ou en cours d'échange avec l'extérieur. Le fait de libérer un périphérique permet l'insertion d'un travail dans cette file, sans toutefois entraîner son exécution immédiate (Figure 6).

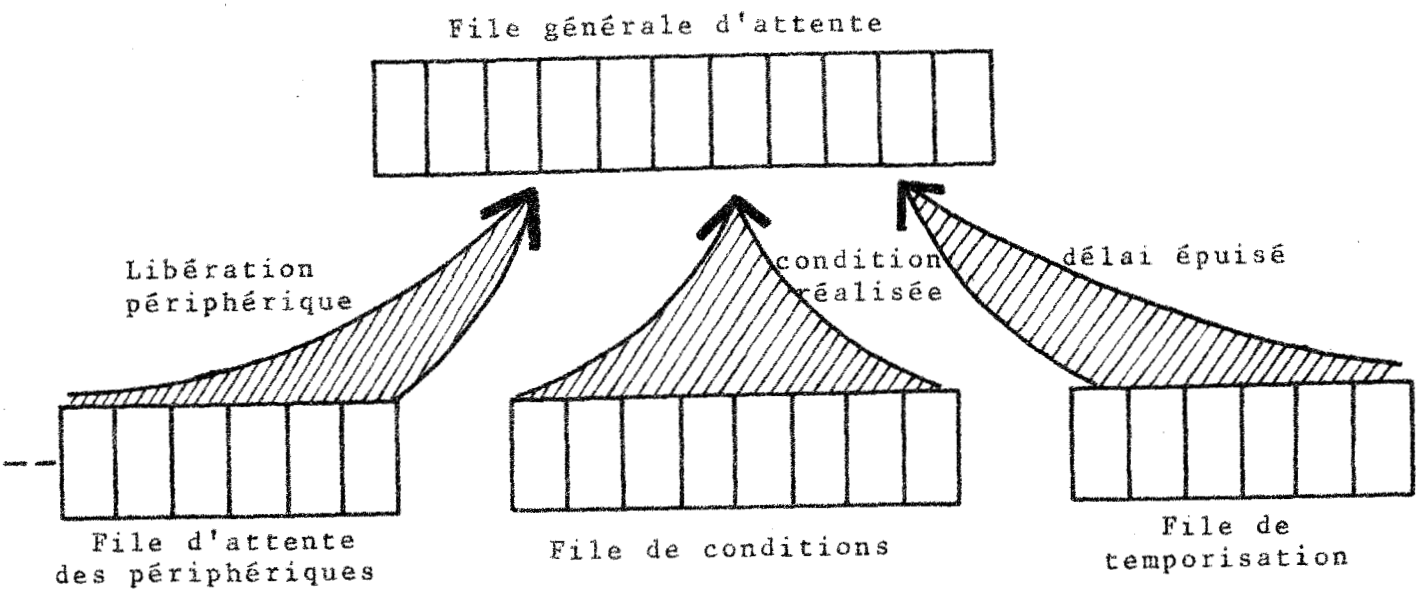


Fig. 6

Les files d'attentes

Une solution permettant d'optimiser l'utilisation des périphériques consiste à "moduler" leur emploi en incluant les échanges dans des sous programmes spécialisés, considérés comme des travaux à part entière. L'achèvement de l'échange correspond à la fin du travail et provoque la libération du périphérique (Figure VI.7).

Il est nécessaire dans ce cas de choisir judicieusement la priorité à accorder à ces sous programmes, pour éviter un trop grand nombre de conflits lors de l'enchaînement des travaux de priorités différentes.

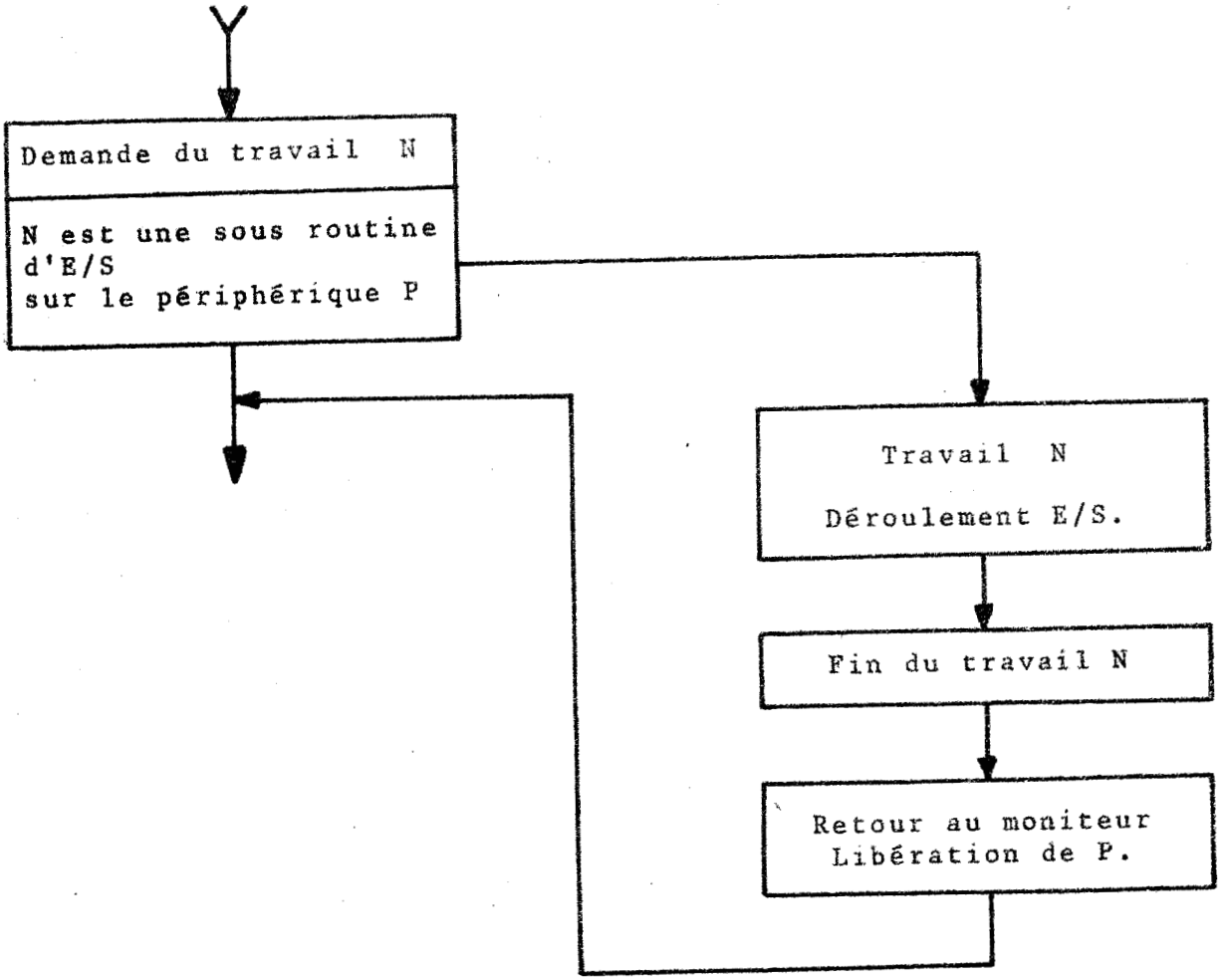


Fig. 7

Sous programmes d'Entrées - Sorties

Cette méthode présente un intérêt lorsque le nombre d'entrées sorties sur un périphérique est relativement peu élevé.

En résumé, nous avons trois méthodes d'utilisation, chacune différant par le temps d'occupation théorique du périphérique :

1) Utilisation parallèle au travail. L'unité n'est libre qu'à la fin du travail.

2) Utilisation jusqu'à la libération sur l'initiative du travail.

3) Demande et libération au rythme de l'échange.

Ce que nous représentons par les diagrammes de la figure 8

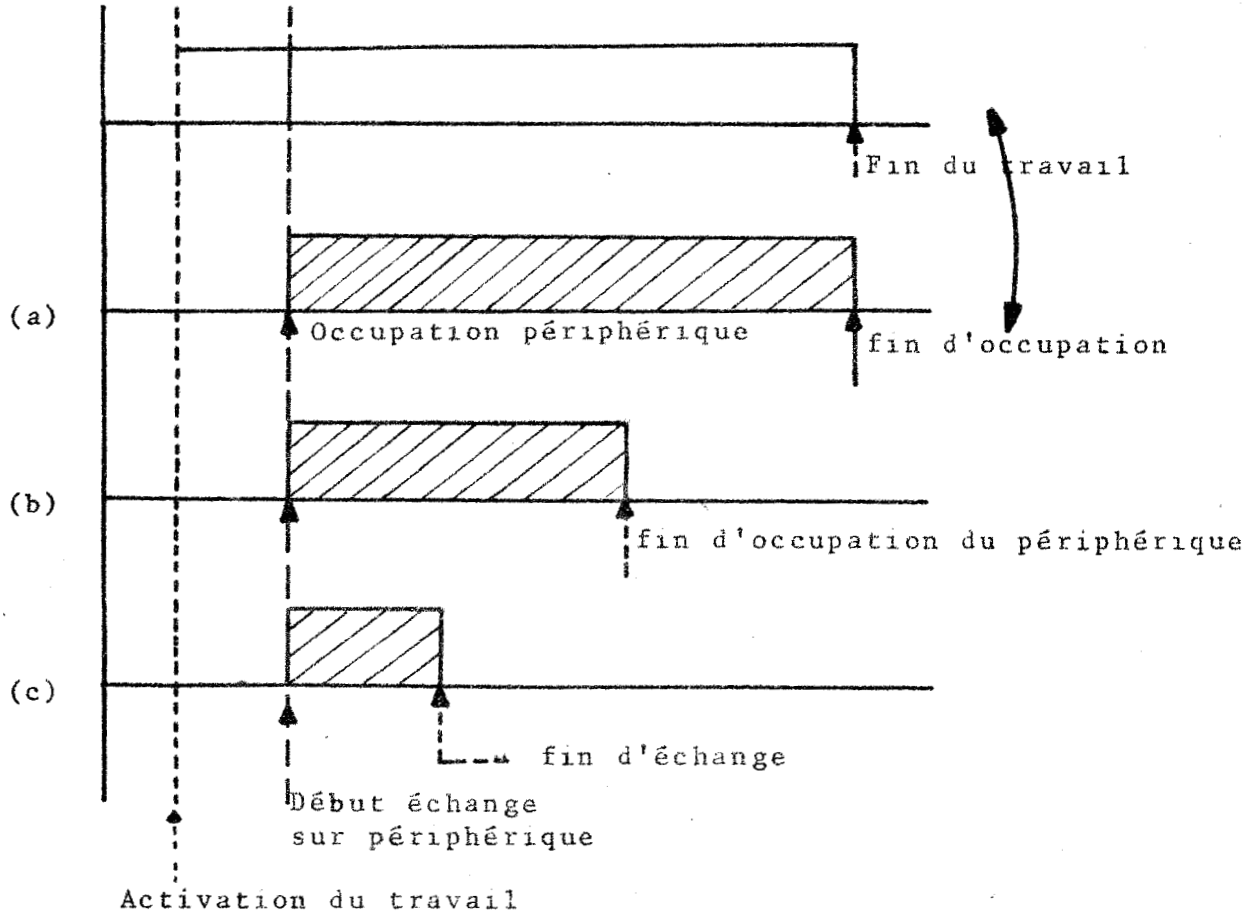


Fig. 8

Temps d'occupation des périphériques

2.c - Fonctions du moniteur

- Articulation des programmes

Pour une bonne gestion de la mémoire, tout travail est déclaré au moniteur avec son adresse d'implantation initiale et suit certaines régles permettant de gérer au mieux les échanges avec l'extérieur. Toute tâche est donc ponctuée d'appels au moniteur, à son début (insertion dans la fournée des travaux, création de tables.....) et à sa fin (libération des périphériques utilisés.....) suivant le schéma de la Figure 9.

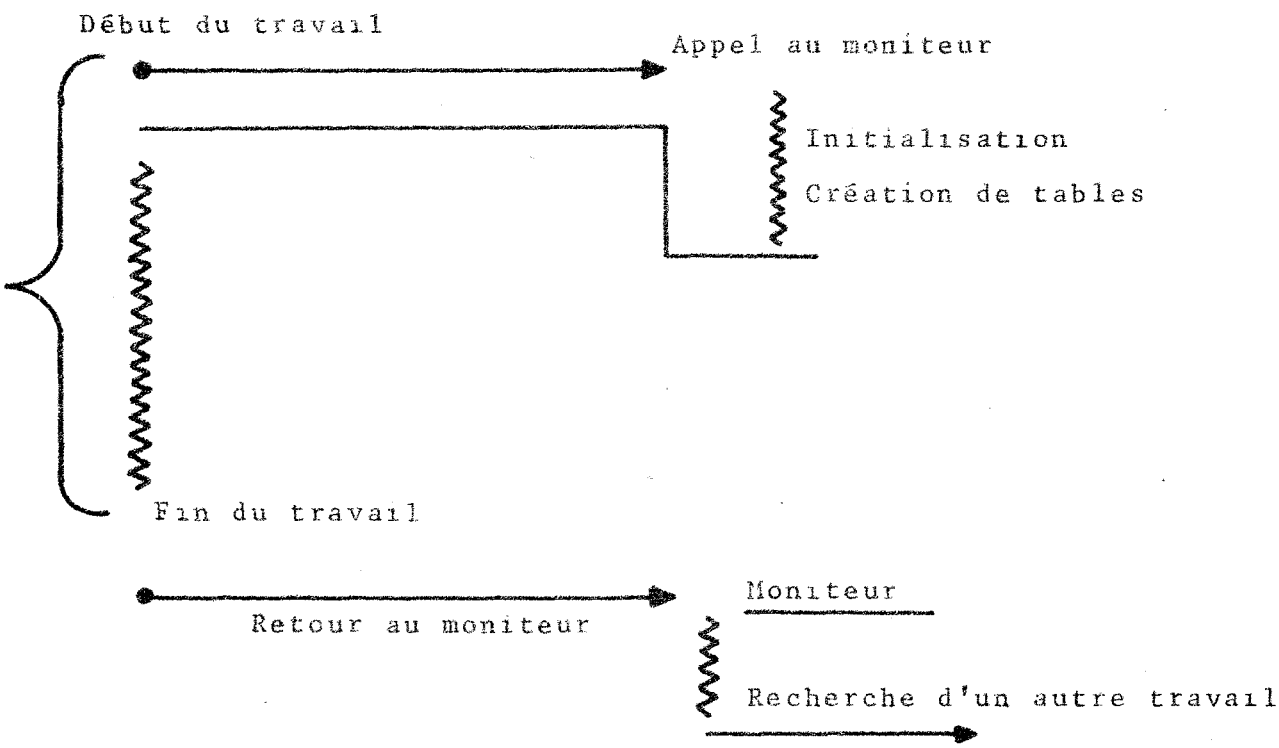


Fig. 9

Appel au moniteur : début et fin de travail

D'autre part, afin de faciliter la programmation et permettre à l'utilisateur l'organisation de son travail, un certain nombre de fonctions sont utilisables par appel au moniteur.

Nous avons étudié un exemple où il était possible de contrôler plus étroitement le déroulement des opérations. Il existe d'autres cas où une telle nécessité peut se faire sentir, pour synchroniser notamment certaines tâches en utilisant les modules de temporisation (traitement des retards, délais) ou de condition (validation d'un travail si et seulement si une ou plusieurs conditions sont réalisées). Ces fonctions sont schématisées figure 10.

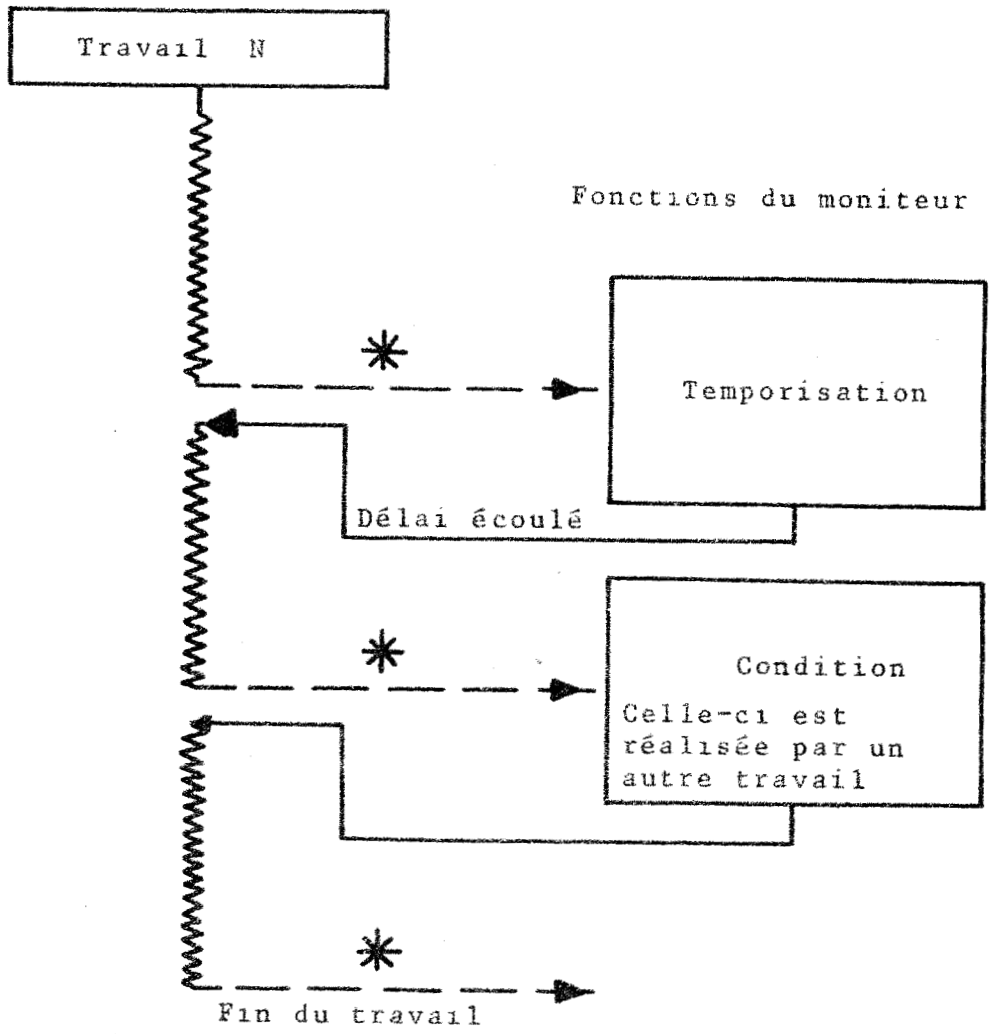


Fig. 10

Travail et fonctions du moniteur

Il existe une possibilité de segmentation au niveau de l'utilisateur par appel au module spécialisé correspondant. Cela nécessite des contrôles très stricts afin d'éviter les perturbations et de permettre un bon cloisonnement des travaux (figure 11).

L'utilisateur peut prévoir une articulation plus ou moins complexe des programmes et des tâches nécessaire à la résolution de son problème. Diverses structures s'offrent alors :

. Une structure simple

. Une structure avec recouvrement des parties de programmes définies à priori

. Une structure dynamique séquentielle

. Une structure dynamique avec déroulement parallèle des tâches.

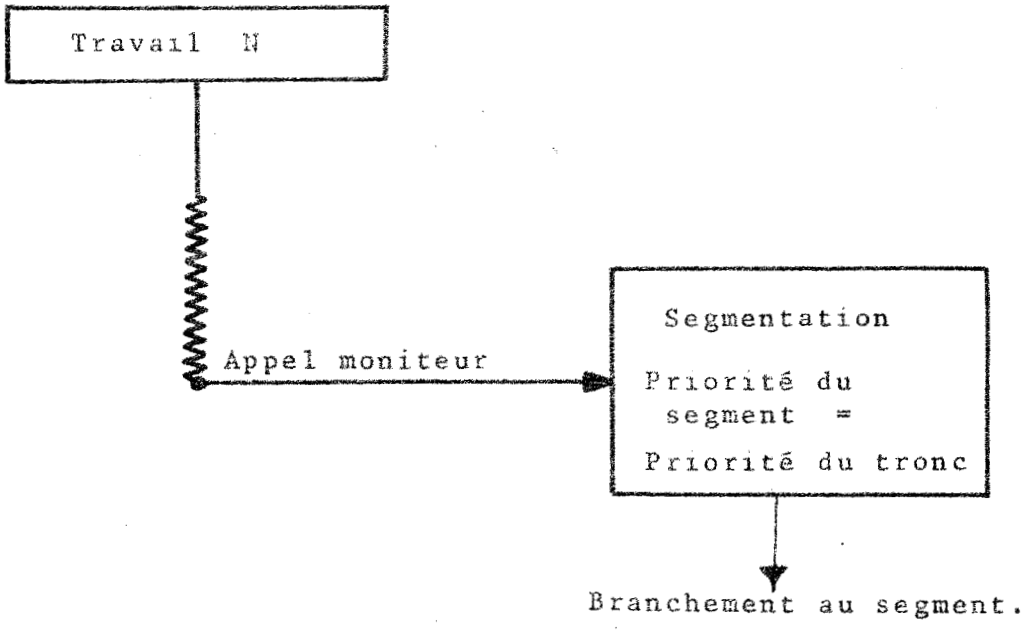


Fig. 11

Segmentation

- Structure simple

Un programme a une structure simple s'il ne nécessite le chargement que d'un seul module. La création d'une tâche consiste à le placer en mémoire et à lui donner le contrôle. L'ensemble représente un travail d'un seul tenant, repérable par le moniteur à l'aide de sa priorité et défini en mémoire par son adresse d'implantation fournie lors de l'initialisation. Ceci n'interdit pas la présence de sous programmes et l'allure générale du travail est décrite dans la figure 12.

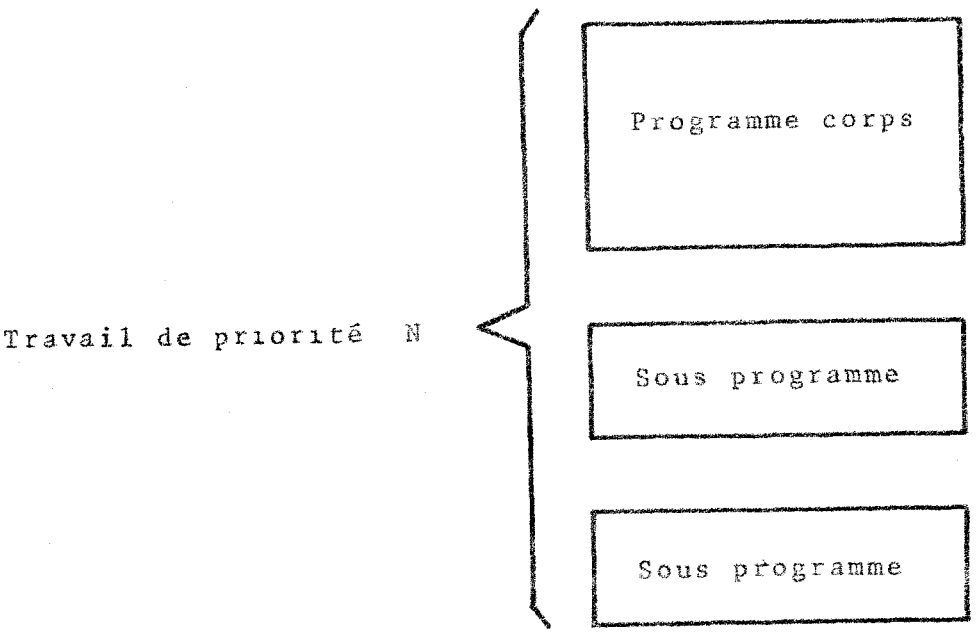


Fig. 12

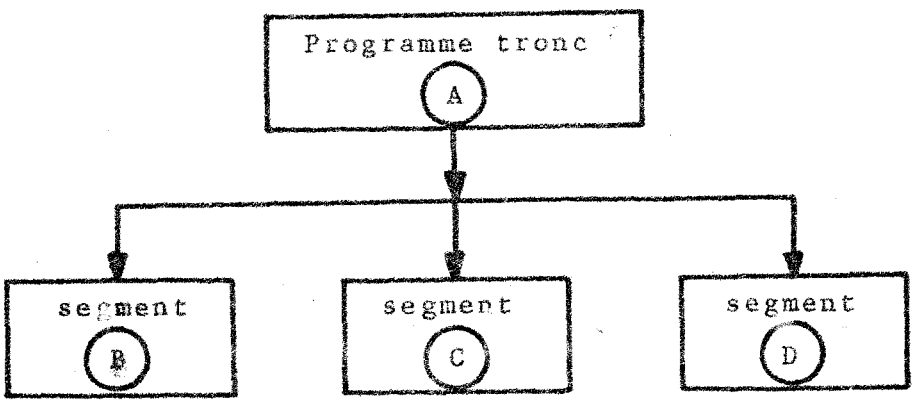
Structure d'un programme.

Si les sous programmes sont réentrants, il est alors possible de les "banaliser", c'est à dire de permettre leur utilisation par des tâches ou des sous tâches correspondant à des travaux de priorités différentes.

- Structure avec recouvrement à priori
  Overlay

L'utilisation de l'organisation simple décrite précédemment est limitée par la taille de la mémoire vive disponible pour le chargement du programme, d'où l'idée de ne charger que partiellement le programme et de recouvrir les parties inutilisées à un instant donné.

Un programme décomposé en parties indépendantes se prête bien à une telle organisation. Le code présent en mémoire étant directement exécutable, il est nécessaire de prévenir le moniteur de ce mode de travail afin d'éviter lors d'un tassement de mémoire, le transfert intempestif d'un segment.



Les segments B, C, D occupent le même emplacement mémoire.
Le moniteur charge les programmes B, C ou D à partir de la bibliothèque en les repérant par leur nom.
Le transfert se fait généralement en parallèle avec le programme demandeur ; il est donc nécessaire de prévoir une attente dans le cas d'utilisation immédiate du segment.

Fig. 13

Structure en Overlay

Les références d'un segment vers la racine seront immédiates mais celles de segment à segment n'est pas possible. D'autre part, les changements fréquents de segments dégradent les performances du programme.

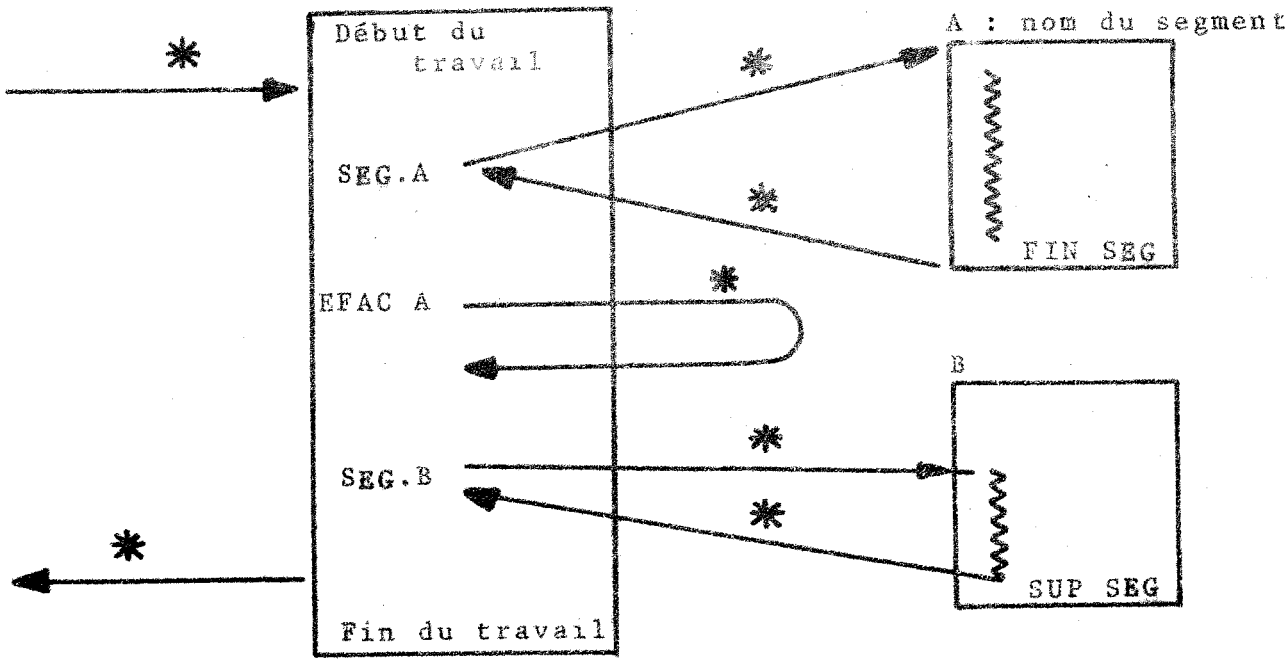
- Structure dynamique séquentielle

L'organisation précédente impose à l'utilisateur de connaître exactement l'ordre dans lequel vont s'enchaîner les diverses parties de son programme. L'impossibilité de charger deux segments exclusifs d'une structure de recouvrement peut aussi être une gêne, d'où l'idée de charger dynamiquement les parties de programmes lors de leur appel. Les parties sont alors des modules chargeables indépendants. Ceci est possible grâce à certaines conventions du système d'exploitation.

. Un module peut être repéré par son nom et son point d'entrée dans une bibliothèque.

. La place en mémoire est gérée dynamiquement par le moniteur.

L'utilisateur gère alors ces transferts par l'intermédiaire du moniteur. D'une façon générale, tous les appels au moniteur se font à l'aide de macro-opérations. Il est possible de demander la libération mémoire, après exécution, ou de garder le module jusqu'à recouvrement décidé lors d'un appel d'un autre module : figure 14.



✳ Intervention du superviseur

A la fin du travail, tous les segments sont supprimés.

SEG, EFAC, SUPSEG sont des MACRO opérations servant à rechercher le module A, l'effacer dans le programme corps (EFAC), en retournant dans le programme corps avec l'effacement automatique (SUP SEG).

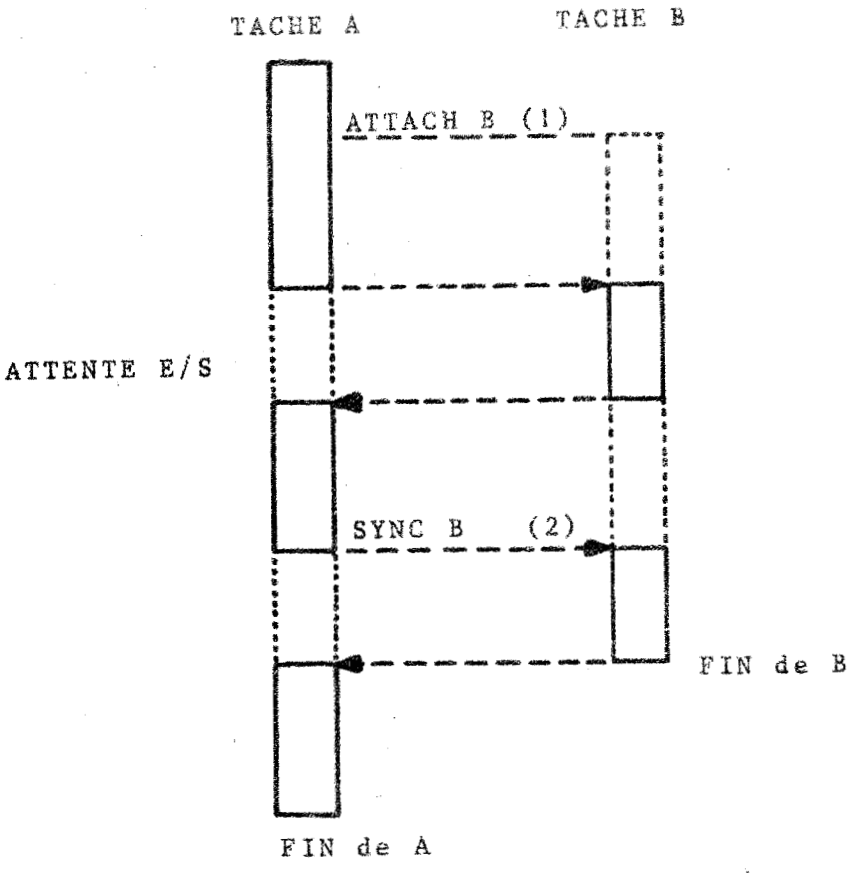
Fig. 14

Structure dynamique parallèle.

- Structure dynamique parallèle

Jusqu'à présent, le transfert de contrôle d'un module à l'autre fait qu'à un instant donné, un seul module peut s'exécuter. Dans le système susceptible de gérer en mémoire plusieurs tâches, un module peut demander la création d'une autre tâche pour résoudre une partie du problème à traiter. Cette tâche pourra s'exécuter lorsque le premier sera en attente (entrée - sortie par exemple), on a alors une structure dynamique parallèle : la tâche mère crée une tâche fille et celle-ci s'exécute pendant le temps d'attente.

Pour ce faire, il suffit de demander au superviseur d'exécuter la tâche fille en précisant sa priorité. Toujours à l'aide de macro opérations, la tâche mère pourra se synchroniser sur la fin de la tâche fille (Fig. 15).



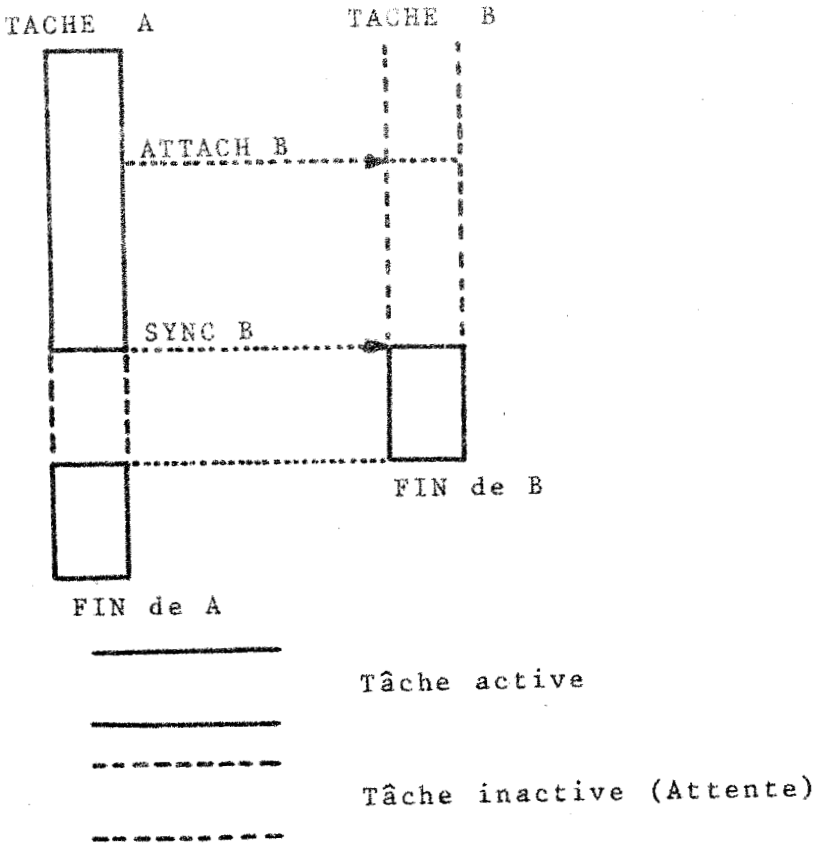
La MACRO (1) permet de prévoir l'exécution de B dès qu'une attente de A se produira

La MACRO (2) permet une synchronisation dans le cas où la tâche B ne serait pas terminée.

FIG. 15

Structure dynamique parallèle

Mais dans le cas de la Figure 16, une structure dynamique parallèle ne présente pas d'intérêt.



La tâche A n'a pas de temps d'attente.

Fig. 16

Cas d'une structure dynamique mal utilisée

2.d - Gestion des interruptions - Alarmes

Le système est conçu pour fonctionner en multi-programmation. L'utilisation optimum de toutes les ressources conduit à avoir en mémoire plusieurs programmes représentant des tâches à exécuter. Lorsqu'une tâche attend la fin d'un évènement, le résultat d'une entrée - sortie par exemple, elle

libère l'unité centrale et le contrôle est donné à une autre tâche prête. La conception traditionnelle du traitement séquentiel des instructions n'est pas remise en cause par la multiprogrammation : l'unité de traitement va chercher une instruction, la décode, l'exécute, incrémente le compteur ordinal et répète l'opération. Une instruction de transfert provoque la modification du compteur d'instructions et le traitement se poursuit à partir de cette nouvelle adresse.

L'exécution simultanée de plusieurs programmes n'est qu'apparente.

Le contrôle, par le moniteur des opérations se déroulant dans le calculateur, demande que celui-ci soit averti lorsqu'une condition quelconque modifiant l'état général du système intervient. Il peut alors prendre les mesures nécessaires.

Soit l'exemple d'une entrée sortie : une fois initialisée, elle se déroule sous le contrôle du canal ou de la logique du périphérique sans intervention de l'unité centrale. Lorsqu'elle est terminée, il est nécessaire que le moniteur reprenne le contrôle afin d'assurer :

- D'une part, la poursuite des opérations après avoir testé les conditions dans lesquelles s'est déroulé l'échange.

- D'autre part, reprendre le déroulement de la tâche en attente de cet évènement (fin de l'échange).

Pour réaliser un tel fonctionnement, on peut imaginer une méthode de scrutation périodique de l'état du périphérique, mais ceci présente l'inconvénient de monopoliser par trop l'unité centrale si le calculateur contrôle un grand nombre de périphériques. Il est préférable de prévoir un

fonctionnement indépendant par des signaux d'interruption. Nous retrouvons la dualité entre les parties cablées et programmées, l'efficacité de l'ensemble est améliorée dans la mesure où les fonctions les plus fréquentes sont reportées sur l'interface.

Des signaux provoquent le transfert automatique du contrôle du programme en cours d'exécution au module de gestion des interruptions du moniteur. Les informations nécessaires à la reprise du programme interrompu sont stockées ; cette opération de rupture n'est donc pas perceptible à l'utilisateur.

D'une manière plus générale, le mécanisme d'interruption est déclenché toutes les fois que le moniteur doit intervenir. Nous pouvons classer les types d'interruptions en plusieurs groupes :

- Les interruptions d'entrée - sortie

Ce type d'interruption est provoqué par un périphérique pour indiquer à l'unité centrale qu'un évênement vient de se terminer, tel que la fin de l'exécution d'un programme canal, ou la fin d'une opération d'entrée - sortie ou encore le changement d'état d'une unité.

- Les interruptions de type programme

Le bloc de commande de l'unité centrale enchaîne automatiquement les instructions. Cependant, certaines conditions anormales d'exécution peuvent être détectées : ces conditions provoquent une interruption de type programme.

Certaines ont trait aux opérandes ou aux résultats (débordement , opérateurs incorrects, perte de précision ....)

d'autres ont trait aux conditions empêchant le déroulement normal du programme lui-même (code opération invalide , adressage impossible ....), d'autres encore provoquent l'intervention du moniteur dans le cas de tentative de masquage intempestive des niveaux d'interruption, ou à la suite de définition de zones qui provoquerait la détérioration du moniteur. Le repérage de ces défauts se fait à l'aide de fonctions hardware.

- Les appels au moniteur

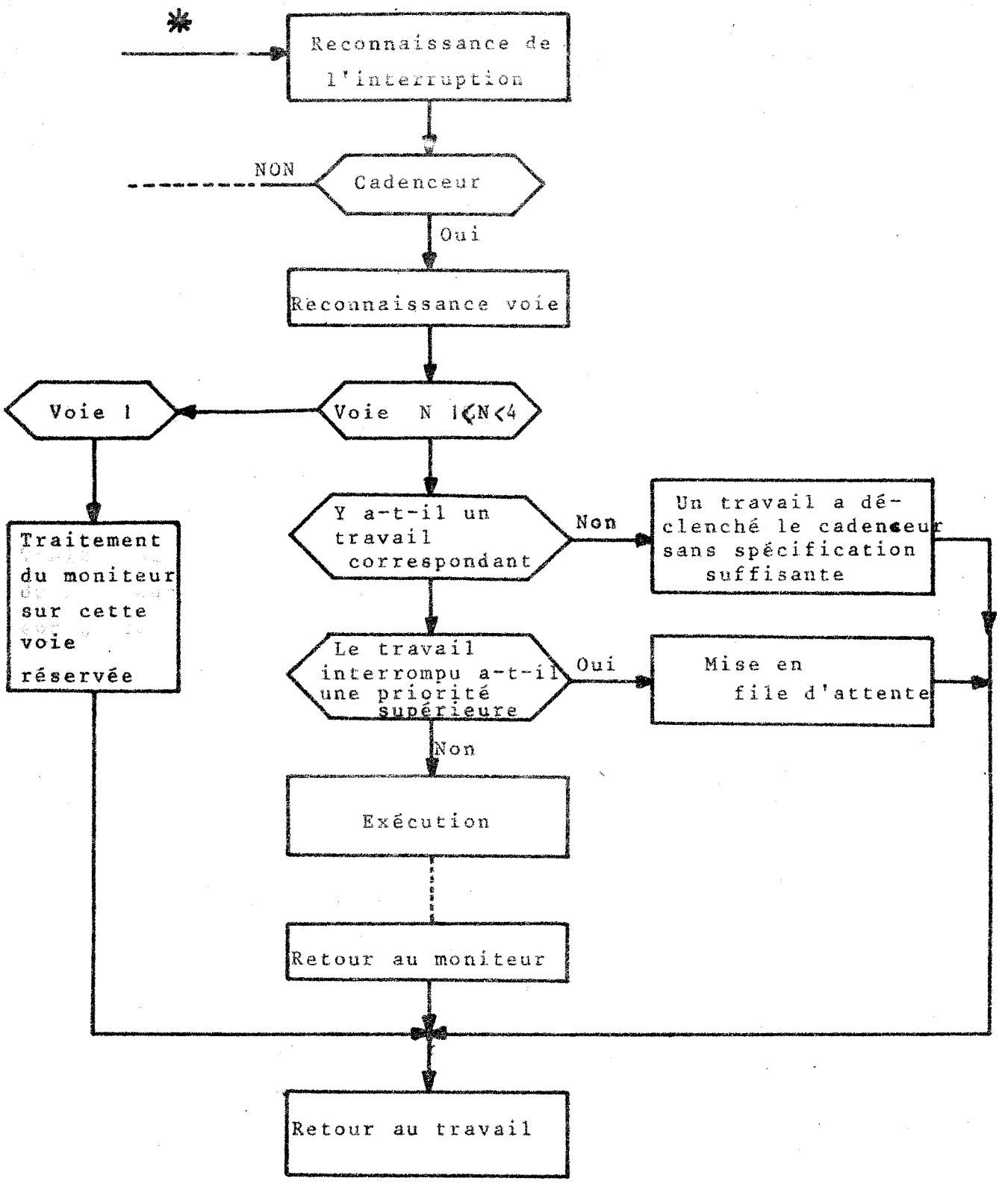
Toutes les fois qu'un utilisateur désire faire appel à l'une des fonctions disponibles à partir du moniteur, il provoque une interruption, arrêtant momentanément le déroulement de son programme pour permettre le déroulement de la fonction moniteur correspondante.

- Interruption externe

L'opérateur dispose sur la console d'un bouton d'appel générant une interruption. L'interprétation de cette interruption dépend alors de la convention établie entre le système et l'opérateur. Un travail sous le système moniteur est ainsi mis en attente d'une commande.

Un autre élément, indépendant de l'unité centrale est l'horloge temps réel. C'est un compteur décrémenté à intervalle régulier, provoquant une interruption dans l'unité centrale lors d'un passage à zéro. Ce compteur est chargé par programme. Le cadenceur possède quatre compteurs dont l'un est réservé au moniteur pour mesurer les temporisations, et la durée des travaux.

Il existe enfin d'autres possibilités d'interruption sur "appels externes", module spécialisé permettant les appels sur un niveau cablé.



A l'aide des macro-instructions appropriées, l'utilisateur demande l'affectation d'une tâche à une des voies du cadenceur. Si une voie est disponible, le moniteur tient compte de la demande et la satisfait ; sinon, le travail est inhibé jusqu'à la libération d'une des voies.

Fig. 17

La fonction horloge

- Interruption d'erreur machine

Lorsqu'une erreur est détectée dans l'exécution de l'instruction en cours, celle-ci est interrompue. Les erreurs machine proviennent exclusivement de la défaillance d'un élément physique du système. Certains masquages, interdits à tous les travaux, peuvent être utilisés par le moniteur, afin de se garder des interruptions qui pourraient survenir lors des parties non réentrantes du programme, ou lors de manipulations de tables.

D'une façon générale, toutes les interruptions sont traitées par le moniteur, mais l'utilisateur a la possibilité de gérer certaines interruptions en précisant le niveau du travail correspondant.

Un exemple est présenté figure 17.

En annexe, nous fournissons les organigrammes de quelques fonctions du moniteur, permettant de préciser le mode de fonctionnement de celui-ci.

Au niveau du système, il existe un programme périodique de scrutation qui gère notamment les piles de condition, de temporisation et de synchronisation des tâches. Il traite aussi la pile des interruptions ; en effet, lorsque deux interruptions ou plus arrivent simultanément sur un même niveau, le moniteur traite celle de plus haut sous niveau mais mémorise l'arrivée des interruptions de niveau inférieur ; les travaux correspondants seront alors traités par la suite (Fig. 18).

Nous voyons à ce propos qu'il existe deux sortes de priorités. Les priorités des interruptions qui correspondent à celles des périphériques utilisés, et sont fixées par hardware

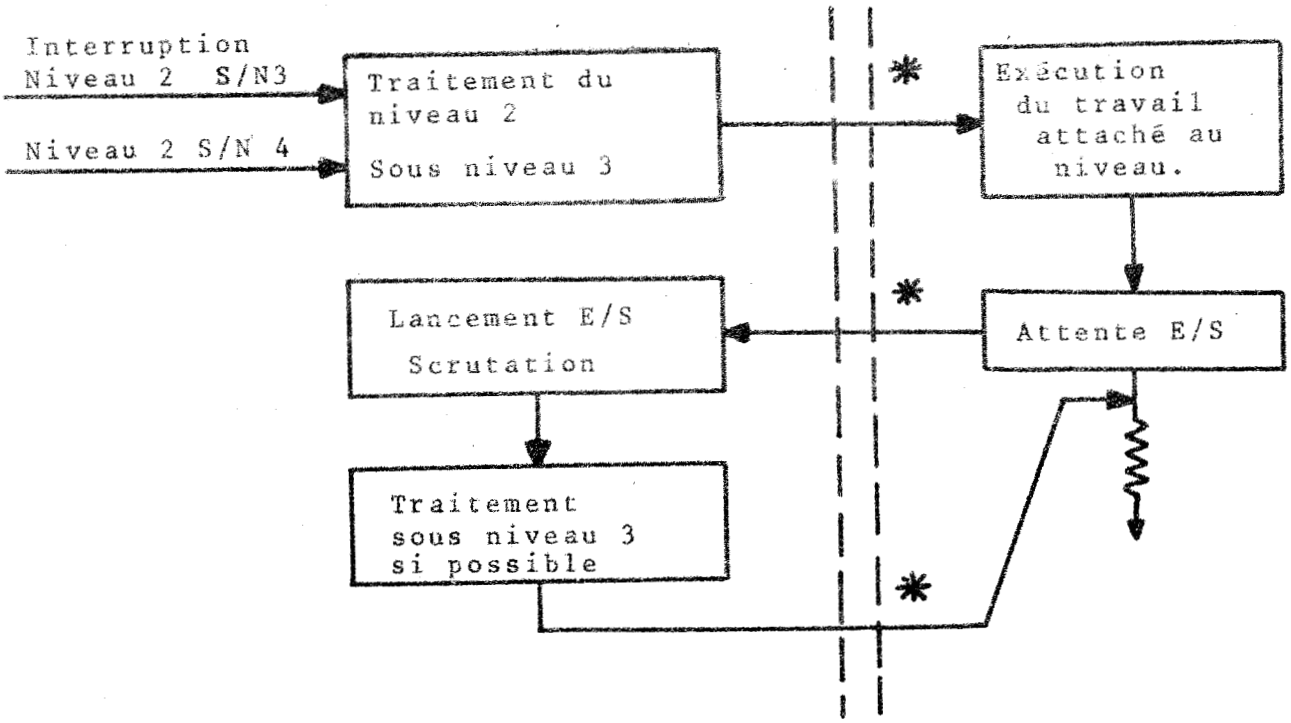


Fig. 18

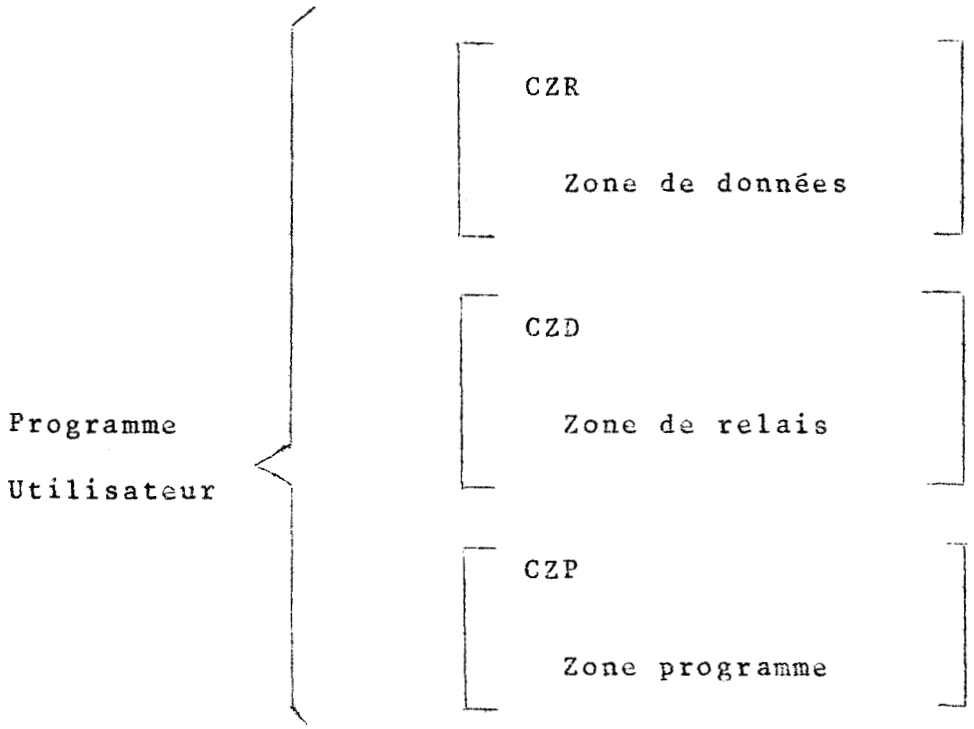
Traitement des interruptions

et d'autre part, les priorités affectées aux travaux et fixées par programmation. Ce mode de fonctionnement permet une hiérarchisation très poussée des tâches présentes en mémoire.

1.3 - Spécifications et contraintes

Comme nous l'avons remarqué précédemment, tous les appels au moniteur se font par l'intermédiaire de macro-instructions, ce qui permet une grande souplesse dans l'élaboration du programme de l'utilisateur, sans nécessiter de sa part une connaissance approfondie de la structure de la machine. Cependant, la structure par bloc de la mémoire pour la gestion dynamique des programmes impose certaines contraintes sur leur structure externe. En effet, pour pouvoir déplacer un

programme en mémoire, il faut nécessairement modifier certaines adresses. Si le mode d'adressage est tel que l'adresse effectuée est calculée à partir d'une base et d'un déplacement, ceci n'offre pas de difficultés majeures. Mais tel n'est pas le cas sur le calculateur utilisé, où certaines adresses peuvent se confondre avec des instructions. Il faudra donc que le moniteur ait la possibilité de distinguer les zones de données, essentiellement fixes, les zones de relais à modifier ainsi que le code exécutable à transformer partiellement. La structure externe devra donc présenter l'allure suivante (Fig. 19)



L'ordre des zones n'a pas d'importance puisque les zones sont repérées à l'aide des macro-instructions CZR - CZD - CZP

Fig. 19

Une autre contrainte importante est que le moniteur n'est conçu à ce stade que pour les programmes assembleurs. Ceci ne décharge pas totalement l'utilisateur au point de vue

de la programmation. Nous avons été amenés à prévoir un superviseur acceptant les commandes en fortran et permettant l'utilisation du couplage en langage évolué. Ce système présente l'inconvénient de n'accepter qu'une seule tâche à la fois et de travailler moins rapidement. Nous reviendrons sur la description un peu plus loin. Nous joignons en annexe de ce chapitre des exemples d'utilisation du moniteur à partir de la console.

2 - SOFTWARE RELATIF AU SOUS ENSEMBLE DE LIAISON ANALOGIQUE - LOGIQUE

2.1 - Le programme de couplage

Nous décrirons les possibilités de l'ensemble sous le contrôle du moniteur dans le cadre d'un fonctionnement en mode prioritaire, mais il semble à ce niveau indispensable de présenter le matériel sur lequel nous avons réalisé le couplage. Nous nous attacherons toutefois, dans un esprit de généralisation, à définir une conception modulaire de l'ensemble, la partie concernant la réalisation technique ayant été décrite aux chapitres précédents.

1.a - Description du matériel

L'ordinateur de processus est un calculateur utilisant des mots de 20 bits, dont un de parité mémoire. Il possède actuellement une mémoire de 16 K mots (16384 mots) extensible à 32 K (32768 mots), divisée en blocs de 4 kilomots. Une série de 56 instructions de base permet de travailler sur

plusieurs registres et sélectionnent un mot en mémoire suivant un adressage direct, indexé, indirect, indexé ou indirect à niveau multiple etc...; le principal registre est l'accumulateur, par lequel transitent toutes les informations lors des échanges avec les périphériques (Fig. 20).

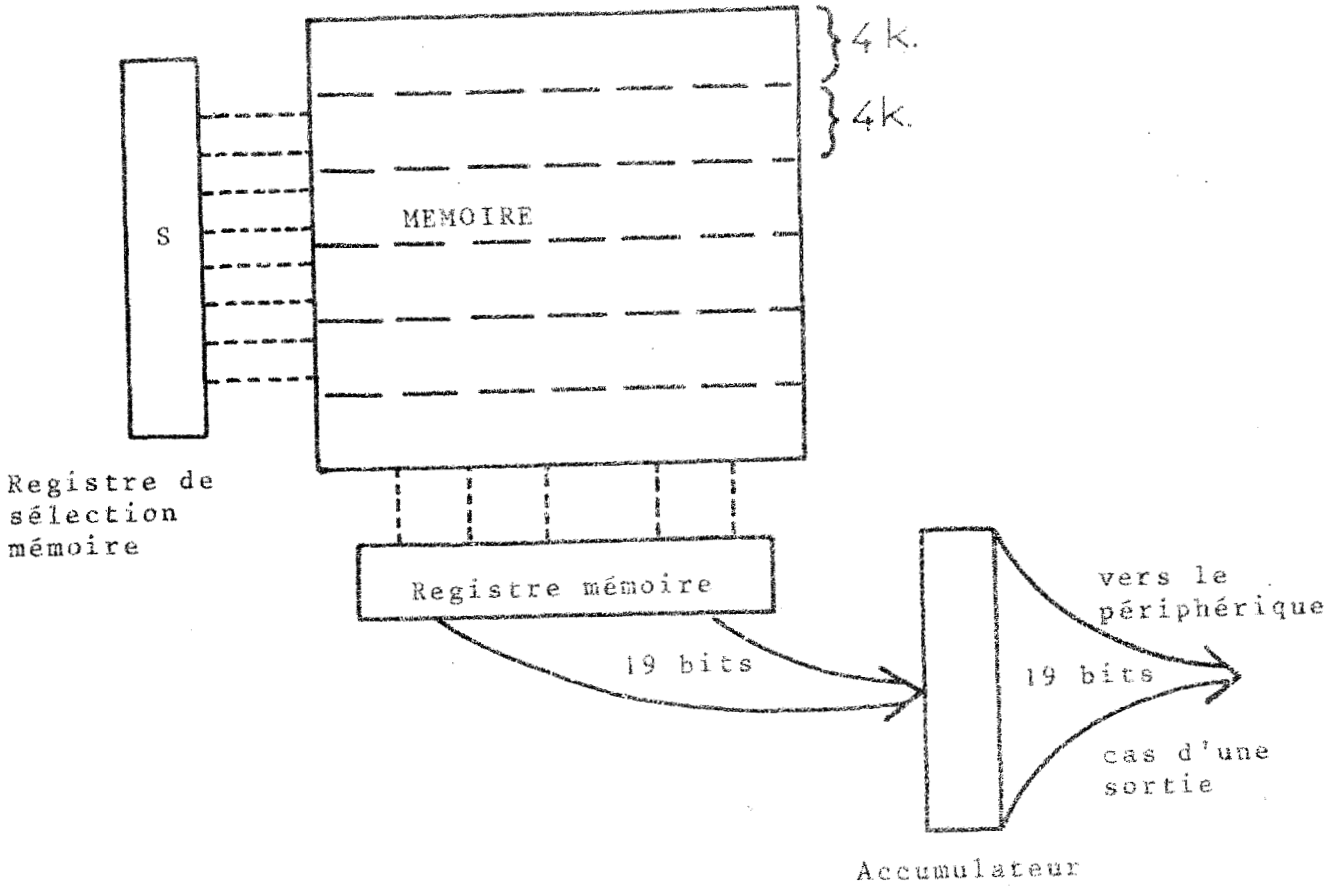


Fig. 20

La mémoire

Toutefois, la gestion des échanges nécessite la présence d'un élément qui permettra une mémorisation des informations, émises ou reçues, le contrôle et la commande des séquences nécessaires au déroulement des entrées ou sorties , ce coupleur permet une autonomie vis à vis de l'unité centrale et réalise l'adaptation des signaux (format et nature des niveaux).

Un coupleur se compose donc de deux parties (Fig. 21) :

- Le tampon qui assure la fonction de mémorisation et la reconnaissance d'adresse

- La fonction contrôle commande et adaptation technologique.

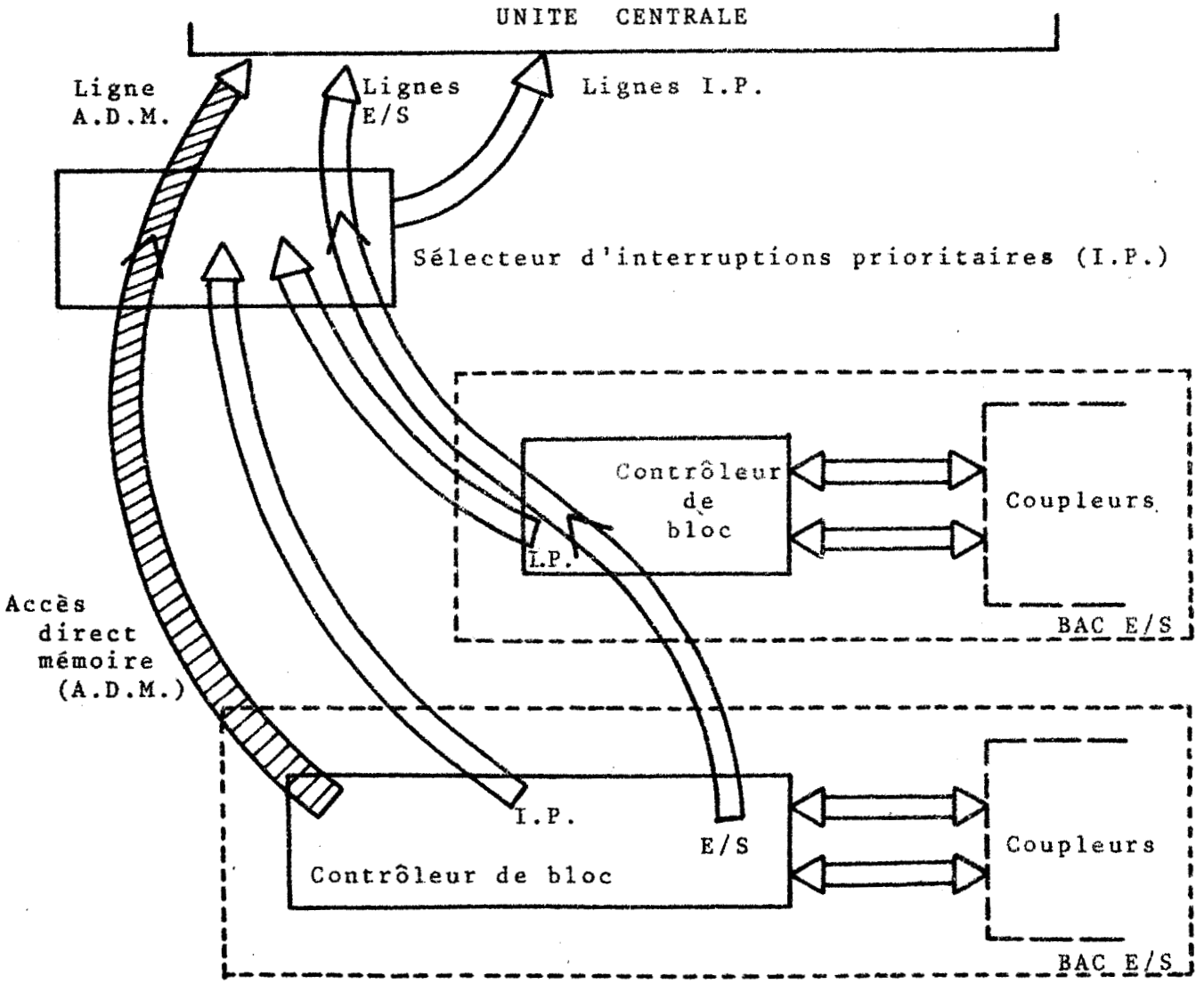


Fig. 22

Les différents bus

Les signaux de service sont aiguillés sur le coupleur en même temps que les informations et servent à déclencher et suivre le déroulement des séquences d'échange élaboré par le contrôleur.

Ce rôle d'interface consiste à aiguiller et à conditionner les signaux mis en jeux au cours des échanges d'informations entre l'unité centrale et les coupleurs de périphériques d'entrée - sortie. Cette adaptation terminale est dans le cas le plus général connectée à l'unité centrale par la ligne d'entrées - sorties programmées, la ligne d'inter-

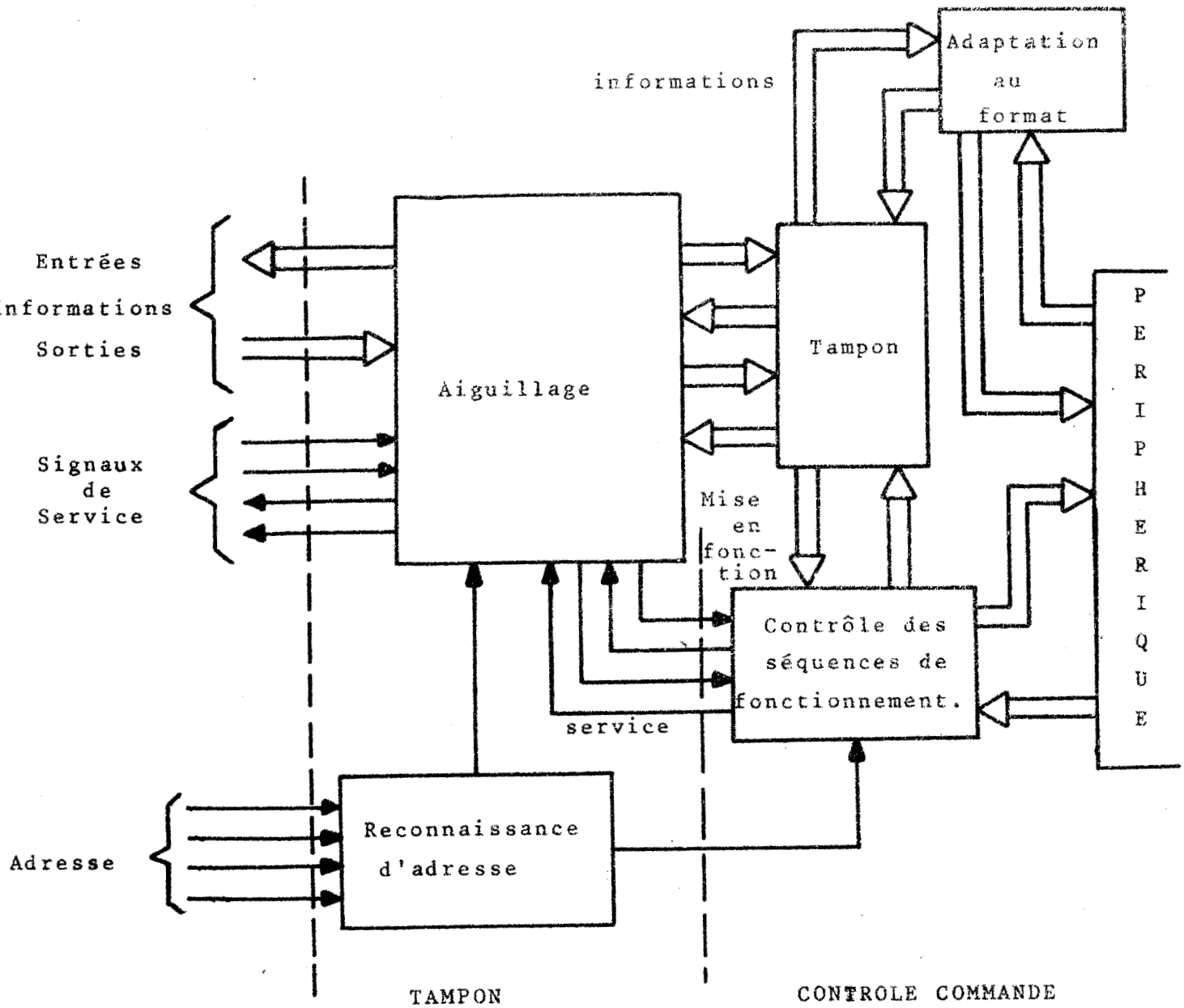


Fig. 21

Le coupleur

ruption prioritaire et la ligne d'accès direct mémoire (canal) auxquelles correspondent différents modes d'échanges (Fig. 22)

Le bloc entrée - sortie assure la connexion d'un certain nombre de coupleurs de périphérique à un contrôleur de bloc, qui est particulier à chaque mode d'échange et qui assure la connexion à l'interface interne du calculateur.

1.b - Modalités d'échange

Les échanges avec les périphériques peuvent se faire de plusieurs manières :

- Le mode programmé simple, que nous utiliserons peu, car le temps d'attente lors des échanges peut devenir important dans le cas de périphériques lents.

- Le mode programmé prioritaire, très utilisé pour la synchronisation entre le calculateur et le périphérique.

- Le mode canal qui est le système d'échange le plus performant car il correspond à faible temps d'occupation de l'unité centrale.

Nous allons envisager les possibilités correspondant à ces modalités en détaillant celles qui nous seront le plus utiles.

- Mode programmé simple

Le programme a alors l'initiative des échanges par les instructions EA (entrée dans l'accumulateur des informations envoyées par un périphérique), ou SA (sortie de

l'accumulateur, opération inverse de la E.A). Un seul mot est échangé au cours de chaque instruction, l'adresse effective contenue dans la partie opérante sélectionne le périphérique par son coupleur (Fig. 23).

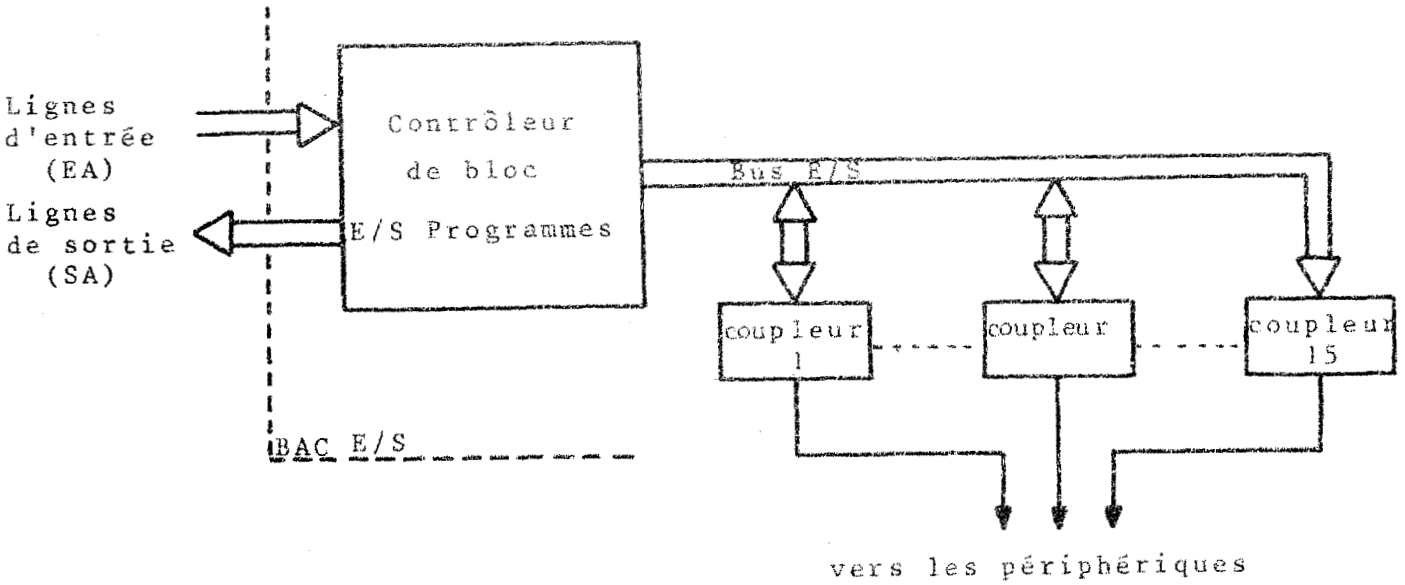


Fig. . . 23

Le bac d'Entrée - Sortie

- Mode programmé prioritaire

Si en mode programmé simple, il faut interroger systématiquement chaque coupleur de périphérique et éventuellement renouveler cette interrogation jusqu'à la disponibilité de l'organe, on ne fait intervenir en mode prioritaire l'unité centrale que lorsque le périphérique est disponible.

Ce mode d'échange utilise le même contrôleur de bloc qu'en entrées - sorties programmées simple, mais on y adjoint un dispositif de connexion au système d'interruptions prioritaires. Les organes périphériques informent l'unité

centrale de la disponibilité (occupation, validation des informations etc...) par l'intermédiaire de la ligne d'interruptions prioritaires, les échanges proprement dits se faisant par les lignes d'entrées - sorties (Fig. 24). Les coupleurs sont en liaison avec un coupleur spécial (coupleur zéro) qui lui-même est relié à l'unité centrale par l'intermédiaire d'un module de sélection de priorité.

Le système d'interruption se compose de 8 niveaux cablés, chacun pouvant se décomposer en 32 sous niveaux, qui offrent un éventail de combinaisons largement suffisant.

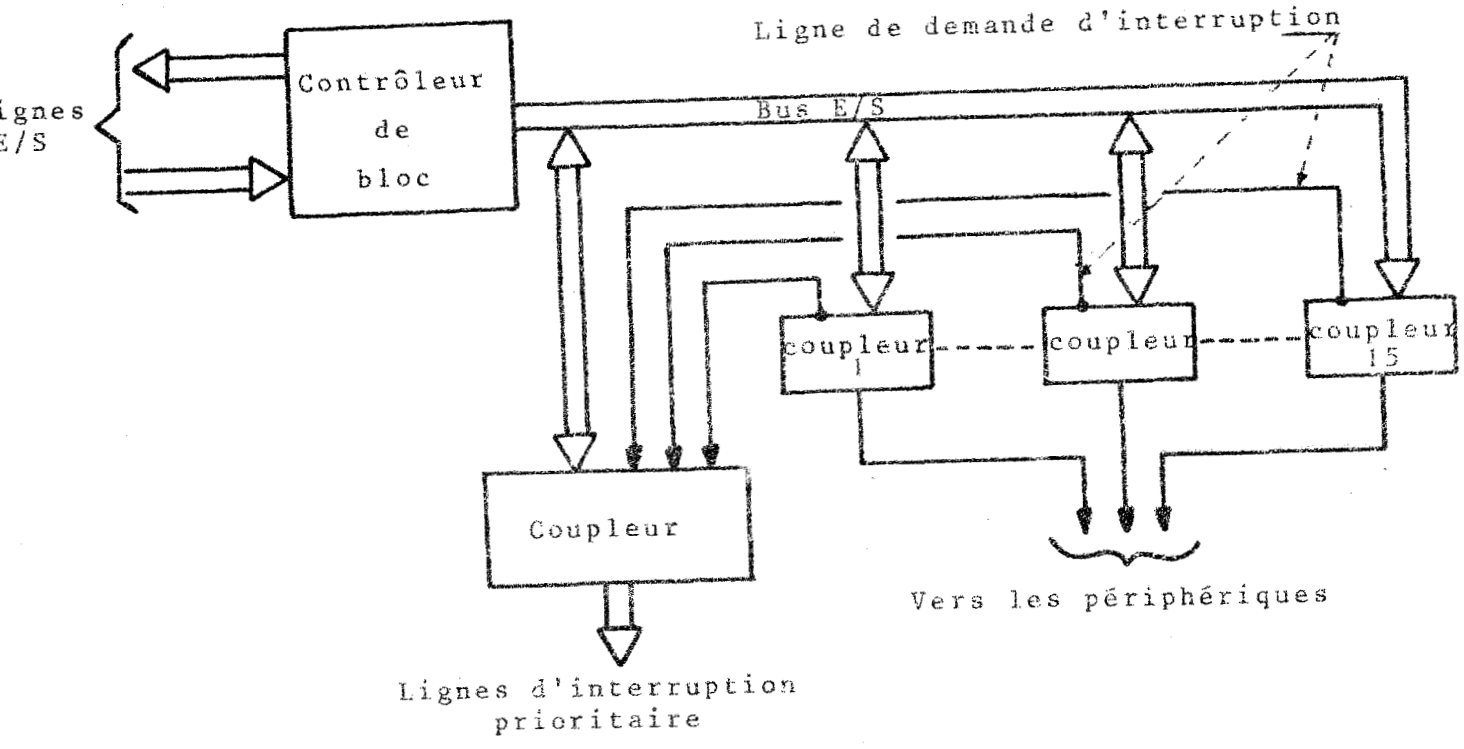
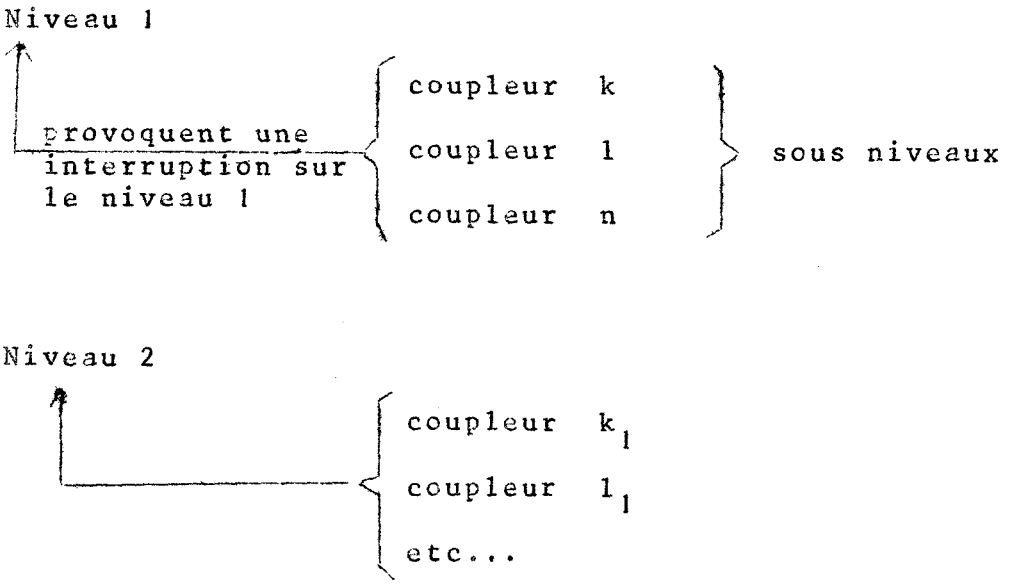


Fig. 24

Le mode prioritaire

On distingue les sous niveaux dans le bloc d'entrées - sorties à l'aide de deux registres se trouvant dans le coupleur zéro, ROCC (registre des occupations) et RDEF (registre des défauts). A chaque bit de ces registres correspond un coupleur, ainsi l'étude de ces registres permet le repérage du coupleur provoquant l'interruption sur le niveau principal.

Exemple :



Le passage au niveau logique "1" de l'un quelconque des 16 bits de ROCC provoque une demande d'interruption se traduisant par un branchement inconditionnel du programme correspondant au niveau affecté au bloc d'entrée - sortie. Le rôle de ce programme sera d'identifier par acquisition de ROCC et RDEF l'origine de l'appel et de provoquer l'exécution de la routine spécifique à l'appel (Fig. 25).

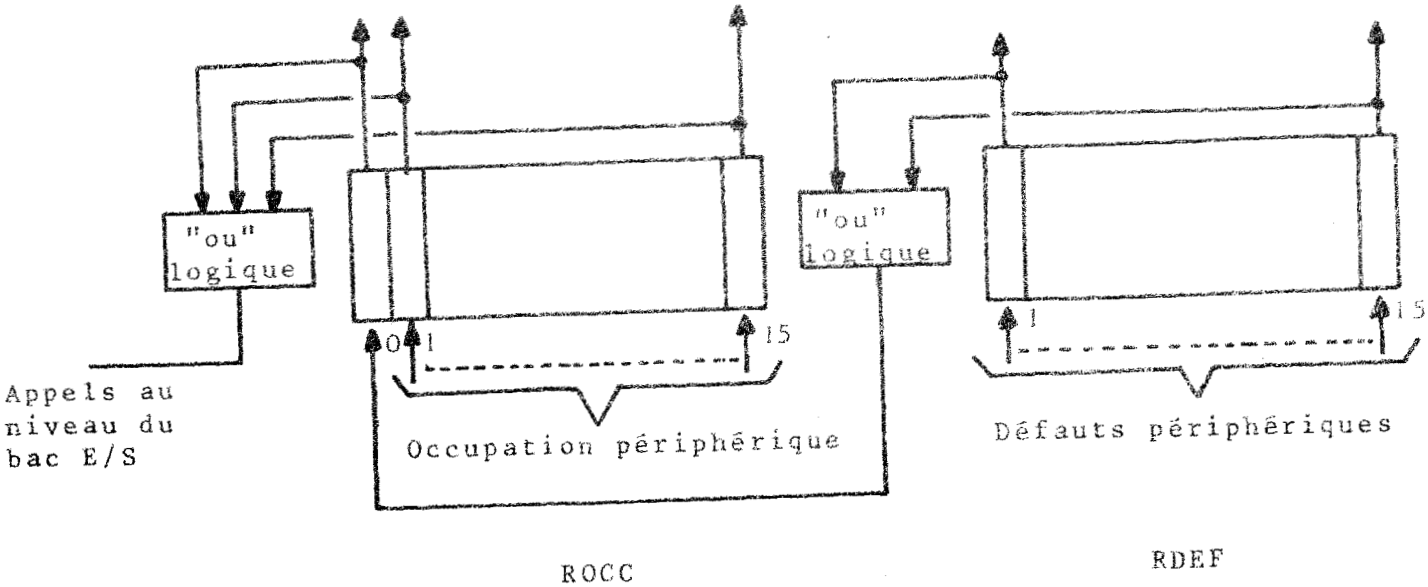


Fig. 25

Les registres d'interruption

- Mode Canal

Dans ce mode de fonctionnement, l'unité centrale se borne à fournir au bloc d'Entrées - Sorties canal les données de l'échange et celui-ci demande et gère chaque cycle élémentaire à l'intérieur du message. Le bloc d'Entrées - Sorties est doté d'une certaine autonomie vis à vis de l'unité centrale puisque sur sa demande il assure seul la gestion des cycles élémentaires d'échange avec les coupleurs du périphérique. L'unité centrale déclenche l'échange à l'adresse du coupleur travailleur en fournissant l'adresse de la table de travail dans laquelle se trouvent les modalités de l'échange. A la fin de l'échange, le canal est libre et provoque une interruption pour indiquer la manière dont s'est déroulé cet échange (Fig. 26).

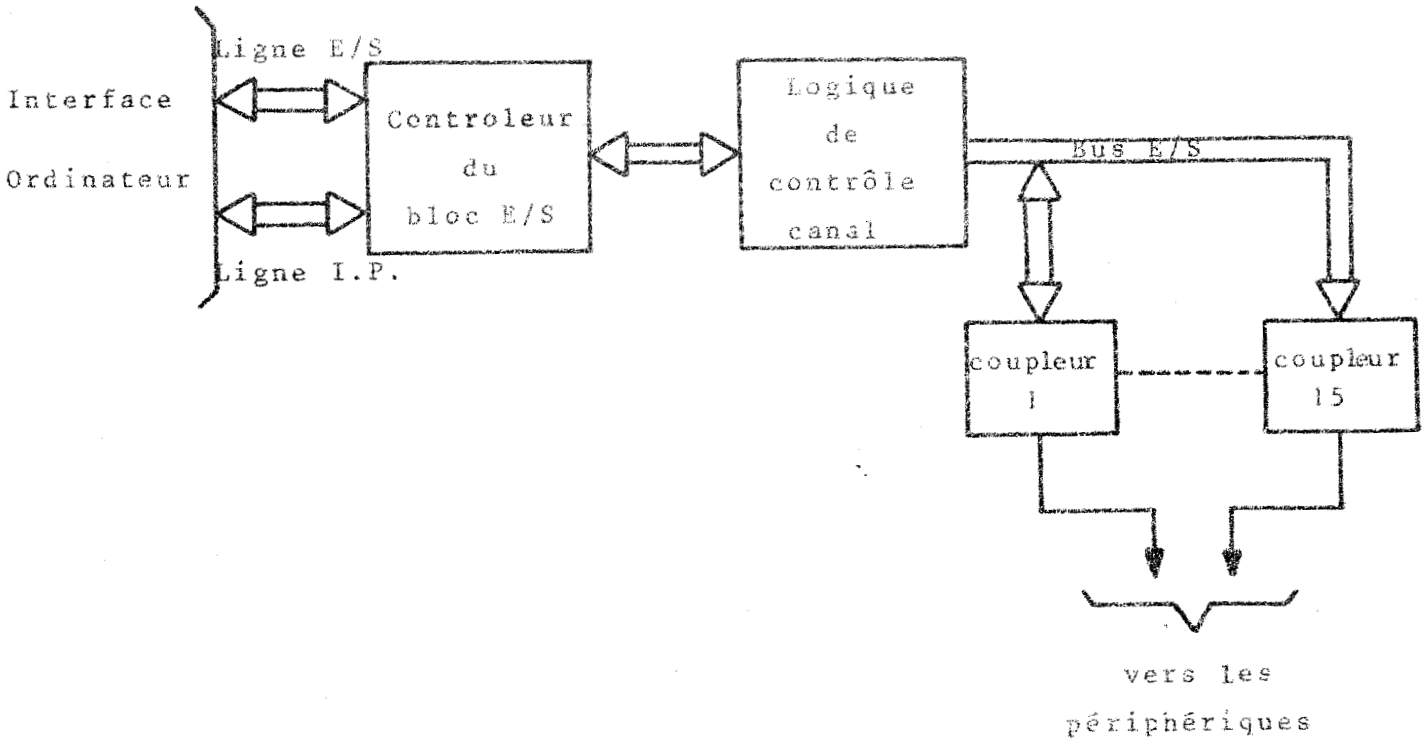


Fig. 26

Le Canal

1.c - Possibilités techniques du calculateur

- Le Calculateur Numérique.

Nous avons vu que le grand nombre de niveaux d'interruption permet la gestion de plusieurs périphériques dans des configurations très variées. Il sera donc possible d'établir une hiérarchie des tâches prioritaires, telle que les échanges avec la machine analogique se fassent dans les conditions optimales, en lui imposant par exemple des fonctions de calculs se déroulant à des vitesses différentes (calcul multivitesse, commande des temps d'intégration ....). Le rôle de l'ordinateur est d'autre part d'intervenir dans les

plus brefs délais, en cas d'alarme. Il détecte l'origine de l'alarme en consultant le mot d'état de la calculatrice analogique et exécute la routine de traitement du défaut.

Le format de 19 bits pour les registres et pour la mémoire présente l'avantage considérable de limiter le nombre d'opérations d'Entrées - Sorties, permet un gain de temps appréciable.

La précision dans les calculs se trouve accrue. Il n'est pas nécessaire de travailler en double précision, d'où une diminution de l'occupation en mémoire.

Un répertoire très complet d'instructions allié à une grande variété d'adressages permet une grande souplesse dans la programmation des échanges et des traitements de tables de valeurs ; les opérations logiques facilitent les reconnaissances rapides au niveau du digit, et favorise la pleine utilisation de chaque mot.

D'autre part, l'horloge interne du calculateur envoie des interruptions à des intervalles de temps variable. Celles-ci sont utilisées pour mettre en attente des tâches durant un laps de temps programmable. Cette possibilité est riche d'applications, par exemple la simulation d'un retard pur dans le cadre de l'ensemble hybride.

Le calculateur numérique présente de grandes possibilités mais se révèle donc inapte dans des domaines particuliers. Notre but est de faire en sorte que certaines limites soient levées grâce à l'adjonction d'une machine analogique, mais d'autres dépendent de la structure même de l'ordinateur et restent sans solutions immédiates, les améliorations ne dépendant que des progrès technologiques ou plus simplement du remplacement du calculateur par un autre plus performant.

- La calculatrice Analogique

L'E.A.I. 580 est une machine hybride de type -I, à logique parallèle, comprenant des sommateurs, des multiplieurs,..... Son organisation peut se présenter sous la forme d'un schéma synoptique (Fig. 27).

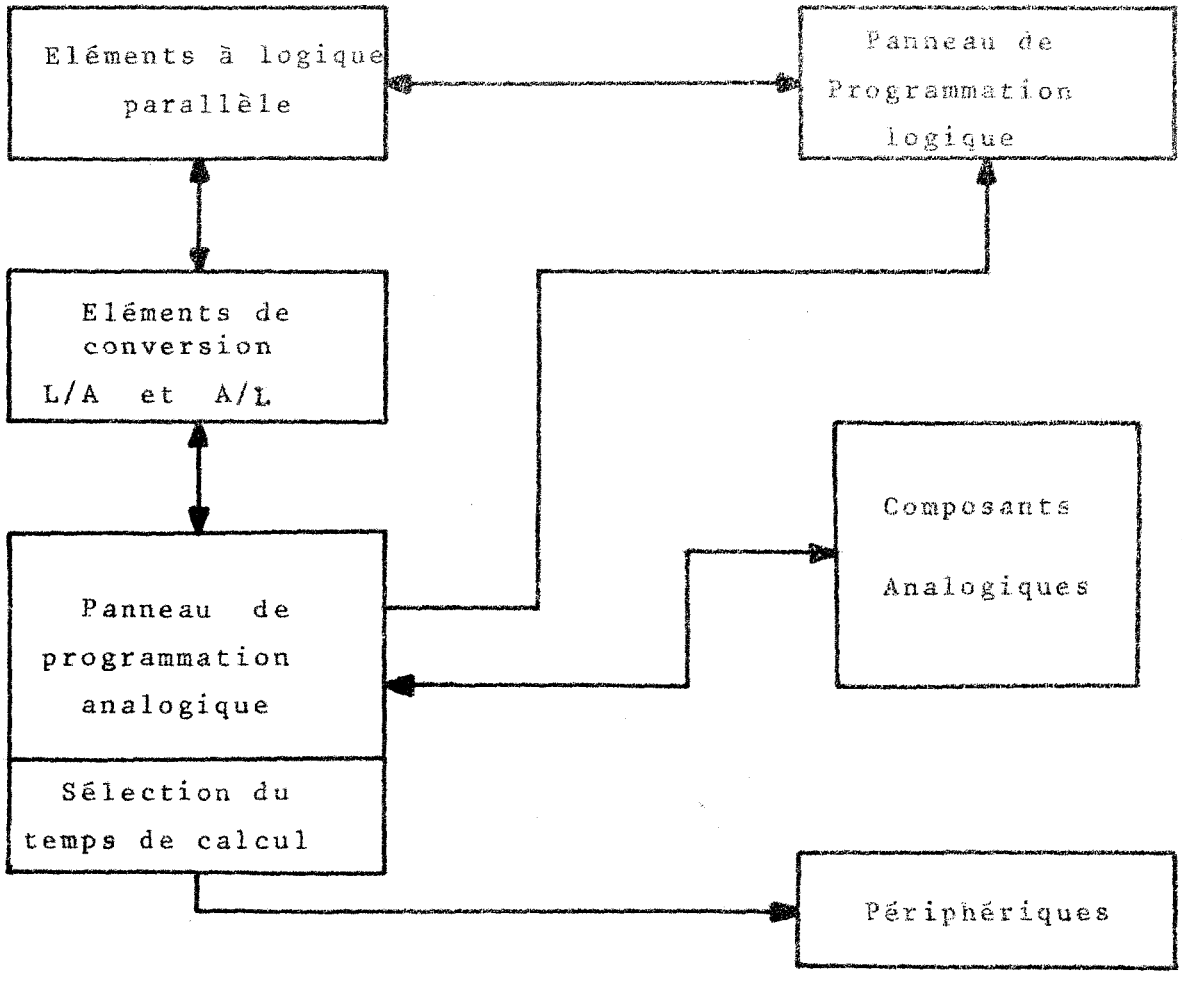


Fig. 27

La machine analogique

On peut distinguer sur cette machine les commandes analogiques - logiques, les commandes d'horloge correspondant à l'échelle des temps choisie et l'affichage des potentiomètres.

. Les commandes analogiques et logiques

On les utilise pour régler le mode opératoire des intégrateurs : conditions initiales, intégrations, gel. Certains intégrateurs peuvent être commandés à partir de la logique positive parallèle, de niveaux 0 volt et + 5 volts, alors que d'autres pourraient l'être manuellement. Dans le cas d'une saturation, l'amplificateur en défaut est repéré grâce à un tableau lumineux indiquant son numéro.

. Echelle des temps

Plusieurs possibilités s'offrent à l'utilisateur : les fréquences d'horloge s'étagent de 1HZ à 1MHZ d'une part, il existe un choix sur la rapidité du mode répétitif et la constante d'intégration (1s, 2ms) d'autre part.

. Affichage des potentiomètres

Il convient de distinguer les potentiomètres numériques, et les potentiomètres utilisant un cerveau moteur. Cependant à notre niveau, cela ne représente qu'une différence de principe, leur fonctionnement ne se différenciant que par leur durée d'affichage. Les échanges se font en mode canal ou en mode prioritaire pour permettre une exploitation en temps réel, ce qui nous autorise à ne pas les distinguer du point de vue de la programmation, le mode d'adressage étant le même.

. Les commandes programmées

Lors de la description du moniteur, nous avons fait une différence selon le mode de travail, en Fortran ou en assem-

bleur ; elle ne s'impose plus ici, car le traitement des échanges hybrides se fait à partir de sous programmes des routines d'accès appelés soit par la procédure CALL du Fortran, soit à l'aide de macro instructions spécifiques.

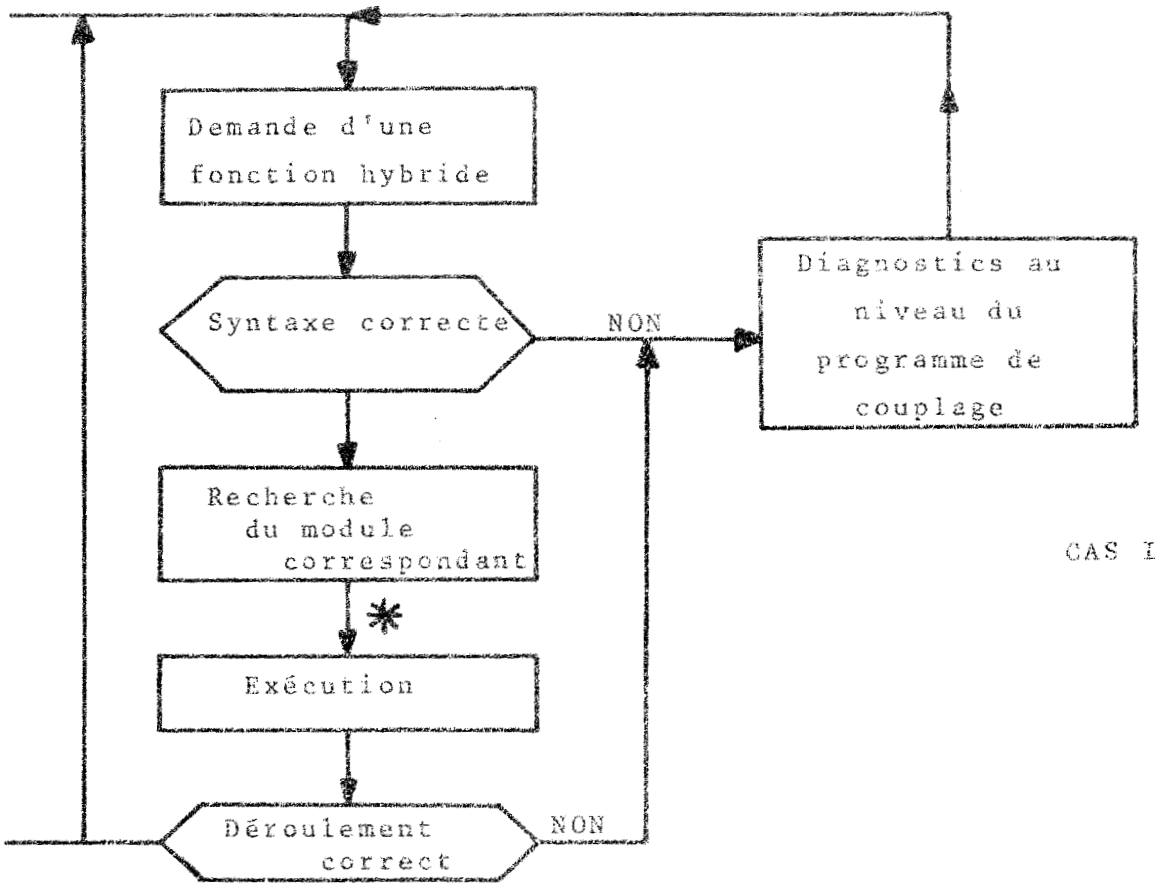
En effet, deux possibilités s'offrent à l'utilisateur. Il lui est possible de traiter son problème, pas à pas, en mode conversationnel, directement sans le moniteur, l'interprétation des commandes se faisant immédiatement. Certaines séquences peuvent être réalisées à l'aide de programmes en mémoires, mais lors de l'achèvement des tâches correspondantes, l'opérateur reprendra le contrôle automatiquement, ce qui lui donne la possibilité de voir évoluer son problème et de le résoudre au rythme voulu, en en modifiant le traitement s'il le désire.

Une autre possibilité consiste à traiter le programme hybride comme un travail global ne laissant aucune initiative à l'opérateur. Dans ce cas, il va de soi que toute erreur entraîne l'abandon pur et simple du travail sans possibilité de correction. L'émission de messages dans ce cas facilite certains réglages nécessitant l'intervention manuelle de l'opérateur. Ces diverses possibilités sont schématisées dans la figure 28.

Le moniteur traite et prévient les erreurs au niveau des entrées - sorties, le programme de couplage interprète les commandes dans le cas (I), les refuse ou les accepte suivant leur compatibilité et la syntaxe de la phrase.

Dans le mode conversationnel (cas I) toute erreur est donc immédiatement détectée et les alarmes au niveau du moniteur n'interviendront pratiquement jamais. Cependant dans le cas II, ainsi que dans le cas III (travail inclus dans un train de travaux), les vérifications se feront au niveau du programme de couplage. Dans le cas de détection d'une erreur,

nous avons un retour au moniteur, qui suivant les spécifications qu'il a reçues à l'introduction du travail, décide s'il l'abandonne (cas III) ou s'il se met en attente d'une intervention à partir de la console.



Intervention du moniteur par la gestion des modules et le lancement de l'exécution.

Il est cependant nécessaire que toutes les entrées - sorties se fassent par le moniteur, et l'on est en fait continuellement sous son contrôle

Fig. 28 a

Le cas II correspond au cas I, mais le déroulement ne se fait pas fonction par fonction. L'utilisateur travaille alors en mode opérateur et gère à partir de la console, les tâches incluses dans son travail, chaque tâche pourrait être considérée comme des modules de priorité différente.

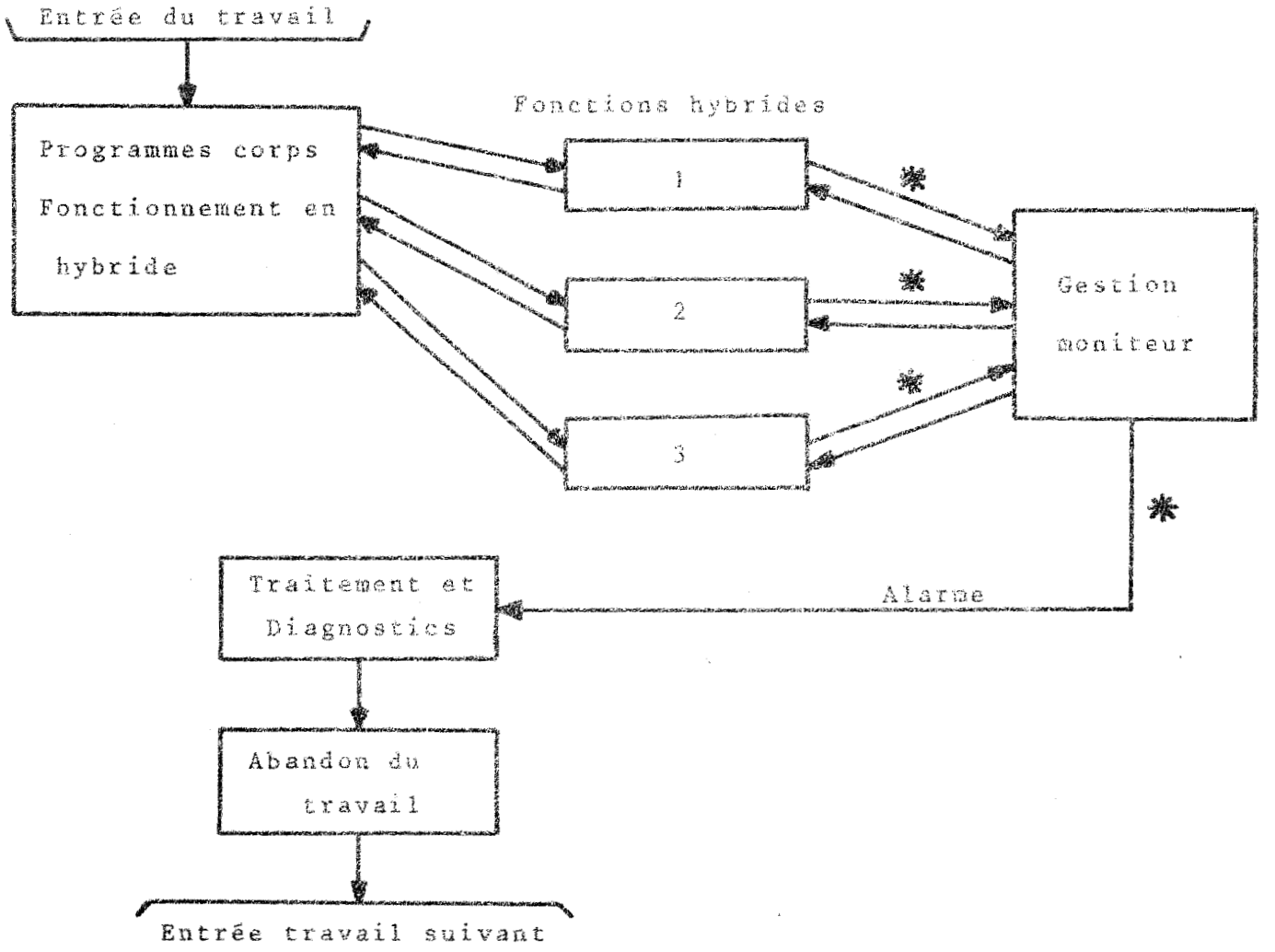


Fig. 28 b

Fonctions hybrides

Comme nous avons plusieurs télétypes, l'une est affectée au système, c'est à dire qu'elle rend compte des états des travaux, les autres sont affectées aux différents travaux suivant les besoins. Il est toutefois possible de ne travailler qu'avec une seule console, mais cela présente l'inconvénient d'avoir des commentaires du moniteur au milieu des résultats. En effet, le moniteur "prend" systématiquement la télétype dès qu'il abandonne un travail à la suite d'une alarme, mais la "rend" après utilisation, à la tâche à laquelle elle était affectée.

1.d - Commande des modes analogiques et logiques

Nous avons distingué auparavant les différentes fonctions sur les deux machines dont nous disposons actuellement. Leur utilisation ne présente guère de complications, que ce soit à partir d'une console ou par programme. Nous devons sélectionner les machines qui peuvent évoluer indépendamment l'une de l'autre ou en parallèle (mode esclave par exemple). Dès le départ, c'est à dire à l'initialisation, les machines sont mises dans un état standart (repos analogique et logique). Si nous traitons notre problème en mode conversationnel, il nous faut préciser le temps horloge ainsi que la vitesse des intégrations. Le calculateur se met par la suite en attente d'une commande. Celle-ci se fait en utilisant la même notation, que celle des boutons poussoirs de la machine.

Exemple : SP - CL ;

demande la fonction Set Pot (repos analogique) et CLear, (repos logique)

Si par la suite nous faisons :

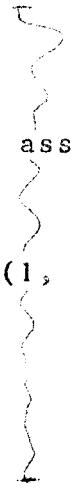
I C ;

demandant la mise en condition initiale de tous les intégrateurs, la commande logique n'ayant pas été définie, la précédente reste valable. Autrement dit, toute commande reste en vigueur jusqu'à sa modification explicite à l'aide d'une autre commande du même genre. Il en va de même pour le choix des temps de calcul et du numéro de la machine sélectionnée.

Nous avons regroupé en fin de chapitre les échantillonnages de commande possibles à partir de la console.

L'utilisation par programme de l'ensemble hybride ne présente pas plus de complications. Comme précédemment, nous nous sommes attachés à garder les mêmes notations, et l'exploitation se fait à partir de macro instructions, pour une étude le language assembleur, ou à partir d'appel de sous programmes pour le langage Fortran.

Exemple : en assembleur, nous aurons :



Programme assembleur

H Y B D (1, OP, RU, T1, H6)

suite

La macro instruction H Y B D demande d'afficher sur la machine I le mode analogique OP (opérate), le mode logique Ru (RuN : mise en fonction de la partie logique), les intégrateurs ayant un temps d'intégration de 1 seconde, avec une horloge de fréquence 10^6.

On pourrait faire suivre cette commande de :

H Y B D (, HD , , , ,)

qui demanderait les mêmes fonctions que précédemment, la commande analogique devenant alors une mise en gel des intégrateurs (HD : HOLD), tous les autres paramètres étant ceux définis par ailleurs.

Ainsi, l'utilisateur travaille au niveau de l'assembleur, avec tous les avantages que cela comporte, mais n'a pas à se soucier des échanges avec les périphériques, conventionnels ou non, qui nécessitent généralement une parfaite connaissance de leur fonctionnement et alourdissent considérablement les programmes. L'emploi à ce niveau des macros instructions permet au système de jouer réellement un rôle de tampon lors de la mise en oeuvre des routines de méthode d'accès.

La formulation est identique lors du traitement dans un contexte Fortran, l'appel aux routines se faisant de manière classique par un CALL :

CALL HYBD (paramètre 1, paramètre 2, paramètre 3,
paramètre 4, paramètre 5)

Les paramètres ont alors la même signification que précédemment.

1.e - Commande des potentiomètres

La commande des potentiomètres asservis nécessite la mise en SP (repos analogique), la sélection proprement dite, suivie de l'affichage de la valeur. Cependant du point de vue de l'utilisateur, seules ces deux dernières étapes importent,

la première se faisant automatiquement. C'est ce qui permet de ne pas différencier, à ce niveau, les potentiomètres asservis des potentiomètres numériques. Pour la réalisation pratique de l'échange, la connaissance de la nature de l'élément importe. Ceci se fait au niveau I du programme de traitement, c'est à dire au niveau II de la chaine hiérarchique de la Fig. 29.

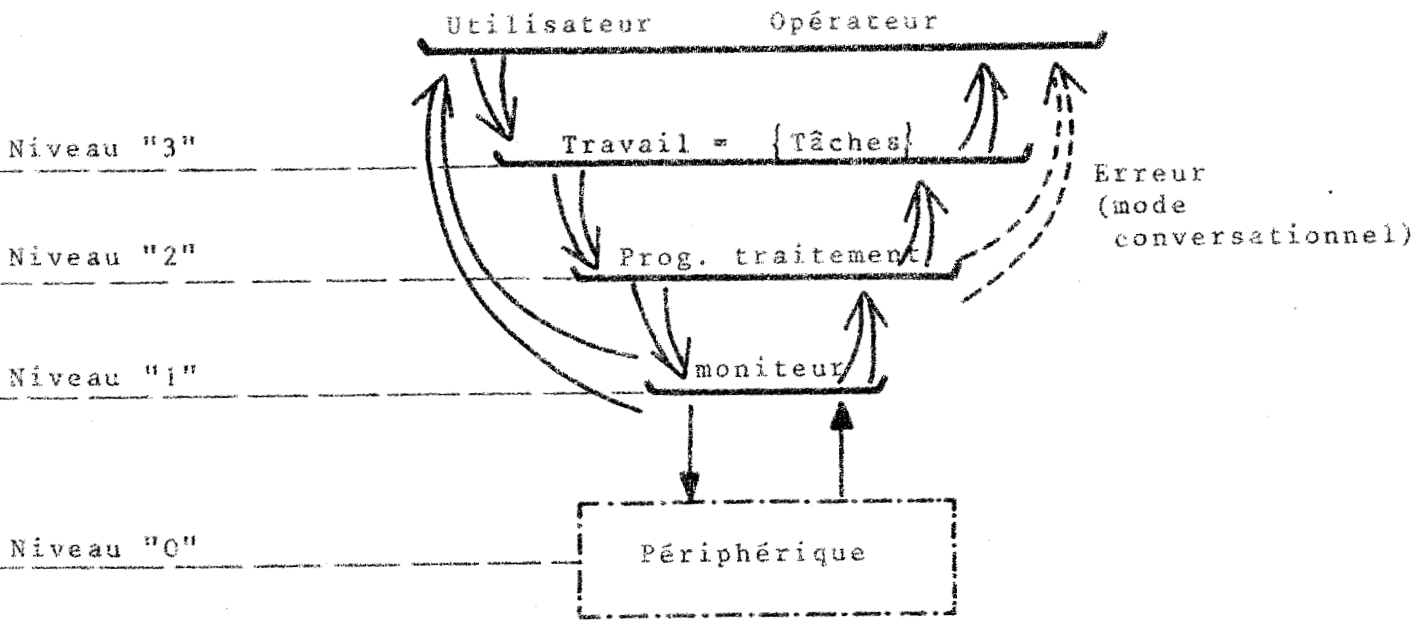


Fig. 29

Les niveaux de contrôle.

Nous sélectionnons le potentiomètre par un numéro, le même que celui qu'il porte sur la machine, concatené à celui de la machine.

Le potentiomètre 108 représentera le potentiomètre n° 8 de la machine n° 1 mais si nous demandons le numéro 008 nous voulons afficher en fait deux potentiomètres, le chiffre "0" correspondant au traitement sur les deux calculatrices.

Pour afficher à partir de la console, nous aurons alors une syntaxe du type suivant :

AP - 208 = 9999

AP correspond à l'affichage des potentiomètres

208 sélectionne le potentiomètre n° 8 de la calculatrice n° 2

9999 est un nombre décimal de quatre chiffres au plus, entier, représentant en fait une fraction de l'unité.

La demande d'affichage d'un potentiomètre non opérationnel ou défectueux implique une erreur qui peut provoquer l'abandon du travail en mode non conversationnel. Cette erreur est facile à éviter en consultant au préalable la table de tous les éléments en défauts, qui est fournie à l'aide d'une clé interprêtée par le programme spécifique, ou reconnue par le moniteur comme une fonction système particulière. Elle entraîne la consultation des tables stockées sur disque et leur édition. Seul le programme de test par l'intermédiaire du moniteur en a le contrôle effectif et peut la remettre à jour. Autrement le moniteur en interdit l'accès ; nous voyons à ce niveau qu'il existe un lien étroit entre le moniteur et le programme de couplage. Celui-ci est considéré en fait comme tâche particulière à priorité fixe.

C'est donc au moniteur qu'est dévolu le rôle de "chien de garde". Il peut toutefois lui-même être retiré de la mémoire par le superviseur pour céder la place au compilateur Fortran par exemple. Pour des raisons de place en mémoire, le moniteur ne peut plus alors jouer le rôle de module d'entrée - sortie vis à vis du superviseur, et seul une partie du noyau est résident. En effet, le superviseur a été élaboré par l'entreprise qui a construit le calculateur, et est de par sa structure mal adapté à nos besoins actuels. Nous ne

faisons que compléter les possibilités d'entrée - sortie, en introduisant de nouveaux modules compatibles avec les précédents, et exploitables à partir du langage Fortran. Nous avons vu en effet que l'utilisation en assembleur ne posait plus de problème grâce à l'utilisation de macro opérations.

Dans un tel contexte, l'affichage des potentiomètres s'écrit simplement :

AFPOT (n° machine, n° potentiomètre, valeur)

soit par exemple AFPOT (1 , 08 , ADRES)

la mémoire ADRES contiendra la valeur à afficher.

En Fortran, nous obtenons encore de plus grandes possibilités :

CALL AFPOT (I , I, A** B + C)

ce qui permet alors des itérations pour l'affichage d'une série de potentiomètres, à des valeurs résultant d'un calcul.

La lecture des potentiomètres est du point de vue de la programmation absolument identique :

LP - 208 = 0,9999

Le calculateur édite alors la valeur lue du potentiomètre choisi en décimal. Cette lecture comme nous l'avons vue au chapitre précédent, se fait par l'intermédiaire d'une chaine extrêmement rapide, en mode canal.

La macro instruction correspondante est du type :

LECPOT (n° machine, n° potentiomètre, MEMOIRE)

MEMOIRE indique l'emplacement mémoire où l'on doit entreposer le résultat de la mesure, en binaire.

La correspondance avec l'appel Fortran est immédiate :

CALL LECPOT (paramètre 1, paramètre 2, paramètre 3) avec les mêmes avantages que ceux que nous avons soulignés précédemment.

1.f - Commande des intégrateurs

La commande des intégrateurs peut se faire globalement ou individuellement. Chaque bloc de deux intégrateurs est repéré par un numéro résultant de sa position sur le panneau de cablage de la calculatrice.

Nous avons à lire la sortie des différents amplificateurs ou à commander le mode de fonctionnement des intégrateurs.

La lecture des amplificateurs se fait d'une manière homologue à celle des potentiomètres :

LA - 215 = Lecture Amplificateur n° 15 de la machine 2

Le résultat est fourni dans le même format que pour les potentiomètres.

La macro instruction assembleur correspondante est alors :

LECAMP (n° machine, n° ampli, ADRES)

avec les mêmes conventions que précédemment.

La correspondance avec le sous programme Fortran se fait immédiatement. Comme pour le potentiomètre, il est possible de reconnaître les amplis défectueux en demandant une édition de la table des éléments. Cependant, comme ici nous ne faisons qu'une opération passive (lecture), adresser et lire un amplificateur défectueux ne sera pas considéré comme une faute. Il se peut en effet que la précision de l'élément suffit à l'utilisateur, alors qu'elle a été jugée inacceptable par le programme de test. L'adressage d'un amplificateur inexistant provoque une erreur. Les conditions dans ce cas là, sont analogues à celles du traitement des échanges au niveau des potentiomètres.

La commande des intégrateurs se fait toujours suivant la même syntaxe. Dans le cas de l'assembleur, nous aurons la macro instruction suivante :

COM INT (n° machine, n° intégrateur, commande
temps)

En utilisant cette possibilité, il faut avoir présent à l'esprit que la commande se fait effectivement sur un groupe de deux intégrateurs.

Nous n'imposons pas de contraintes au niveau de la programmation, si ce n'est celle des translations en mémoire. Si certaines précisions ou options ne sont pas définies pour le déroulement particulier d'un programme, elles sont prises automatiquement. Ainsi, il se peut très bien que par erreur

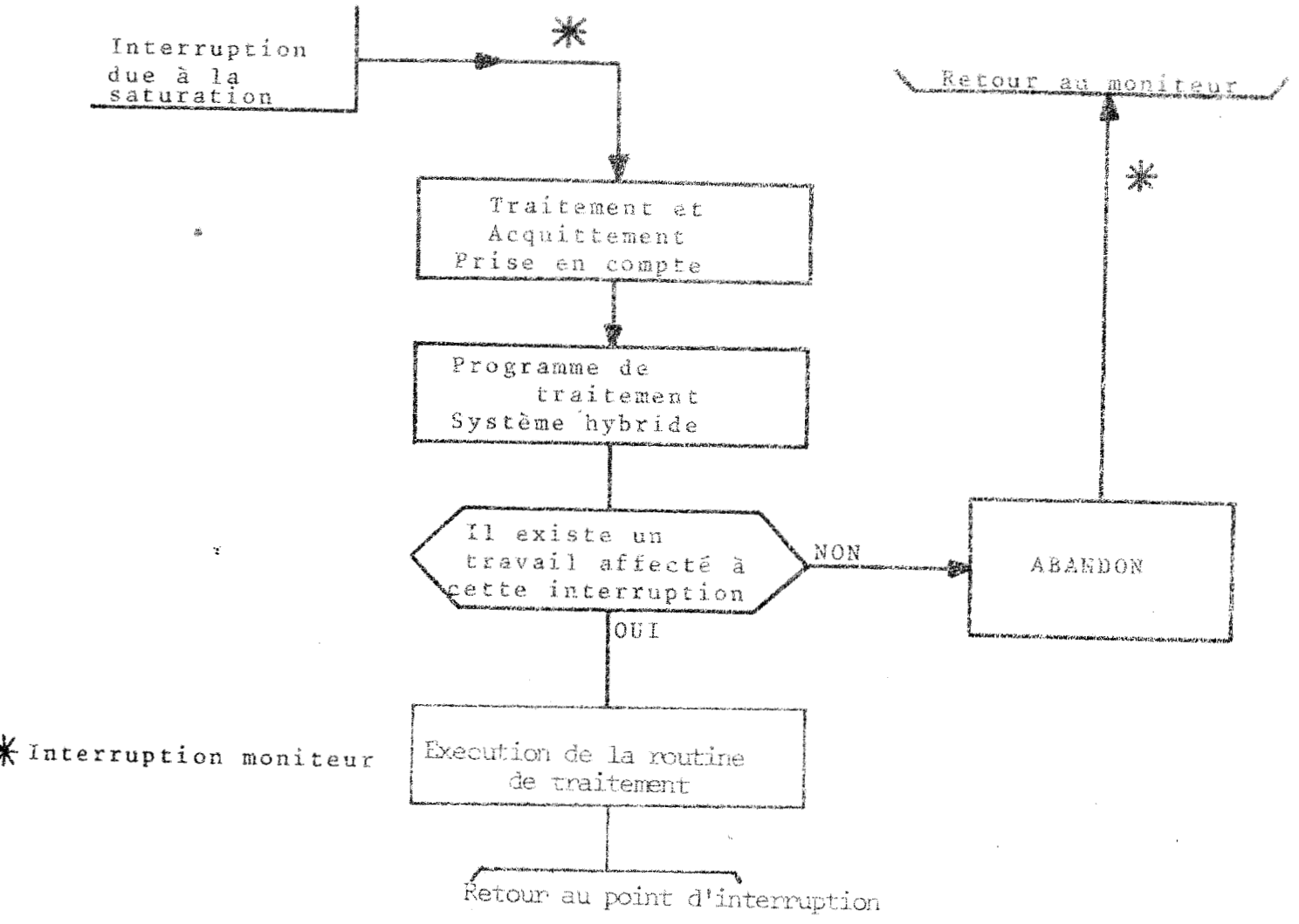


FIG. 30

Saturation

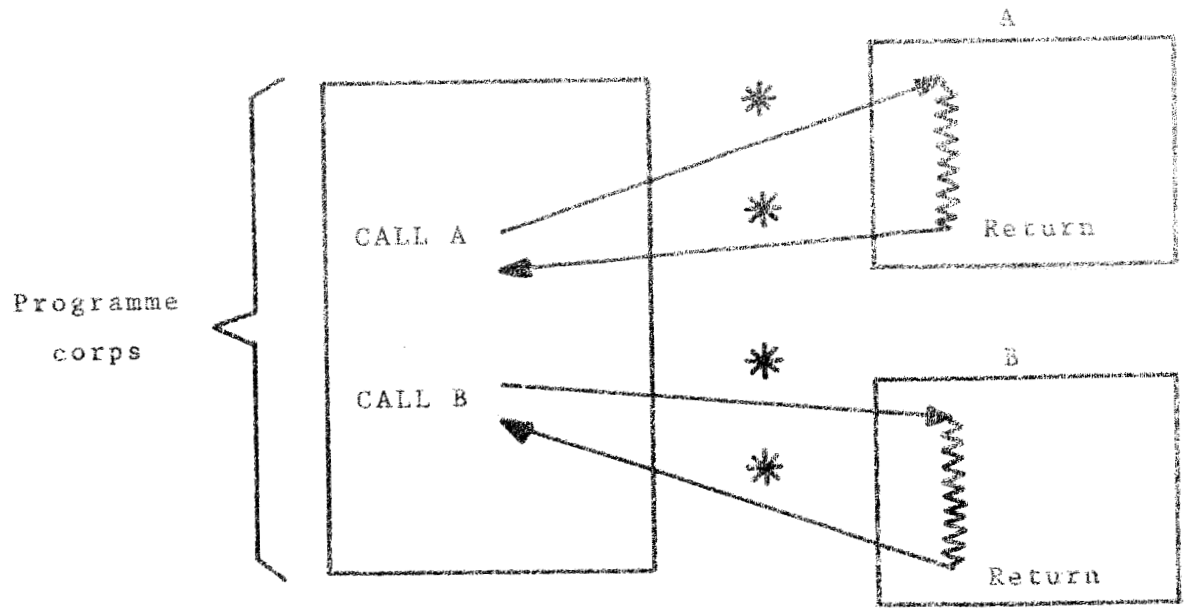
on n'ait pas attribué de travail à une interruption horloge par exemple. Le moniteur alors traite cette interruption mais ne poursuit pas le traitement.

Au niveau du programme de traitement, il en va de même.Prenons l'exemple particulier de la saturation d'un amplificateur. Le système est mis en gel automatiquement ; il lit le mot d'état des machines, repère le niveau de l'amplificateur en défaut. Le programme utilisateur doit alors fixer la conduite à tenir, si une routine de traitement a été prévue, elle est exécutée sinon le travail est abandonné. (Fig. 30).

2.2 - Programmes bibliothèques

La bibliothèque consiste en une série de programmes d'utilisation courante. Pour éviter la saturation de la mémoire, il faudrait que ces sous programmes puissent être réentrants, mais du fait de la structure particulière de la machine utilisée, qui offre des avantages par ailleurs, très peu le sont. Certains sont des programmes spécifiques au couplage, tels que la génération de fonctions, certaines transformations souvent utilisées (transformée de Fourrier, FFT, etc...), génération de nombres aléatoires etc..., d'autres sont des programmes d'utilisation courante, tels que les conversions de nombres, les calculs en double précision etc.... Chaque programme est appelable par l'intermédiaire du moniteur en donnant le nom en paramètre. Ceci n'intervient évidemment que pour une utilisation en assembleur, le superviseur Fortran gérant lui-même la mémoire et pratiquant la gestion dynamique des programmes, Un appel par CALL provoque son chargement automatique en mémoire.

En Fortran, nous avons donc l'organisation de la Figure 31 :



✱ Appel moniteur

Les routines A et B peut occuper en mémoire le même emplacement.

Fig. 31

Les appels de routines

L'inconvénient d'un tel système est qu'une fois utilisé, le programme A est évincé de la mémoire et la place est libre pour installer une routine B éventuelle. Si bien que dans le cas d'appel fréquent d'une chaine de sous programmes, le temps passé lors des transferts de contrôle devient prohibitif.

Le moniteur ne présente pas cet inconvénient, la gestion de la mémoire peut se faire en partie au niveau du programme en cours.

L'organisation est alors celle de la figure 32 :

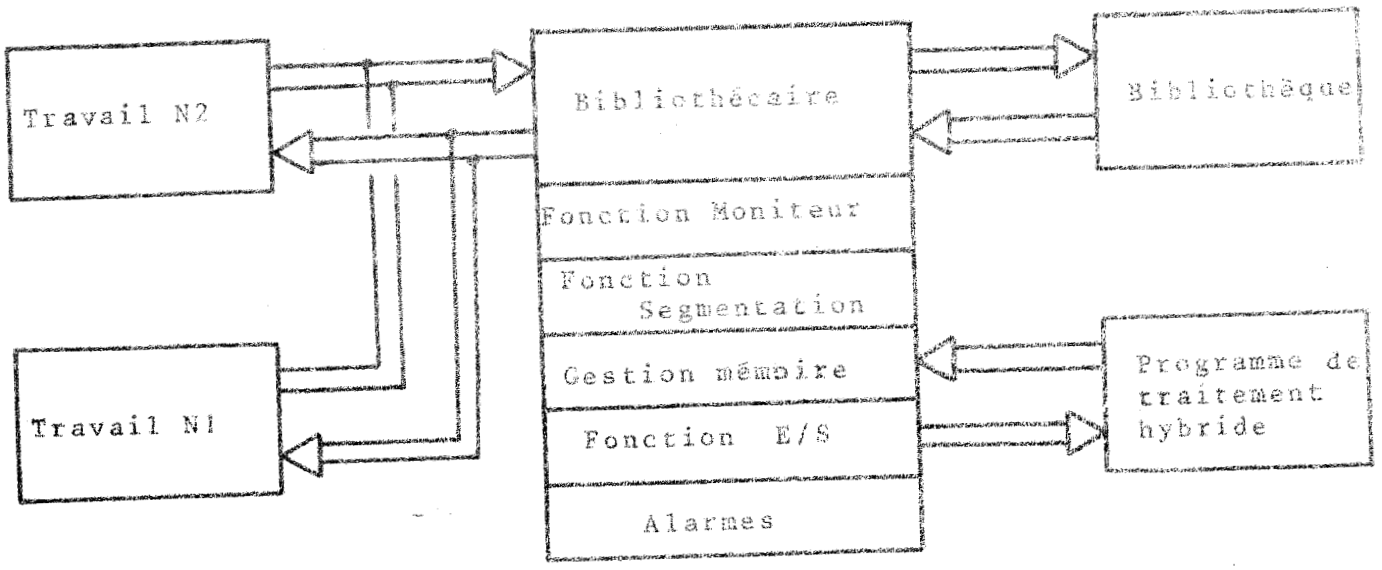


Fig. 32

Gestion du Moniteur

Le rôle du bibliothécaire est de rechercher les programmes sur disques aussi bien les modules du système, pour certaines fonctions nécessitées par le mode conversationnel, que les programmes d'intérêt général. Un contrôle est effectué pour connaître l'identité du demandeur. Le moniteur est en effet privilégié et toutes les fonctions lui sont accordées. Un travail quelconque se verrait refuser des modules appartenant au moniteur.

2.3 - Les tests

Des contrôles au niveau des sorties, aussi bien avec le moniteur qu'avec le programme de couplage, ont pour but de détecter toute incompatibilité entraînant un échange impossible, occupant le périphérique inutilement pendant un temps important ; tout ceci suppose un fonctionnement correct des périphériques. Il importe donc de faire des tests à plusieurs niveaux :

- au niveau des programmes de traitement des échanges sur le périphérique

- au niveau du périphérique lui-même.

Pour mettre au point le programme de traitement, il suffit de simuler le fonctionnement de la machine concernée. Dans le programme d'échanges avec les machines hybrides, nous utilisons une télétype par laquelle nous envoyons les commandes. Le programme de test renvoie alors les différents mots d'état, le contenu de certains registres et simule la sortie. Ainsi il est possible de détecter un mauvais fonctionnement au niveau des routines de méthode d'accès.

Le test des machines hybrides comporte d'abord une vérification de l'interface, puis une analyse de la précision des calculatrices elles-mêmes.

L'étude des circuits de l'interface se fait par bloc de composants à l'aide d'un bac de test. Chaque carte comprenant les circuits est enfichée dans une loge déterminée du bac de test. Le calculateur envoie alors une série de

signaux et teste les sorties. Les valeurs correctes sont tabulées et toute discordance entre la valeur effective et la valeur prévue provoque l'édition d'un journal indiquant le bloc de composants ou le composant en défaut.

Cette méthode est certes hors ligne mais permet une détection systématique des défauts qui seraient plus difficiles à déceler en ligne.

Au niveau des machines hybrides, deux possibilités se présentent : nous pouvons vérifier le bon fonctionnement de chaque élément : ampli, potentiomètre, etc... ou bien prévoir un montage qui mette en jeu le maximum de modules dans des conditions très variées. On envoie alors sur ce système cablé des valeurs fixées à l'avance et l'on prend des mesures en des points précis permettant une comparaison des résultats escomptés et réels (Fig. 33).

Il est aussi possible de combiner les deux méthodes en faisant d'abord le test global puis si nécessaire le test des éléments.

3 - CONCLUSION

En nous préoccupant de satisfaire tous les besoins spécifiques hybrides, nous nous orientons vers l'élaboration d'algorithmes, mais nous ne devons pas perdre de vue que les améliorations des différentes possibilités techniques modifieront l'orientation du software hybride.

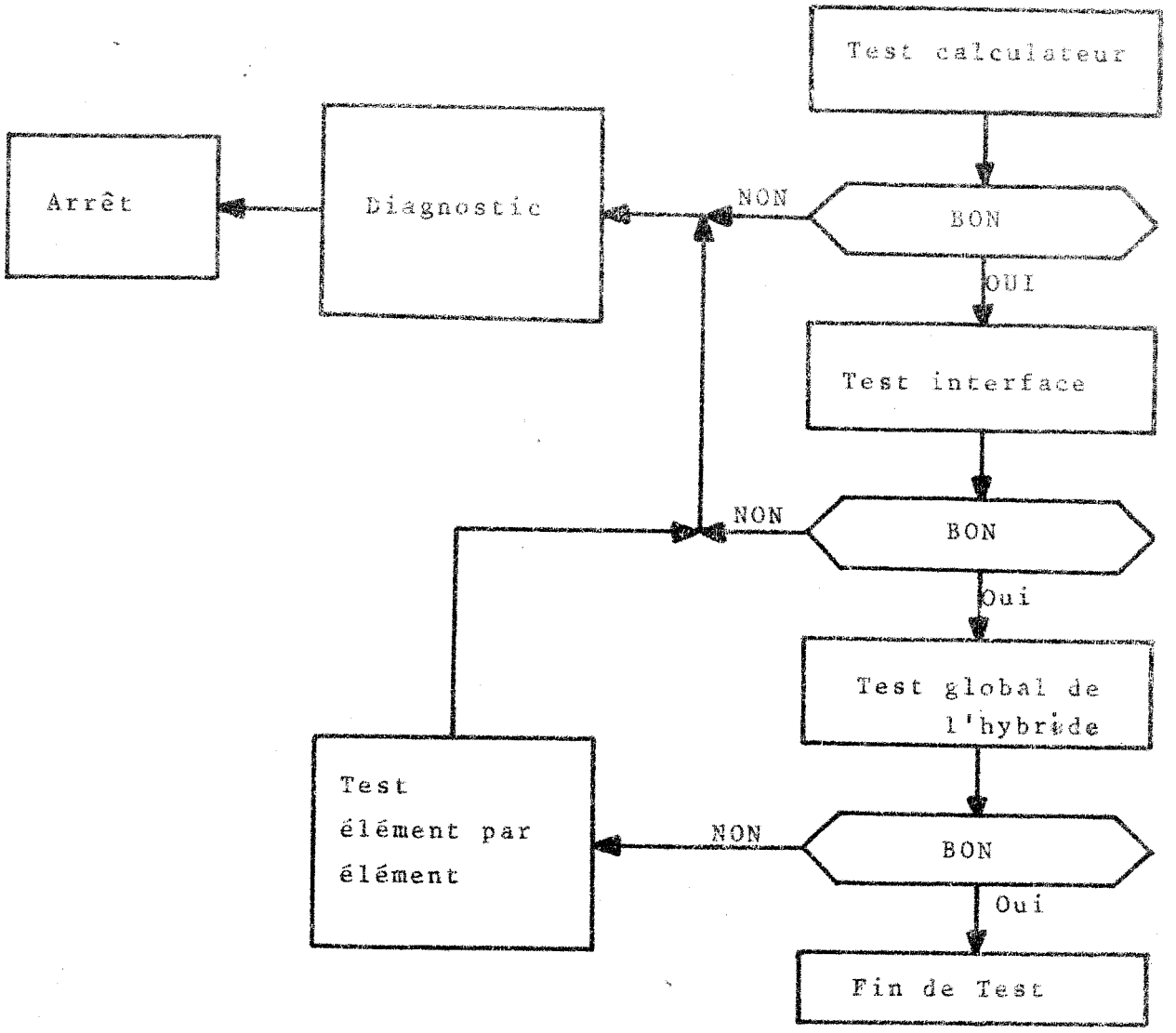


Fig. 33

Les tests.

Nous pouvons constater en effet que le coût hardware diminue régulièrement, tandis que la tendance est inverse pour ce qui concerne la programmation.

Nous devons d'autre part faire en sorte que dans les problèmes hybrides, il soit possible d'éliminer les redondances et les calculs non nécessaires, d'augmenter les possibilités pour travailler avec une précision accrue : un système d'interruption et le partage des ressources du calculateur éliminent les pertes de place mémoire dues à une mauvaise estimation des structures.

L'accroissement des capacités du calculateur numérique et les nouvelles possibilités de transmission de données imposent un ensemble dans lequel plusieurs problèmes peuvent opérer simultanément. Autrement dit, on aura un partage du calculateur entre des travaux travaillant en temps réels (foreground) et d'autres en série continue (batch en back ground).

Le problème des entrées - sorties nécessaires à la communication entre l'analogique et le numérique devient plus aigu au fur et à mesure que les structures deviennent moins rigides. Les langages spéciaux doivent réduire les efforts de programmation en évitant un accroissement trop important du prix du software et de ses applications.

On cherche donc à réaliser un très grand nombre de fonctions,plutôt par hardware que par software, dans la mesure du possible afin de simplifier la programmation notamment lors de la lecture ou l'écriture. Une autre possibilité est d'utiliser des minicalculateurs à mémoire morte microprogrammés et spécialisés dans la réalisation des fonctions particulières.

A plus longue échéance, avec l'accroissement des vitesses de calcul et de la capacité, un seul calculateur digital commandant un grand nombre de calculatrices analogiques et travaillant en temps partagé deviendra une configuration normale.

D'une manière pratique, nous cherchons, au cours des échanges, à minimiser le temps d'occupation de l'unité centrale, mais nous devons aussi essayer de réduire dans la mesure du possible le temps d'opération de la machine hybride. Si à partir du calculateur numérique il est possible de travailler en multiprogrammé, ceci n'est pas un fonctionnement qui lui est propre et les machines que nous utilisons par exemple peuvent avoir un fonctionnement hiérarchisé, l'une fonctionnant en mode "maître" et d'autres en mode "esclave".

Pour ce qui nous concerne, nous ne sommes pas encore au stade du multiprocesseur composé de plusieurs calculateurs. Nous cherchons à améliorer le contenu de la bibliothèque en fournissant un grand nombre de routines offrant des possibilités intéressantes, en permettant des fonctions hybrides particulières.

Le software sera développé pour aider à la mise au point des programmes, puis les besoins s'en faisant sentir, il sera possible de mettre au point un interprêteur hybride, qui tout en utilisant les caractéristiques communes à tous les langages de ce genre, pourra permettre une utilisation plus complète des machines couplées ; nous aurons alors à ce stade atteint la troisième étape définie au chapitre premier.
Il sera alors possible d'améliorer les performances en se définissant un nouvel ensemble d'étapes dont la dernière peut consister en un système multiprocesseur, ou toute autre chose suivant l'évolution des techniques hybrides.

PARTIE IV

EXEMPLES

ORGANISATION D'UN PROBLEME

1 - Introduction

Ce chapitre a pour but de montrer sur quelques exemples les possibilités des deux machines couplées.

Nous abordons en première partie une étude générale, mettant en évidence quelques principes que nous appliquerons lors de la résolution de problèmes plus particuliers tels que la résolution d'équations récurrentes, ou la simulation appliquée à une identification.

1.1 - Généralités

A partir des équations qui définissent le phénomène à étudier, il convient de trouver un modèle mathématique exact ou approché qui puisse être pris en compte par la machine. Ce travail consiste à appliquer les méthodes de résolution spécifique au calcul hybride telle que les méthodes d'optimisation dynamique, la méthode des caractéristiques pour les équations aux dérivées partielles , la méthode d'approximation fonctionnelle, etc...

En général, les différents rôles des calculateurs découleront directement de la méthode choisie pour le résoudre.

Nous avons vu que les principales tâches dévolues au calculateur numérique sont la mémorisation de fonction, le calcul algébrique ou intégral, ou parallèle ou en série avec la machine analogique, la décision et la commande.

La partie logique du calculateur analogique, sera utilisée en général comme organe de synchronisation entre le programme numérique et le programme analogique ; de plus elle sert d'intermédiaire pour l'exécution sur la machine analogique de certaines décisions prises sur la machine numérique.

Notons au passage la différence essentielle entre les décisions logiques séquentielles prises par le calculateur numérique et celles parallèles prises par la logique hybride.

Le travail particulier sur chaque partie, numérique, logique, analogique nécessite soit un organigramme, soit un schéma particulier selon les méthodes propres à chaque calculateur.

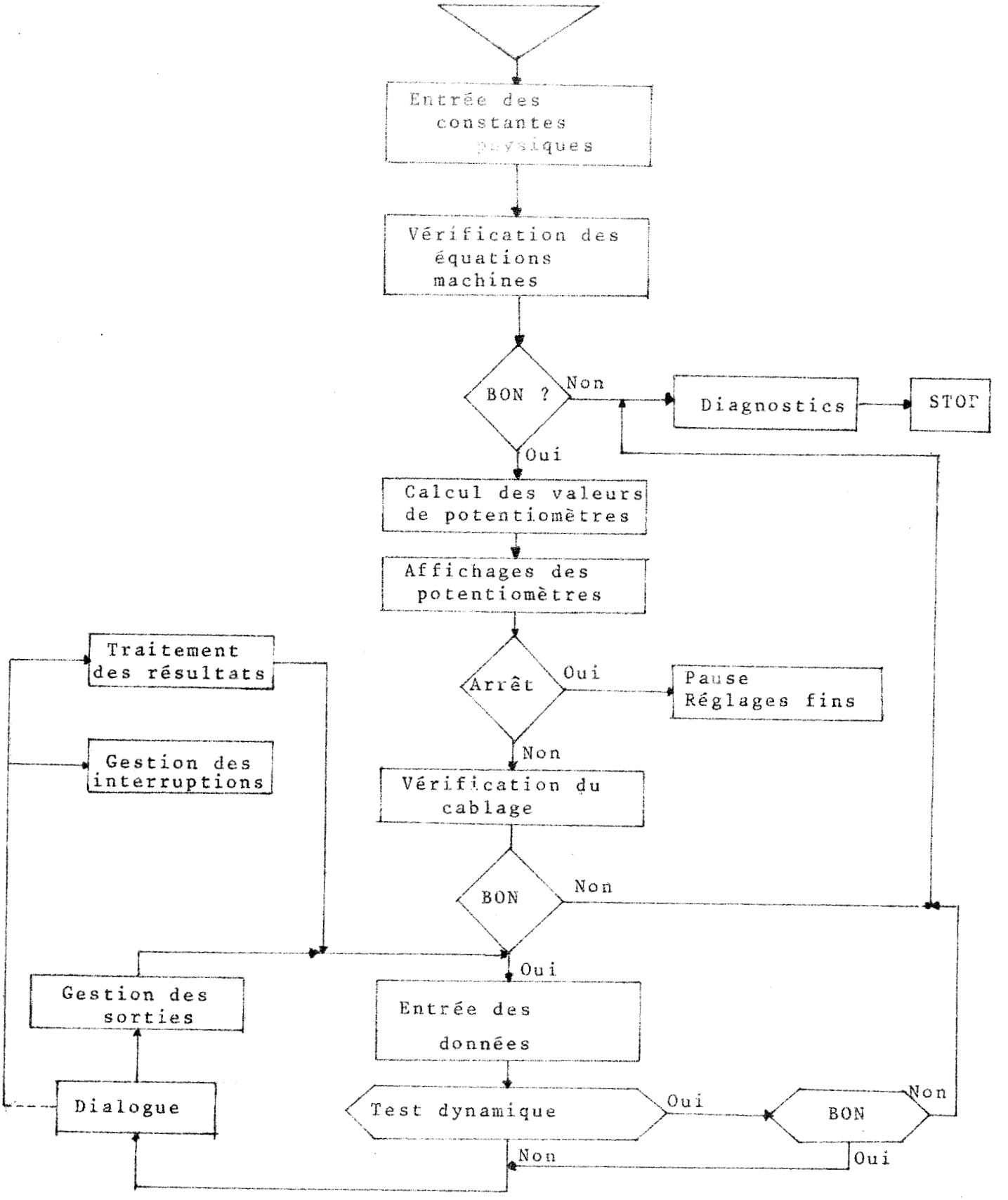
Il est cependant nécessaire de concevoir une organisation générale, un organigramme hybride (figure 1)

1.2 - Vérifications diverses et réglages

Cette étape permet de calculer et de régler les différentes valeurs des potentiomètres et d'effectuer certaines vérifications du modèle simulé. Les grandeurs de travail des physiciens, c'est-à-dire les paramètres physiques du problème, doivent être des grandeurs d'entrée du système.

Il est ensuite nécessaire de vérifier si les équations proposées à la machine correspondent bien aux équations physiques. Dans ce cas, on calcule les valeurs des potentiomètres et on les règle. Il est intéressant de prévoir un réglage éventuel à la main dans les cas délicats.

On vérifie ensuite que le câblage réalisé correspond bien aux équations données à la machine. C'est une vérification statique. La machine analogique est mise en mode "vérification statique". Les opérations réalisées en calcul analogique sont effectuées également en numérique et l'on compare les résultats. S'il y a erreur, les équations non


Entrée des constantes physiques
Vérification des équations machines
BON ?
Non
Diagnostics
STOP
Oui
Calcul des valeurs de potentiomètres
Affichages des potentiomètres
Traitement des résultats
Arrêt
Oui
Pause Réglages fins
Gestion des interruptions
Non
Vérification du cablage
BON
Non
Oui
Gestion des sorties
Entrée des données
Dialogue
Test dynamique
Oui
BON
Non
Non
Oui

vérifiées et les amplificateurs correspondants sont notés sur la machine imprimante et le programme numérique passe en pause pour permettre éventuellement de modifier le cablage.

1.3 - Résolution

1.3.a - Calcul ou entrée des données dynamiques

On entend par entrées dynamiques les grandeurs de commande du système simulé. En d'autres termes, on veut en général connaître la réponse du système lorsqu'on lui applique un certain nombre de signaux d'entrée. Si l'on choisit pour la simulation d'un système multivariable une représentation discontinue d'une des variables, par exemple le temps, on doit connaître la valeur des grandeurs d'entrée pour chaque valeur du temps. Cela peut être soit une donnée, soit le résultat d'un calcul intermédiaire réalisé par le calculateur. De plus, l'état du système au temps initial doit être connu. Cela peut nécessiter un calcul préliminaire qui peut être lui-même hybride, notamment quand la connaissance des conditions initiales nécessite la convergence d'une série d'itération (cas des conditions finales connues).

1.3.b - Vérification dynamique

Il est très utile de pouvoir vérifier la logique du programme numérique ou programme de dialogue. Dans la phase de calcul où les calculateurs travaillent de concert, il est nécessaire de prendre un pas en "temps machine". Ce n'est que pour des valeurs discrètes de ce temps machine qu'il pourra y avoir conversation. Notons que ce temps machine est proportionnel à une variable quelconque du système. C'est suivant cette variable que se font les intégrations continues en calcul analogique, son choix ne dépendant que de la méthode de résolution retenue.

Pour vérifier la logique du programme numérique, il est donc nécessaire d'effectuer un pas en temps machine. On peut alors voir si les différent transferts ou opti isations élémentaires se sont réalisées normalement. Certains programmes permettant cette vérification sans utiliser la machine analogique.

1.3.c - Programme de dialogue

Machine numérique et machine analogique doivent opérer ensemble ; il est donc nécessaire de synchroniser les deux programmes. Si en calcul analogique il est possible de garder une variable continue, pour le calcul numérique, toutes les grandeurs seront réduites à des valeurs discrètes. La résolution analogique supposera un état déterminé des grandeurs discrètes puis un nouveau calcul fera progresser les indices d'une unité. La synchronisation se fera donc à l'aide d'horloges propres ou connues à chaque calculateur, générées à partir de compteurs et permettant d'envoyer des interruptions sur le calculateur travaillant en mode esclave.

1.3.d - Programme d'interruption

Le programme de dialogue peut être aussi interrompu à tout moment par une décision du programme logique résultant en général d'un évènement analogique. Par exemple, cela peut être le passage à un changement d'état survenant aléatoirement en nécessitant le calcul et le réglage de nouveaux paramètres.

1.3.e - Le programme de gestion des sorties

Il sera chargé de gérer toutes les sorties telles que l'impression sur l'imprimante l'envoi des résultats sur disques, exploitation de la table traçante, etc...

1.3.f - Traitement des résultats de calcul

Il s'agit ici d'un traitement en ligne, c'est-à-dire que les résultats du calcul sont analysés en vue de permettre un nouveau cycle, par exemple la recherche des conditions initiales en fonction des conditions finales.

2 - Traitement d'un exemple hybride

Le développement des techniques de commande amène à considérer de plus en plus des systèmes décrits par des relations de récurrence, tels que les systèmes échantillonnés à période fixe ou variable, les systèmes non conservatifs décrits par des équations différentielles linéaires à coefficient périodique, etc...

Le modèle mathématique se présente sous la forme d'une récurrence non linéaire.

(1) $\underline{Y}_{n+1} = f(\underline{Y}_n)$ où $\underline{Y}_n$ est le vecteur d'état du système à un instant t_n et $\underline{Y}_{n+1}$ le vecteur d'état à l'instant t_{n+1}.

Les propriétés des solutions de ce type d'équation sont d'une grande importance par l'analyse des systèmes qui leur correspond. La connaissance de ces propriétés permet alors de mieux comprendre certains comportements particuliers de ces systèmes et de dégager des méthodes utiles d'analyse et de synthèse.

Nous allons montrer à travers un exemple, et à l'aide du calculateur hybride, comment il est possible d'étudier de manière systématique des propriétés telles que la recherche des états d'équilibre stables et instables, la détermination du domaine complet d'attraction d'un état d'équilibre stable, la détermination des régimes oscillants.

On peut montrer que l'étude de ces différentes propriétés sur la récurrence (1) se ramène à la détermination de tous les états d'équilibre stables ou instables, chacun de ces états d'équilibre étant donné par la résolution d'un système d'équations algébriques.

2.1 - Stabilité des systèmes modulés en largeur

La difficulté d'étude de ces systèmes provient de leur non linéarité. Un échantillonneur à modulation d'amplitude classique, est essentiellement un organe linéaire dans le sens qu'entrée et sortie sont gouvernées par le principe de superposition. Cependant un échantillonneur à modulation de largeur doit être considéré comme un élément non linéaire, le principe de superposition entre la sortie et l'entrée ne s'appliquant pas.

Pour un échantillonneur à modulation de largeur, l'état du système à la fin de la période d'échantillonnage est la somme d'un vecteur dépendant de l'état du système au début de la période et d'un vecteur dont la longueur et la direction sont facteurs non linéaires de la largeur de l'impulsion.

Par une étude approfondie de la stabilité, le système à modulation de largeur des impulsions doit être traité comme un système essentiellement non linéaire, sans possibilité d'approximation linéaire.

2.1.a - Représentation du système

Nous considérons ici l'asservissement de position moteur commandé en modulation de largeur, avec une compensation tachymétrique (figure 2).

La commande s'effectue par un signal d'amplitude constant, du signe du signal d'erreur défini aux instants d'échantillonnage.

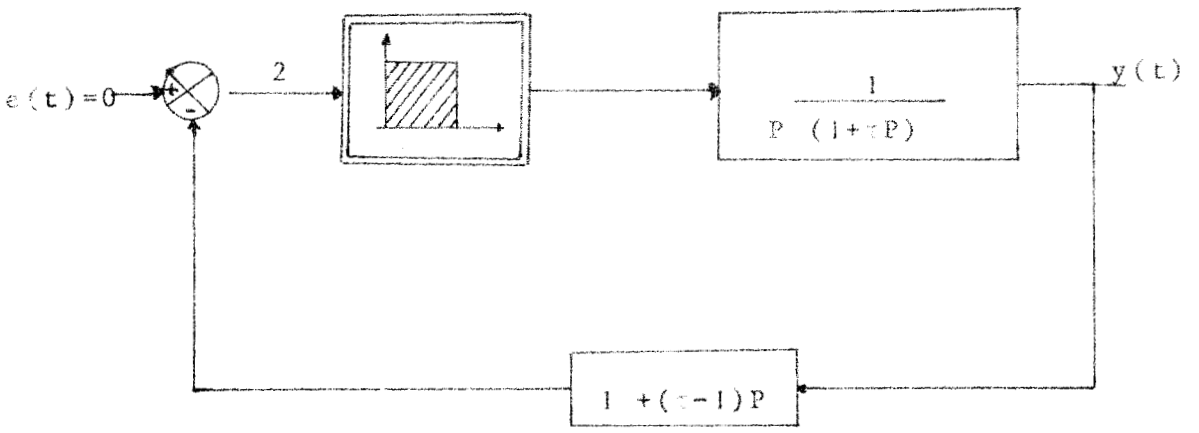


Figure 2

Asservissement

T désigne la période d'échantillonnage, Tin = k|εn| la durée de l'impulsion de commande en régime non saturé, la constante de temps du moteur.

λ caractérise l'effet de retour.

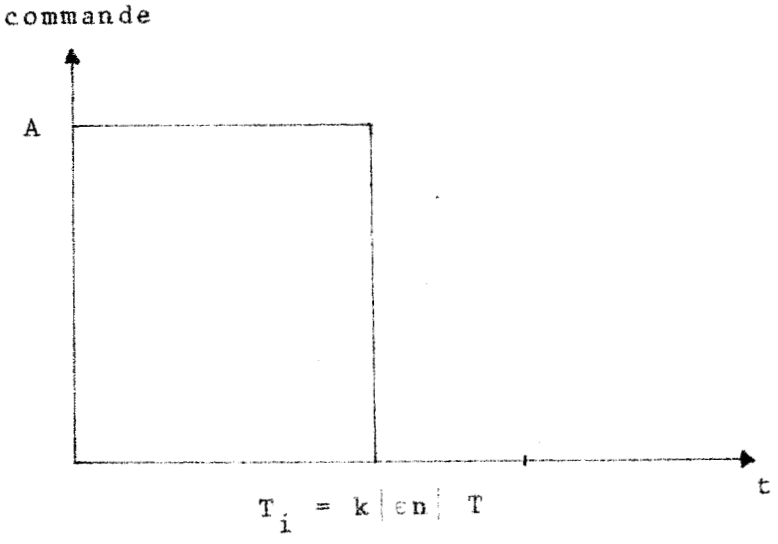


Figure 3

Modulation de largeur

2.1.b - Méthode de simulation

Il n'est pas dans notre intention d'établir toute la théorie relative à de tels systèmes, nous nous bornerons à mettre en évidence les possibilités de l'ensemble hybride.

Nous désirons effectuer un balayage paramétrique et étudier le stabilité d'un moteur, en régime saturé ou non nous ferons varier suivant différentes méthodes, les conditions initiales et déterminerons l'ensemble des points atteints dans le plan de phase (Y,Y').

Nous étudierons l'influence du coefficient λ du retour tachymétrique ainsi que celle de paramètre $kA=RO$ (figure 3) sur la stabilité.

Le moteur ne présente pas en temps que système continu de difficultés lors de la simulation. Puisque nous considérons un système autonome $(e(t) = 0)$, nous avons

$$\varepsilon(t) = -Y_2(t) = -|1 + (\tau-\lambda)\, p\, Y$$

Cette valeur est envoyée à l'aide d'un convertisseur dans le calculateur numérique qui établit la commande à envoyer : $\varepsilon_2 = \alpha A$ pendant un temps $Ti = k\varepsilon n$

Nous avons alors le schéma de simulation de la figure 4.

Nous avons envisagé plusieurs possibilités sur la méthode de balayage dans le plan de phase Y, Y'. La première consiste à étudier systématiquement les différents points du plan, et à constater s'il y a stabilité ou non.

Mais cette méthode est extrèmement lourde, tout en utilisant un temps considérable ; si d'autre part nous voulons étudier la stabilité par rapport à certains paramètres, par exemple kA, elle se révèle inutilisable. Il est alors préférable de balayer le plan, y, y', suivant une direction définie à priori au départ de la recherche et de modifier par la suite le contour dans le cas d'une instabilité, afin de déterminer le cycle limite.

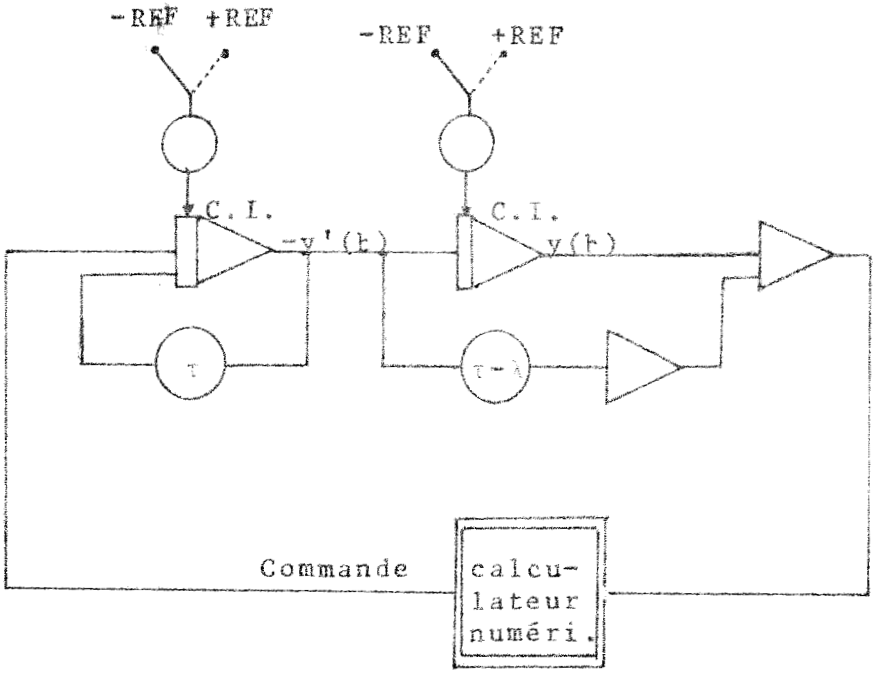


Figure 4

Schéma de simulation

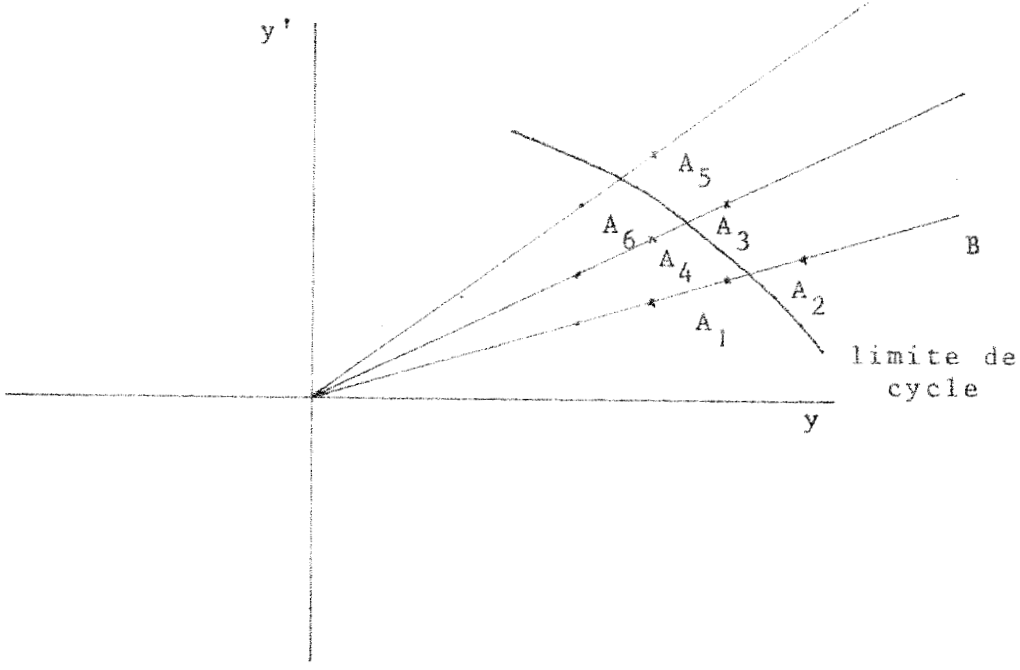


Figure 5

Balayage dans le plan y, y'

Le balayage en coordonnée polaire semble alors meilleur. Définissons une direction à priori AB, et faisons varier le point écrivons y, y' sur A B de A vers B; passons de A_1 à A_2, nous observons une instabilité ; α est alors incrémenté et la recherche se fait entre A_3 et A_4, le cycle se renouvelant nous obtenons alors la représentation du cycle limite avec une précision dépendant du pas choisi sur α et sur les couples de points consécutifs A_1 et A_2. Pour déterminer le cycle limite nous pouvons utiliser une méthode du genre de celle de Little par la théorie des graphes.

2.2 Résultats

Nous avons quatre paramètres de discution :

k

A

λ

T

Dans certains domaines, notamment pour les valeurs relativement faibles de k et de A, nous pouvons raisonner en fonction de kA = BO, les résultats état sensiblement les mêmes pour BO = constant ; comme on peut le voir sur les enregistrements en fin de chapitre (figure A3)

Par $\lambda = 0,25$

$BO = 1$

$T = 1,$

nous avons concordance entre les résultats pratiques et les résultats théoriques. Le système est stable.

Pour BO = $2,5$ nous avons des oscillations, que ce soit avec h = 2,5 A = 1 ou A = 2,5 et h = 1.

Dans un premier cas nous étudions la nature du cycle obtenu et l'évolution de la trajectoire dans l'espace des phases y, y'. Nous obtenons alors en régime saturé (k grand) les cycles de la figure A 5 (annexe).

Les notations sont les suivantes :

I : condition initiale sur y

I': condition initiale sur y'

Nous avons étudié ensuite, en suivant l'évolution du paramètre λ , les types de trajectoires puis la stabilité illimité. Par ce faire, nous faisons varier le paramètre T/ et relevons les valeurs de K indiquant un cycle. La liste des résultats est obtenue sur une télétype par laquelle on peut commander les nouvelles valeurs, ou laisser l'évolution automatique. Nous en déduisons l'évolution des points dans le plan BO plan BO, λ.

Ceci ne représente évidemment qu'une ébauche d'étude des oscillations limite d'un système de deuxième ordre, mais permet d'apercevoir la grande souplesse d'exploitation offerte par le système, pour commander l'évolution des paramètres.

Nous ne nous sommes pas attachés à la précision des résultats, les cycles limites n'étant en fait "perçus" par le calculateur qu'à partir d'un certain seuil, mais à l'allure générale du phénomène. Cependant une exploitation plus précise de cette simulation a permi de vérifier la concordance entre les études physiques et les conclusions de l'étude théorique

Le déroulement des opérations est représenté par l'organigramme de la figure A 6 (annexe).

Il est possible de balayer systématiquement le plan en faisant varier tel ou tel paramètre, ou même en introduisant les valeurs autour desquelles on désire travailler à partir du clavier de la télétype.

Nous envisageons les balayages suivant les différents paramètres en annexe.

Le système se caractérise par une simulation relativement aisée de par sa simplicité initiale (système du deuxième ordre), mais le mode d'utilisation hybride permet

une logique parallèle pratiquement inexistante ; notamment on n'a pas de compteur, d'horloge, de temporisation, etc...

Ce mode de fonctionnement est particulièrement intéressant dans notre cas, où seul l'occurence d'un évènement (présence de cycle, instabilité) peut être pris en compte. Ainsi le calculateur peut noter uniquement les cas de cycle et éditer dans un tel cas la valeur des différents paramètres, commander une table traçante, si bien qu'il sera possible d'éditer par la suite et de visualiser l'ensemble des conditions initiales donnant lieu à un cycle stable ou instable.

En conclusion, nous pouvons résumérles qualités des appareils en les repérant dans le tableau 6 . Certains avantages du calcul hybride par rapport au calcul numérique sont en fait dus au fonctionnement en temps réel de l'ensemble hybride.

La solution par calculateur hybride apparait la meilleure pour ce problème, par rapport aux solutions purement numériques ou analogiques.

	Calculateur scientifique numérique	Calculateur analogique	Calculateur hybride
Intégration	I	A	A
Interation	A	I	A
Non linéarité	A	I	A
Changement de paramètres	I	A	A
Mémorisation	A	I	A
Facteur d'échelles	A	I	I
Visualisation directe	I	A	A
Sortie des résultats	A	I	A
Paramètres et conditions initiales	A	I	A
Logique de recherche	A	I	A
Temps de calcul global	I	I	A

A Avantage

I Inconvénient

Figure 6

Etude comparative

3 - EXEMPLE 2

Nous allons étudier maintenant un problème simulé à la fois sur une calculatrice hybride EAI 580 et un calculateur 360 CSMP. Nous allons reprendre cet exemple, en l'adaptant au système hybride que nous avons réalisé ; nous pourrrons de cette manière comparer les résultats de la publication (17), avec ceux que nous avons obtenus.

3.1 - Définition du problème

L'équation que nous envisageons, traite d'un écoulement de fluide suivant la loi de Poiseuille (17).

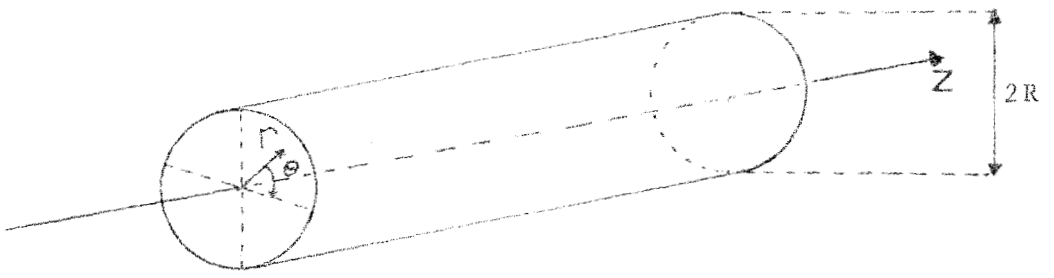


Figure 7

Fluide dans une canalisation - Notations

Pour un fluide circulant dans une canalisation, nous connaissons sa température externe T_1 et sa vitesse au niveau de l'enveloppe (u = 0).

Nous désirons connaître la vitesse et la température de ce fluide au centre de cet écoulement.

Les coefficients de viscosité (μ) et de conductivité thermique (k) sont des fonctions non linéaires. Nous

prendrons uniquement le cas où μ et k sont des fonctions de la température.

$$\mu = \mu_o \left(\frac{T_o}{T}\right)^2$$

$$k = k_o \frac{T_o}{T}$$

avec $T_o = \frac{T_l + T_m}{2}$ T_m valeur maximale de la température.

La résolution de ce problème nécessite de partir de conditions initiales données et de vérifier que les valeurs finales correspondent aux valeurs théoriques à atteindre. Par un critère de notre choix, nous réajustons, à chaque itération, les conditions initiales du départ.

3.1.1 - Equations du système

Nous pouvons écrire les équations du système :

Equation de continuité : $$\frac{\partial u}{\partial z} + \frac{\partial v r}{\partial r} + \frac{v_r}{r} = 0 \qquad (I)$$

Equation du moment : $$\rho u \frac{\partial u}{\partial z} + \rho v_r \frac{\partial u}{\partial r} + \frac{dp}{dz} = \frac{1}{r} \frac{\partial}{\partial_r} \; r\mu(T) \frac{\partial u}{\partial r} \qquad (II)$$

Equation d'énergie : $$\rho u \frac{\partial h}{\partial z} + \rho v_r \frac{\partial h}{\partial r} - \frac{1}{r} \frac{\partial}{\partial r} \left(k(T) r \frac{\partial T}{\partial r}\right)$$

$$- \mu(T) \left(\frac{\partial u}{\partial r}\right)^2 - u \frac{dp}{dz} = 0 \qquad (III)$$

r et z sont les coordonnées cylindriques

v_r et u, sont les composantes de la vitesse suivant les directions r et z

ρ masse volumique du fluide.

Nous disposons d'un certain nombre de conditions :

* le flux est irrotationnel : $v_r = 0$

$$\frac{\partial u}{\partial z} = \frac{\partial T}{\partial z} = 0$$

* Le fluide est incompressible : $\rho = \text{const.}$

* $dh = cdT + \frac{1}{\rho} dp$

* $\frac{dp}{dz} = \text{constant} = C_1 < 0$

L'équation (I) est vérifiée par ces conditions.

L'équation (II) devient :

$$\frac{dp}{dz} = C_1 = \frac{1}{r} \frac{d}{dr} \left(r \mu (T) \frac{du}{dt} \right) \qquad \text{(IV)}$$

$$\frac{d}{dr} \left(r k (T) \frac{dt}{dr} \right) = - r \mu (T) \left(\frac{du}{dr} \right)^2 \qquad \text{(V)}$$

Les conditions de cet écoulement sont :

$r = 0$ $\qquad \frac{du}{dr} = 0 \qquad \frac{dT}{dr} = 0$

$r = R$ $\qquad u = 0 \qquad T = T_1$

Le travail consiste à trouver les quantités u(0) et T(0), qui sont les conditions initiales du système.

Or $\qquad u(0) = u_m \qquad$ et $\qquad T(0) = T_m$

Nous définissons donc : $\quad \bar{r} = \frac{r}{R} \qquad \bar{T} = \frac{T}{T_1} \qquad \bar{u} = \frac{u}{u_m}$

En reportant dans les équations (IV) et (V), nous obtenons :

$$C_1 = \frac{1}{\bar{r}\ R^2}\ \frac{d}{dr}\ \left(\bar{r}\ \frac{\mu_0\ T_0^2}{\bar{T}^2\ T_1^2}\ u_m\ \frac{d\bar{u}}{dr}\right)$$

Soit $$C_1 R^2\ \frac{\bar{r}^2}{2} = \bar{r}\ \frac{\mu_0\ T_0^2}{\bar{T}^2\ T_1^2}\ u_m\ \frac{d\bar{u}}{dr} + C_2$$

$$\bar{r} = 0 \quad \Rightarrow \quad C_2 = 0$$

d'où :

$$\frac{d\bar{u}}{dr} = \frac{C_1\ R^2}{2}\ \frac{T_1^2}{\mu_0\ T_0^2}\ \frac{\bar{T}^2}{u_m}\ \bar{r}$$

Posons $$\Gamma = \frac{2\ C_1^2\ R^4}{\mu_0\ K_0\ T_1} \qquad \lambda = \frac{T_m}{T_1} \qquad u^* = u\ \frac{\mu_0}{k_0\ T_1}$$

$$\frac{du^*}{d\bar{r}} = \frac{C_1\ R^2}{2} \cdot \frac{1}{\mu_0\ k_0\ T_1} \cdot \frac{T_1^2}{T_0^2}\ \bar{r}\ \bar{T}^2$$

Or $$\Gamma^{1/2} = -\frac{2\ C_1\ R_1^r}{\mu_0\ k_0\ T_1} \qquad \text{(puisque } C_1 < 0\text{)}$$

$$(\lambda + 1)^2 = \left(\frac{T_m + T_1}{T_1}\right)^2 = 4\ \frac{T_0^2}{T_1^2}$$

L'équation se met alors sous la forme

$$\frac{du^*}{d\bar{r}} = -\ 2\ \frac{\Gamma^{1/2}}{(\lambda + 1)^2}\ \bar{r}\ \bar{T}^2 \qquad \text{(VIII)}$$

De même :

$$\frac{d}{d\bar{r}}\left(\frac{\bar{r}}{\bar{T}}\frac{d\bar{T}}{d\bar{r}}\right) = -\frac{\Gamma}{(\lambda+1)^3}\bar{r}^3\bar{T}^2 \qquad \text{(IX)}$$

Les conditions sont : $\bar{r} = 0 \qquad \frac{du^*}{d\bar{r}} = 0 \qquad \frac{d\bar{T}}{d\bar{r}} = 0$

$\bar{r} = 1 \qquad u^* = 0 \qquad \bar{T} = 1$

Nous devons donc rechercher les conditions initiales de ce système $u^*(0)$ et $\bar{T}(0) = \lambda$

Nous poserons pour la simulation $\bar{r} = Kt$

En prenant K = 0,1, la période de calcul sera de 10s. On doit alors avoir $u^* = 0$ et $\bar{T} = 1$.

3.1.2 - Simulation hybride de Type I

A partir des équations VIII et IX, nous pouvons élaborer un schéma de simulation (Figure 8).

Nous constatons que nous devons diviser les quantités $-\frac{\bar{r}}{\bar{T}}\frac{d\bar{T}}{d\bar{r}}$ par $\frac{\bar{r}}{\bar{T}}$

Or, cette opération n'est pas définie par $\bar{r} = 0$

pour $\bar{r} = 0 \qquad \frac{d\bar{T}}{d\bar{r}} = 0$

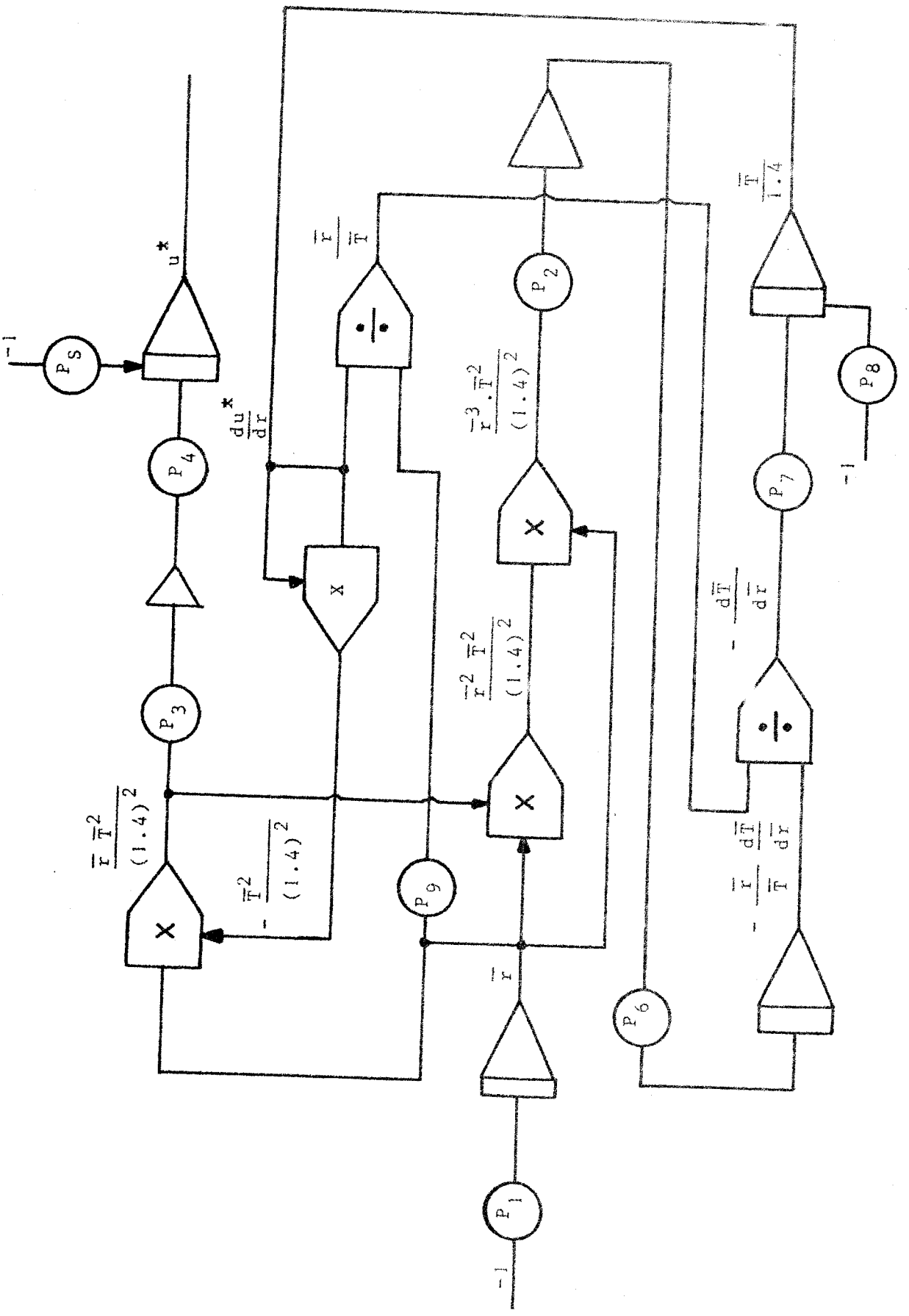


Figure 8

Simulation analogique

- LEGENDE -

$P_1 = 0.1$

$P_2 = \frac{\Gamma}{25}$

$P_3 = \frac{\Gamma^{1/2}}{5}$

$P_4 = \frac{1.388}{(\lambda + 1)^2}$

$P_S = u^* (0)$

$P_6 = \frac{4.9}{(1 + \lambda)^3}$

$P_7 = \frac{1}{14}$

$P_8 = \frac{\lambda}{1.4}$

$P_9 = \frac{1}{1.4}$

Nous devons donc admettre que pour $0 < \bar{r} < r_o$,

$\frac{d\bar{T}}{d\bar{r}}$ reste constant et égal à zéro.

Il faut choisir r_o de manière à n'apporter aucune perturbation au système.

On pourra choisir $r_o = 0,1$

3.2 - Définition de la simulation en hybride de type II

3.2.1 - Recherche du schéma

Si nous considérons le montage de la figure nous constatons que le calculateur peut jouer le rôle de générateur de fonction en élaborant la quantité $\frac{d\bar{T}}{d\bar{r}}$ à partir des informations $-\frac{\bar{r}}{\bar{T}}$ $\frac{d\bar{T}}{d\bar{r}}$ et $\frac{\bar{r}}{\bar{T}}$

Pour $\bar{r} < r_0$ le calculateur par l'intermédiaire de sorties analogiques élabore une tension nulle.

Par $r_0 < \bar{r} < 1$, le calculateur fournit une tension :

$$V = \frac{d\bar{T}}{d\bar{r}} = \frac{\alpha}{\beta}$$

avec $\alpha = \frac{\bar{r}}{\bar{T}} \frac{d\bar{T}}{d\bar{r}}$ $\beta = \frac{\bar{r}}{\bar{T}}$

Nous aurons alors le schéma de simulation donné par la figure 9

Le calculateur a deux tâches à accomplir :

a) Générer sur une période de 10s, le signal $\frac{d\bar{T}}{d\bar{r}}$

Pour cela, il lance une séquence de lecture qui lui définit

$$\bar{r} \quad , \quad \frac{\bar{r}}{\bar{T}} \quad , \quad \text{et} \quad - \frac{\bar{r}}{\bar{T}} \frac{d\bar{T}}{d\bar{r}}$$

Après calcul, il génère le signal $\frac{d\bar{T}}{d\bar{r}}$, puis recommence le cycle.

b) En fin de période, le calculateur "lit" les valeurs de u^* et de $\bar{T}$ et doit, suivant ces valeurs, modifier les conditions initiales

$u^*(0)$ et $\bar{T}(0) = \lambda$.

Cette tâche accomplie, un nouveau cycle de calculs recommence.

Cette séquence se déroulera jusqu'à l'atteinte des conditions finales,
c'est à dire $u^* = 0$ $\bar{T} = 1$ au bout de 10 secondes.

3.2.2 - Recherche du critère d'optimisation (ou de recherche)

Nous considérons les équations VIII et IX, nous constatons que si u^* dépend de $\bar{T}$, $\bar{T}$ ne dépend pas de u^*.

Nous pouvons simplifier l'étude de la manière suivante :

Le paramètre "Γ" étant choisi, nous pouvons déterminer λ tel que $\bar{T}(1) = 1$.

Lorsque λ est trouvé, alors il nous reste à ajuster $u^*(0)$ tel que $u^*(0) = 0$.

Pour cela, il suffit de laisser évoluer le système sans la condition initiale $u^*(0)$.

La valeur de sortie sera l'opposé de la valeur initiale pour $u^*(0)$.

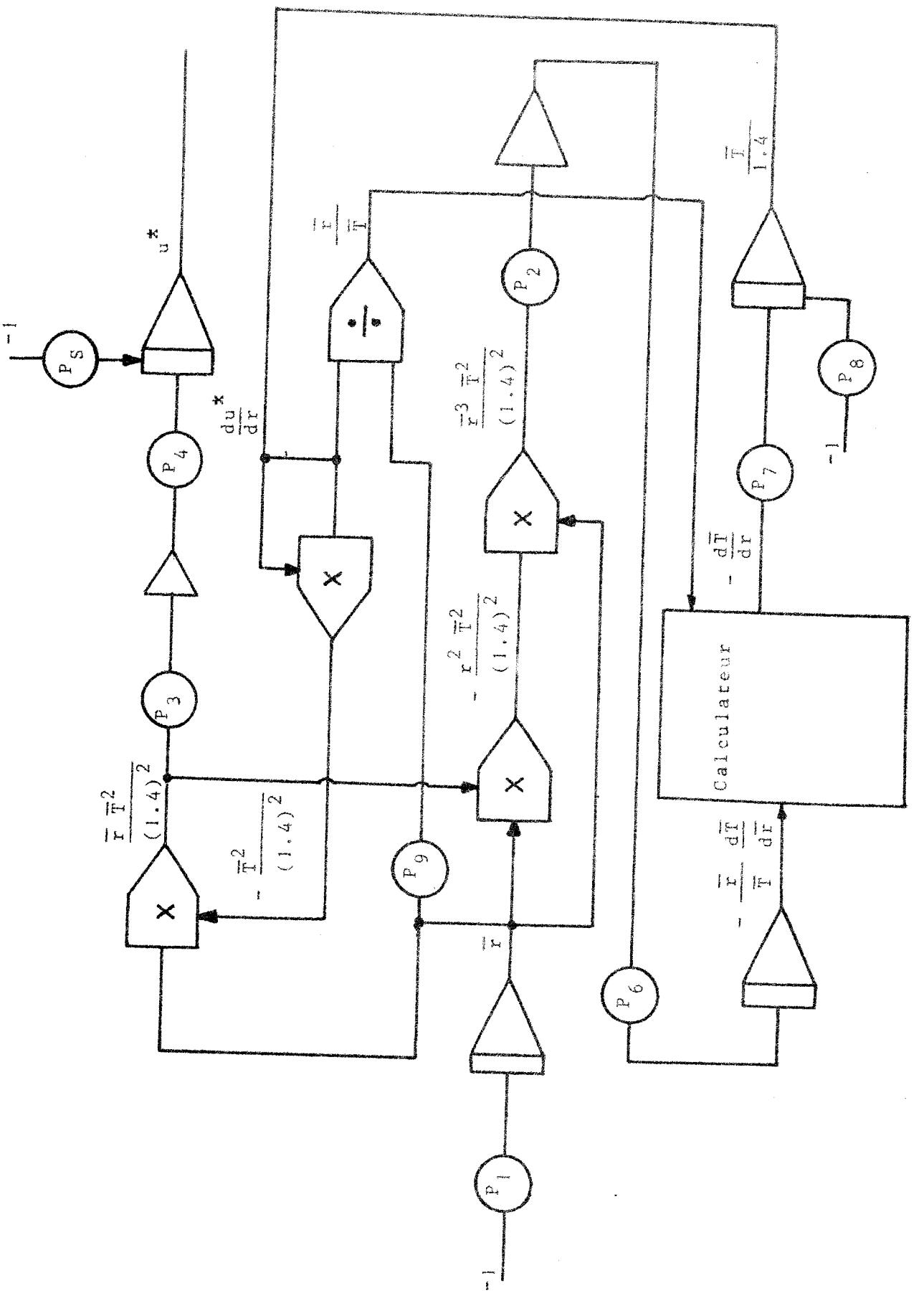


Figure 9
Simulation hybride

L'équation IX nous indique que $\overline{T}$ croit avec λ. L'observation de $\overline{T}$ fournira donc le critère de choix de (λ). Les résultats et l'organigramme du système sont donnés dans l'annexe (D).

4 - EXEMPLE 3

Cet exemple a été réalisé surtout pour montrer la précision du couplage et mettre en évidence une propriété de l'ensemble hybride : la génération de fonctions. L'étude choisie est simple, mais est la mieux adaptée pour discuter du problème de la précision.

4.1 - Problème

Notre intention est de générer une sinusoïde.

Nous disposons sur la partie analogique du signal Y généré et de sa dérivée Y'.

Les signaux sont de la forme : $Y = a \sin(\omega t + b)$

$Y' = a\omega \cos(\omega t + b)$

Nous allons réaliser le produit Y.Y' de deux manières :

- d'une manière analogique

- d'une manière numérique, en lisant par la chaine les valeurs Y et Y', en effectuant le produit numérique et en générant la valeur analogique correspondante.

Soit Z le signal donné par le multiplieur analogique et T le signal fourni par le convertisseur digital analogique du calculateur.

Nous observons chacun de ces signaux et la différence T - Z.

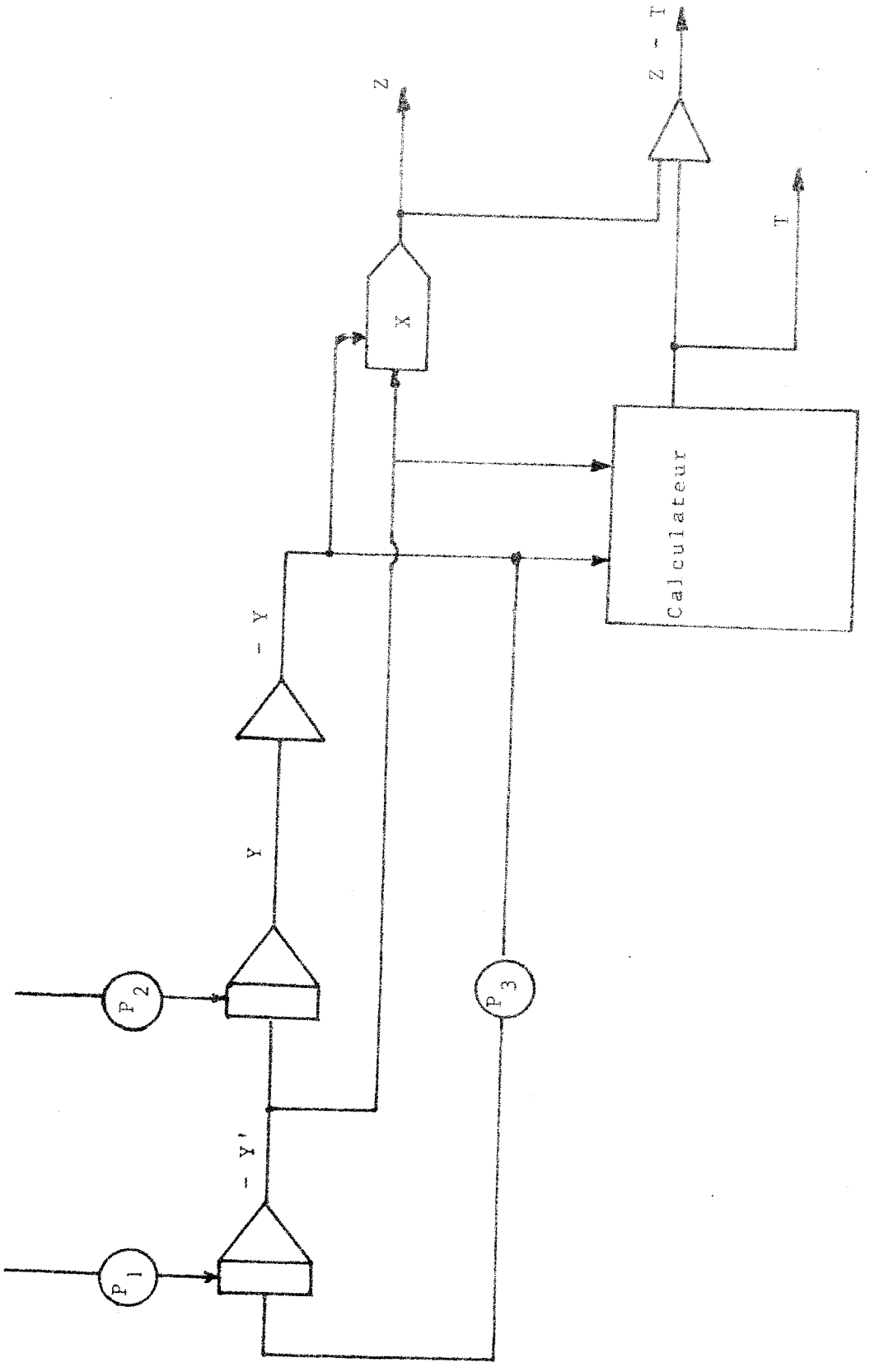


Figure 10

Simulation hybride -- Génération de fonctions

Le résultat est aussi une sinusoïde de période moitié

$$Y.Y' = a^2\omega \sin(\omega t + b) \cdot \cos(\omega t + b) = \frac{a^2\omega}{2} \sin 2(\omega t + b)$$

4.2 - Résultats

Les résultats sont donnés dans l'annexe (E).

Afin de pouvoir observer Z et T, nous sommes obligés de les considérer en opposition de phase sinon les sorties se correspondent.

Pour pouvoir observer l'erreur du système (T - Z), nous ajoutons un gain de 10 dans la chaine.

Si l'on considère l'erreur absolue, elle est de ± 1 m V, compte tenu du bruit qu'apportent les amplificateurs, la table traçante ....,

l'erreur relative est donc de $\frac{1}{2500} = 4\ 10^{-4}$, c'est à dire de l'ordre de la précision du système.

CONCLUSION GENERALE

La plupart des études, au niveau du laboratoire ou lors d'une application industrielle, font appel à une simulation d'un modèle. Les différents modes de simulation prennent donc une grande importance et c'est de l'évolution du matériel que dépend l'application de nouvelles méthodes de recherche permettant une analyse des problèmes plus efficace et une solution plus rapide. En nous préoccupant de satisfaire les nécessités spécifiquement hybride, nous nous orientons vers l'élaboration d'algorithmes sans perdre de vue toutefois la liaison étroite existant entre le hardware et le software.

L'accroissement des capacités des calculateurs numériques et les nouvelles possibilités de transmission des données impose un ensemble ne répondant plus à un concept à structure fixe, mais permettant le déroulement de plusieurs problèmes simultanément : on aura un partage du calculateur avec d'autres travaux travaillant en temps réel ou en mode séquentiel (Batch).

Les langages spéciaux (CSSL) doivent réduire les efforts de programmation et contenir l'accroissement des prix du software. Une solution consiste à introduire des langages orientés vers une classe particulière de problèmes, ou dans un autre sens, en reportant dans la mesure du possible les fonctions programmées sur des fonctions cablées, à utiliser des minicalculateurs microprogrammés spécialisés.

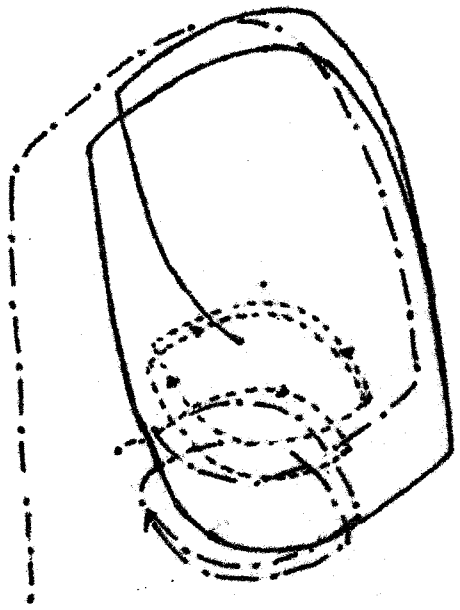
Nous avons décrit dans ce mémoire le système hybride tel que nous l'avons réalisé et non tel qu'il devrait être. Nous sommes conscients de ses limites et songeons, à partir de l'ébauche ainsi réalisée, à élaborer un système plus performant ; nous ne devons pas perdre de vue que cet appareil est utilisé dans le cadre d'un laboratoire, et en ce sens doit répondre à certains critères : il n'est donc pas nécessaire d'en accroître la complexité à seule fin d'obtenir un appareil performant. Certains compléments sont donc prévus au point de vue software pour l'établissement d'algorithmes, et au point de vue hardware pour améliorer la fiabilité de la machine et les contrôles des différentes phases au niveau de l'interface.

A la fin de cette dernière étape, nous aurons atteint le but que nous nous sommes assignés et défini lors de la première partie, nous pourrons prévoir alors dans un autre contexte une nouvelle direction de recherche à partir des techniques hybrides ainsi mises au point.

ANNEXE A

ENREGISTREMENTS GRAPHIQUES CORRESPONDANT

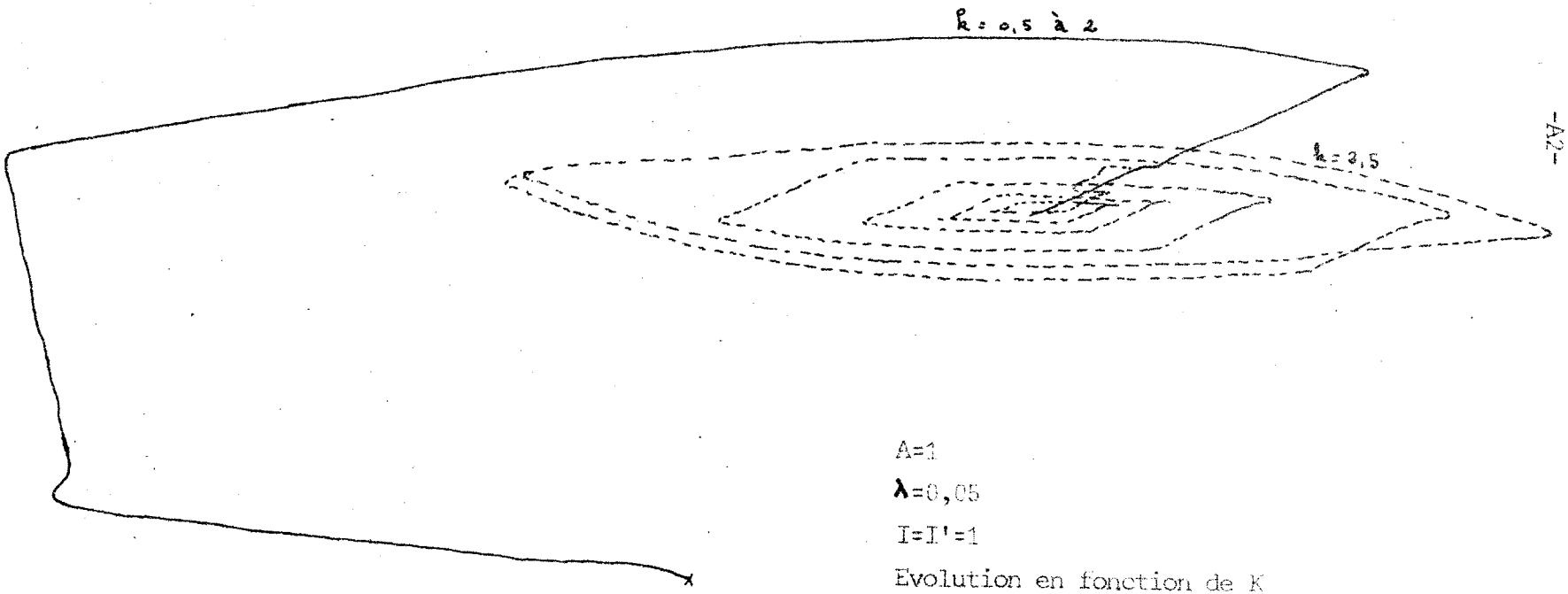
A L'EXEMPLE 1 : RECURRENCES et CYCLES LIMITES



Regime saturé:évolution dans le plan de phase y,y'

suivant les conditions initiales.

FIG. A1.



A=1

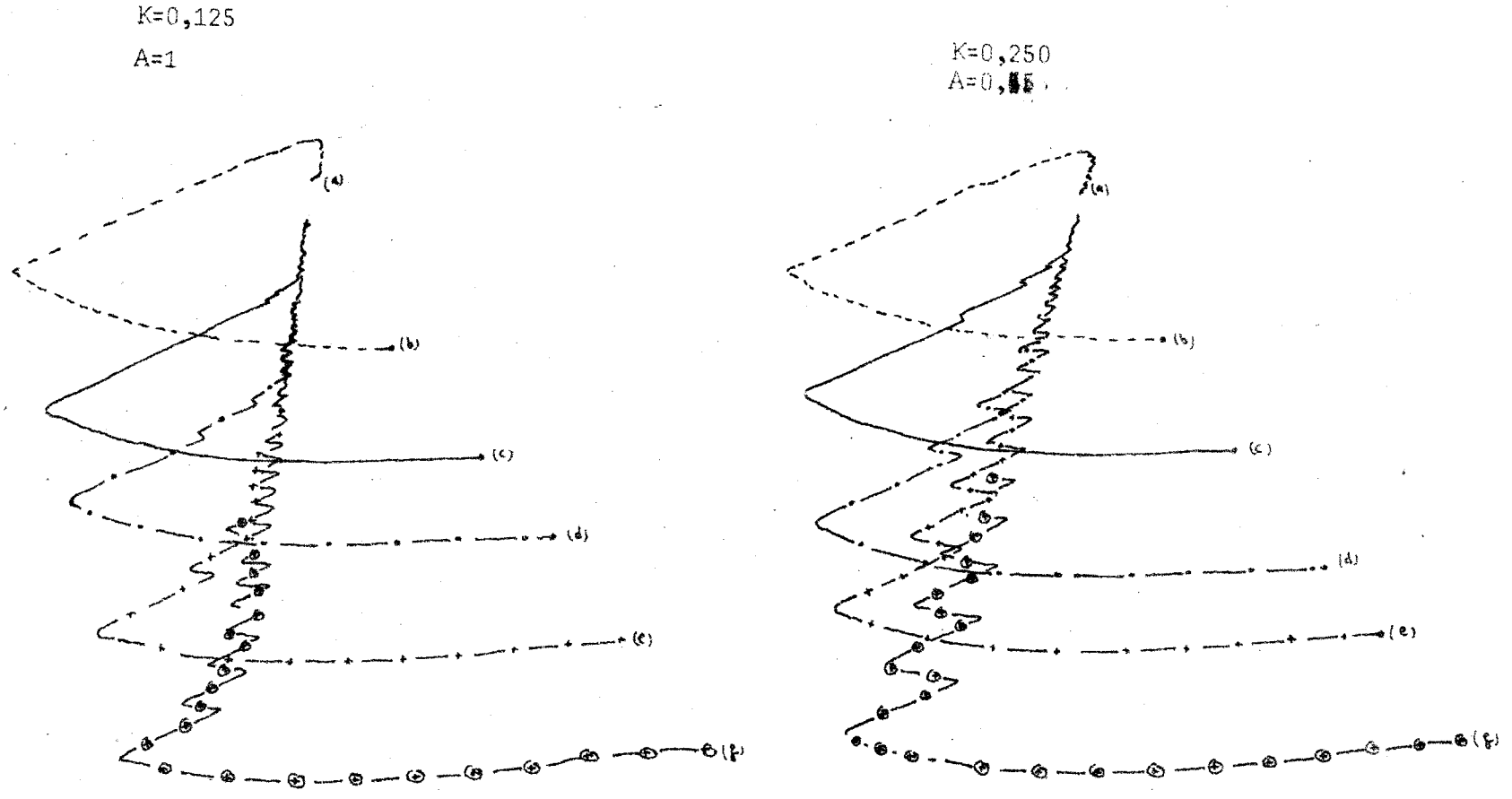
λ=0,05

I=I'=1

Evolution en fonction de K

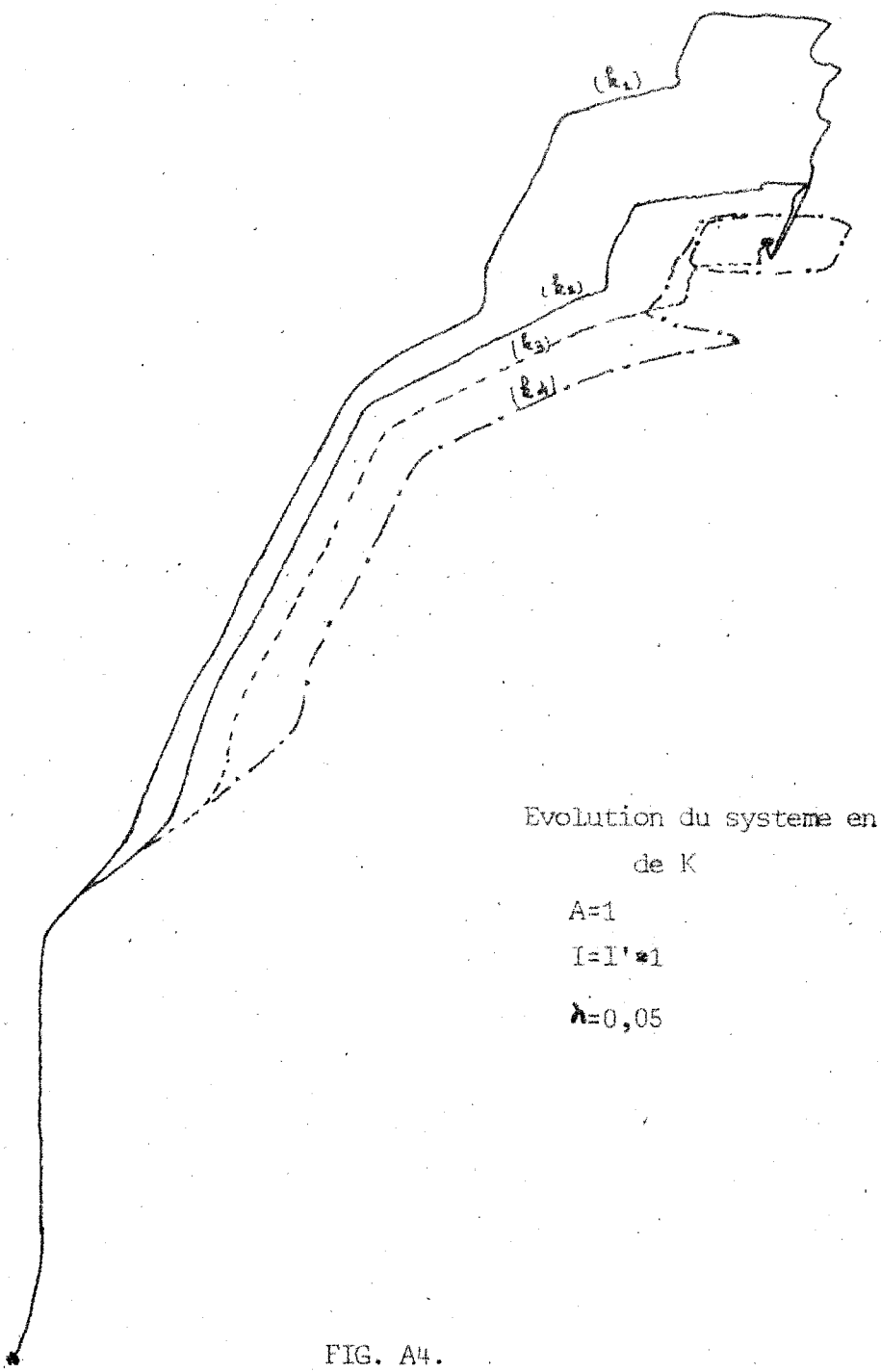
Cycle pour K=2,5

FIG. A2.



Comportement du systeme à K.A=constante.
pour differentes conditions initiales.

FIG. A3.



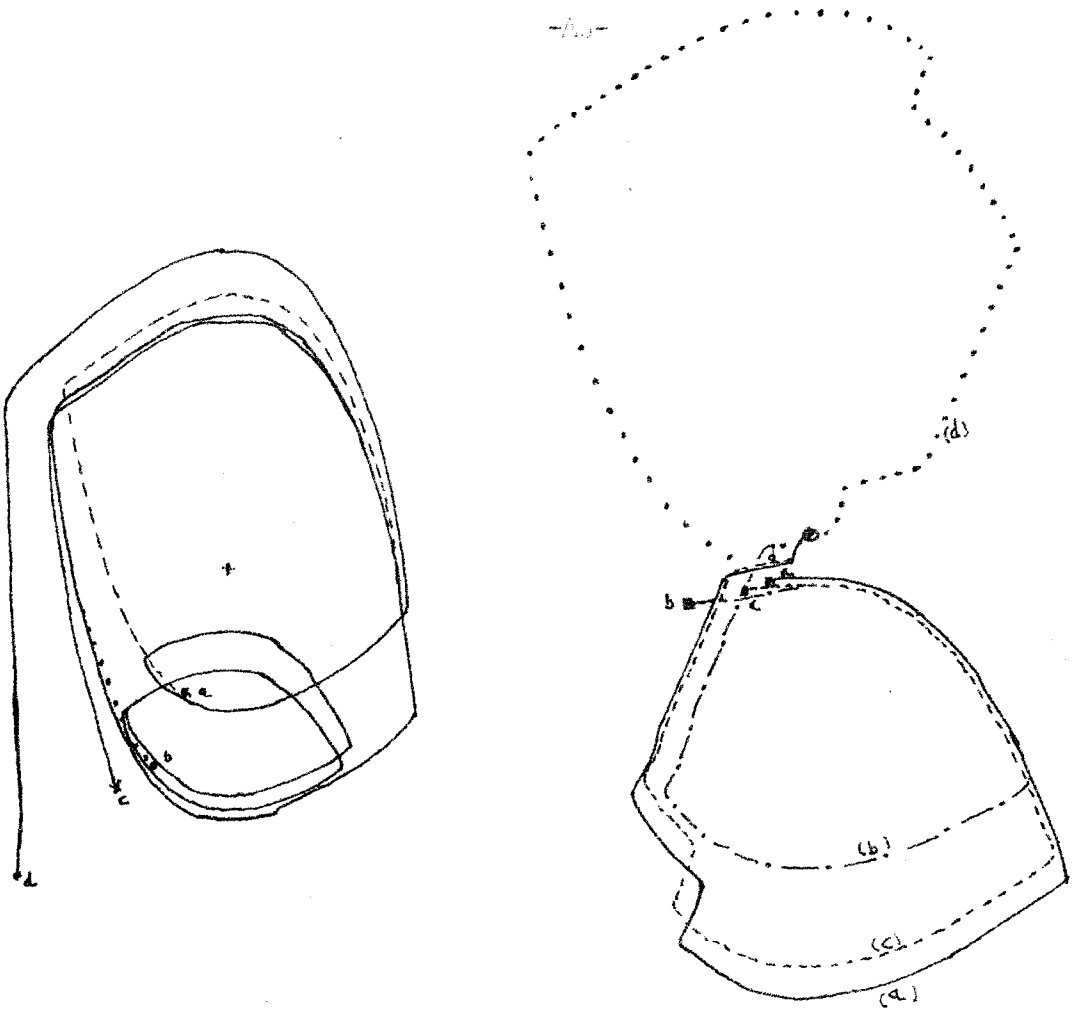
Evolution du systeme en fonction de K

A=1

I=I'=1

λ=0,05

FIG. A4.



A=1,97
K=0,8
λ=0,29
P=1

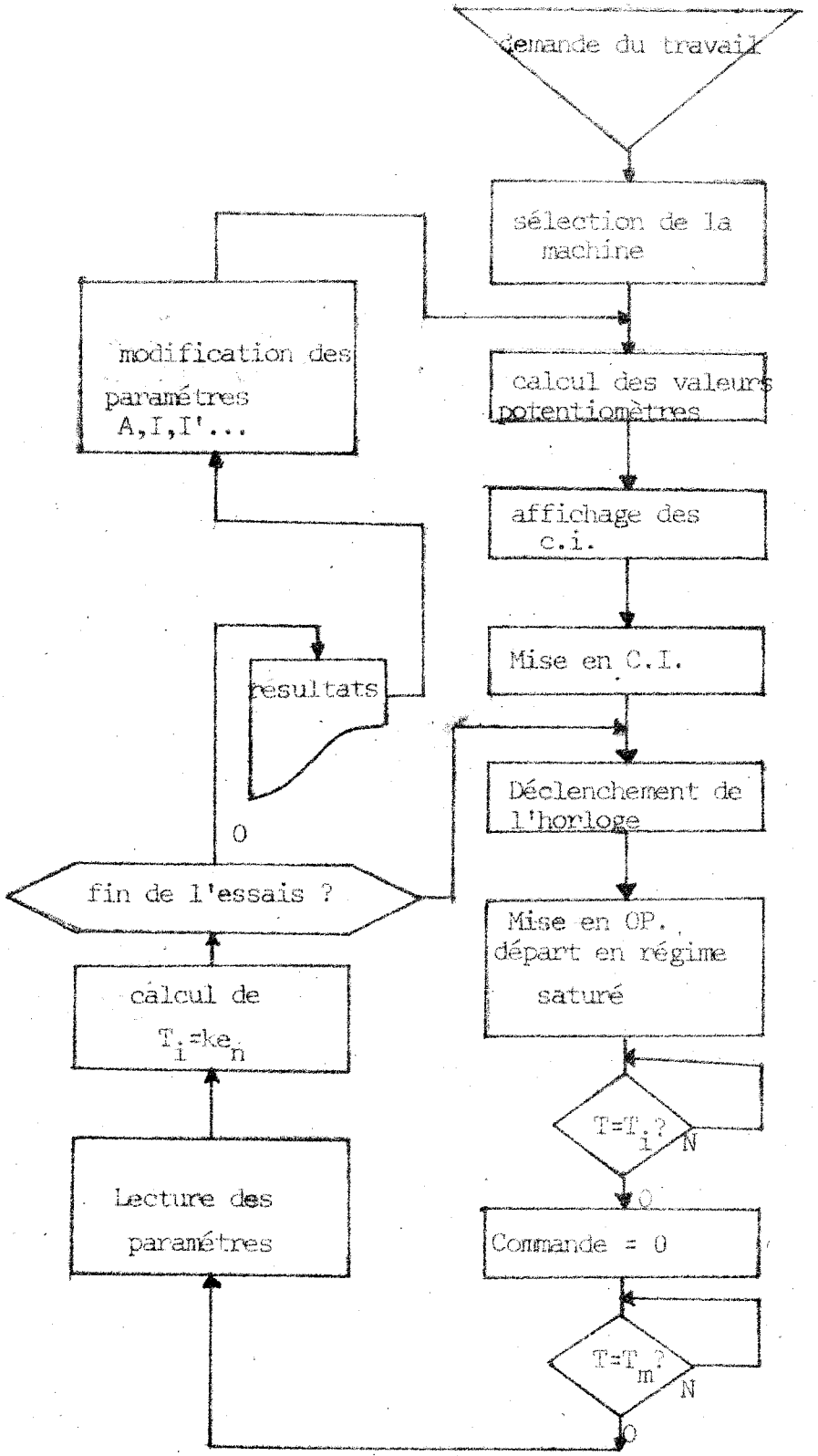
(A) (B)

Evolution en fonction des conditions initiales
plan y,y'

(A) régime staturé type de cycles.

(B) régime non saturé

FIG. A5.



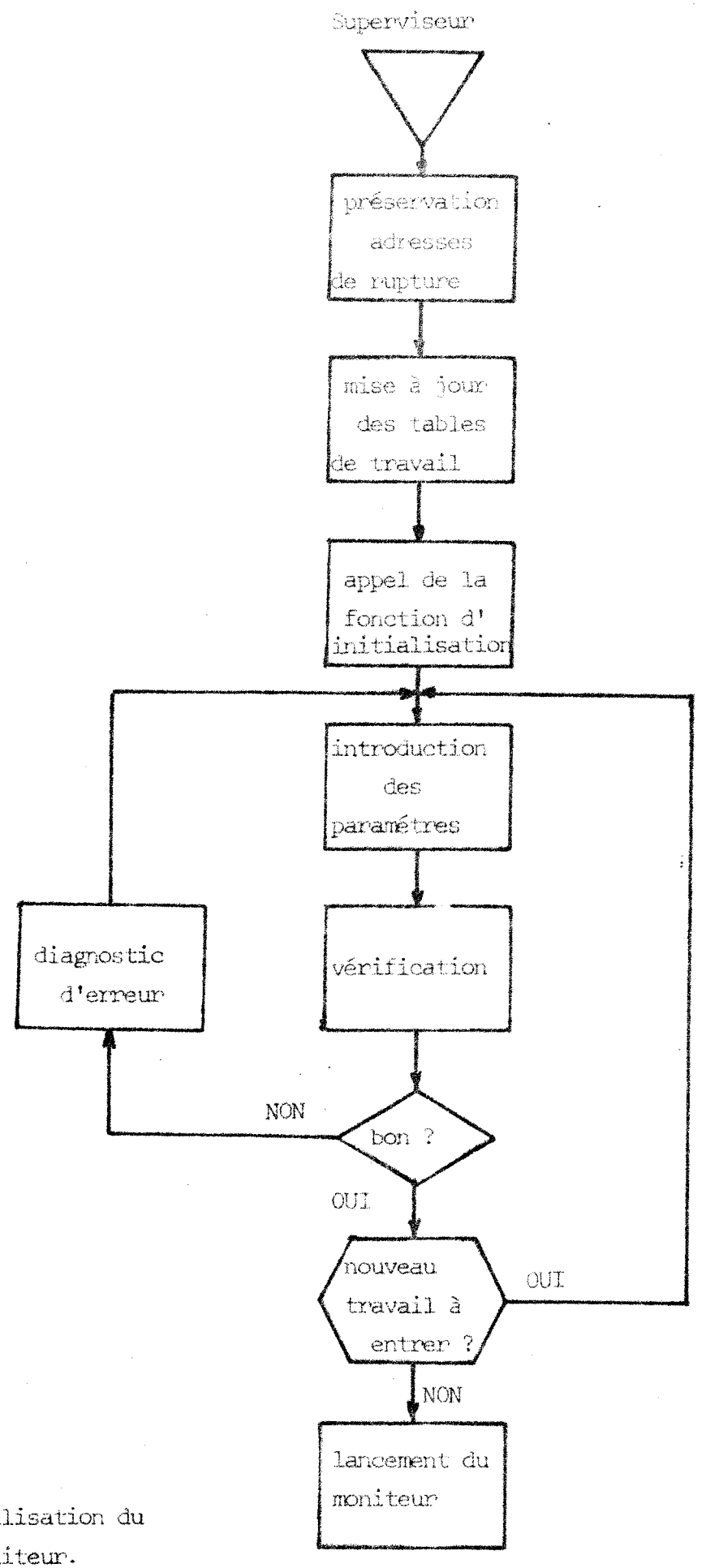
Organigramme de la simulation.

FIG. A6.

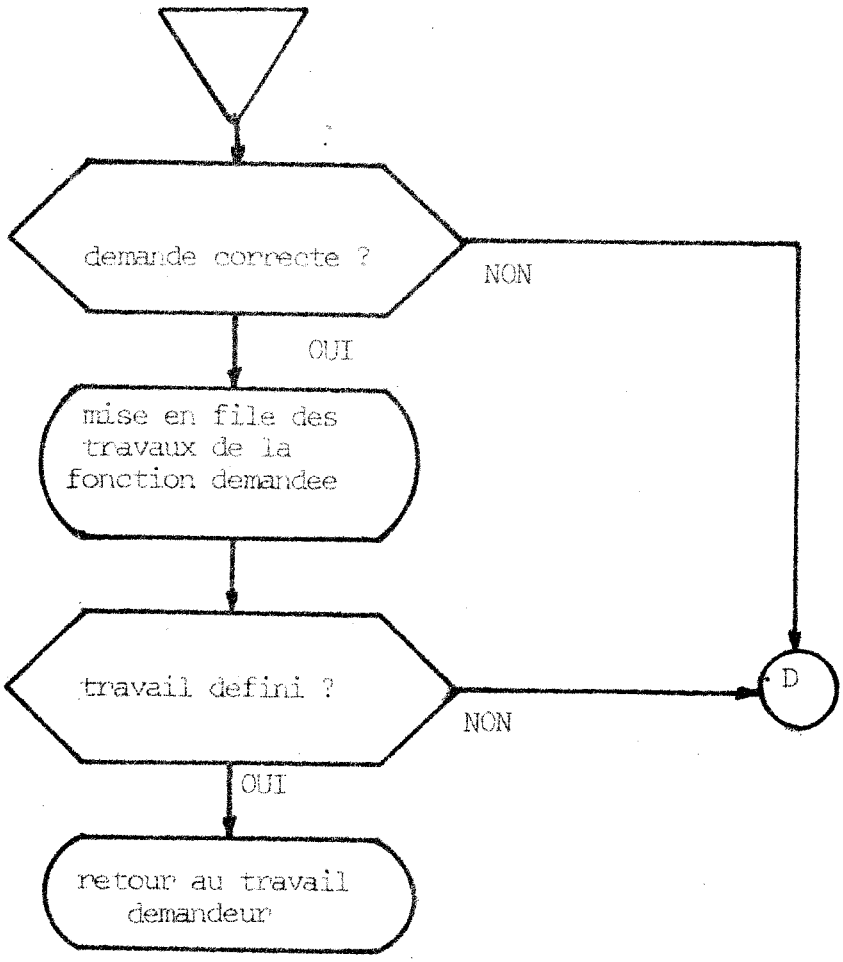
ANNEXE B

ORGANIGRAMMES CORRESPONDANT

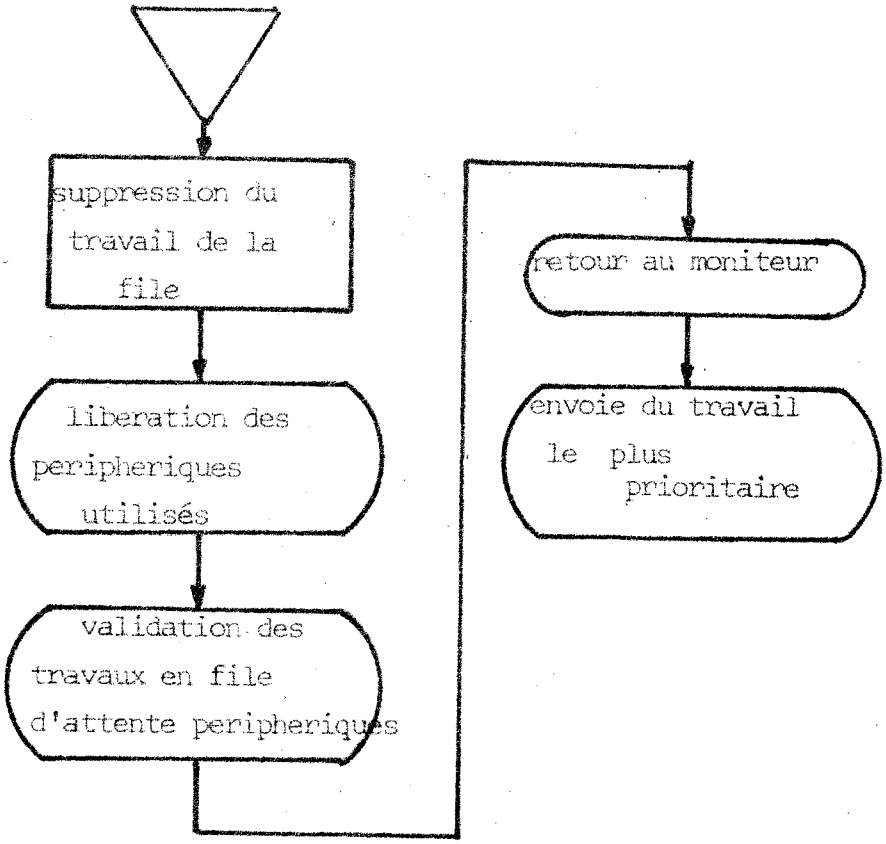
A QUELQUES FONCTIONS DU MONITEUR



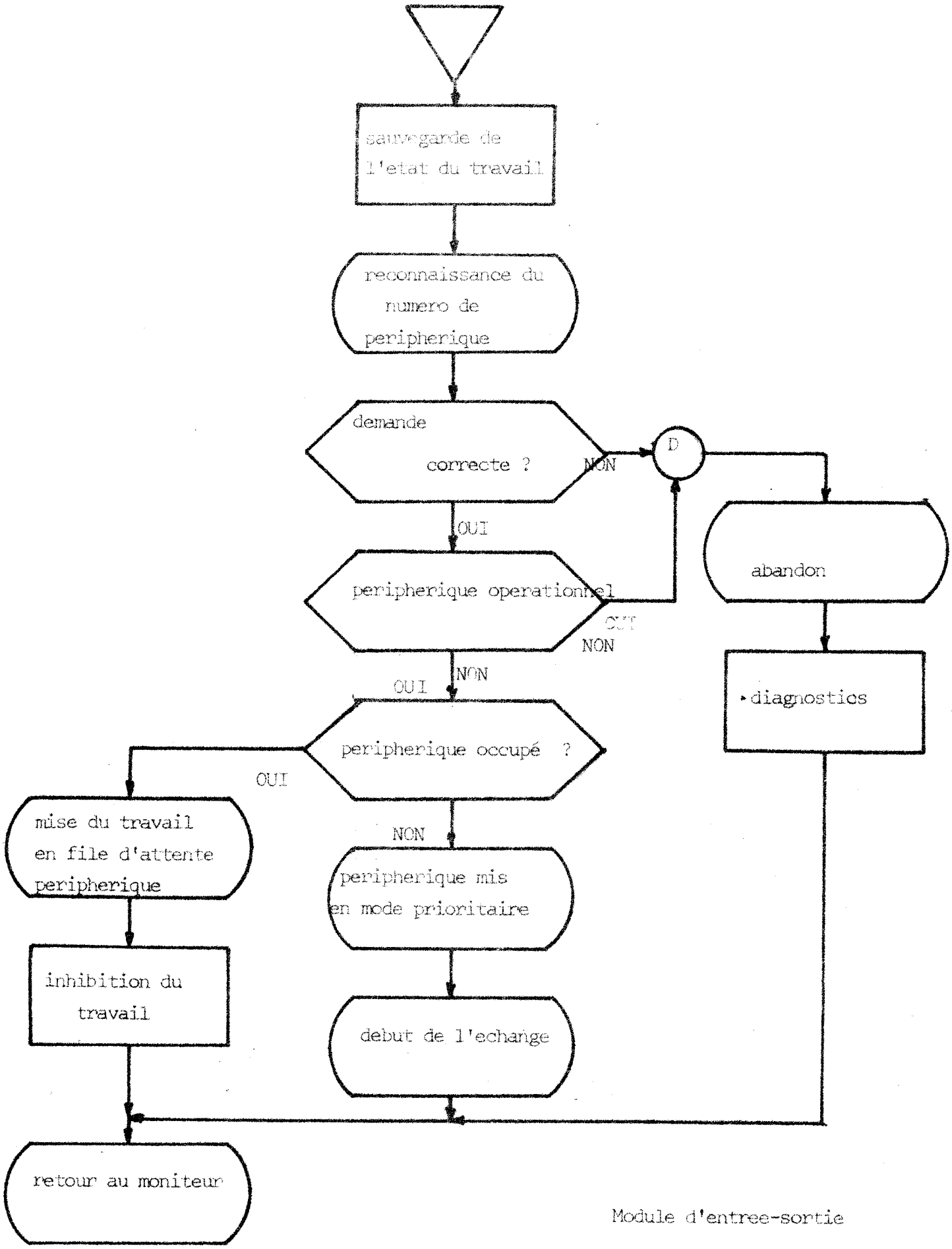
Initialisation du moniteur.



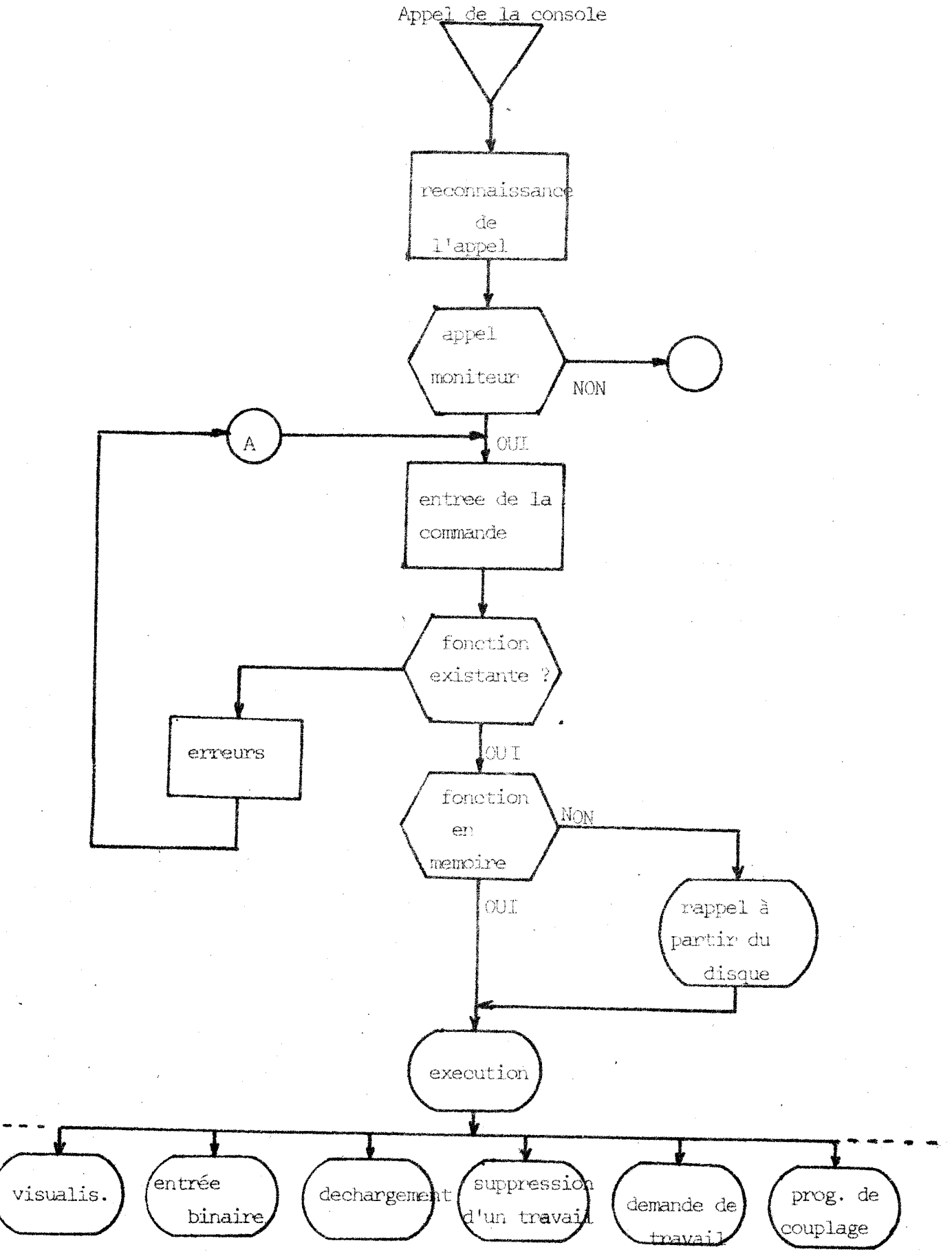
Module de demande des travaux



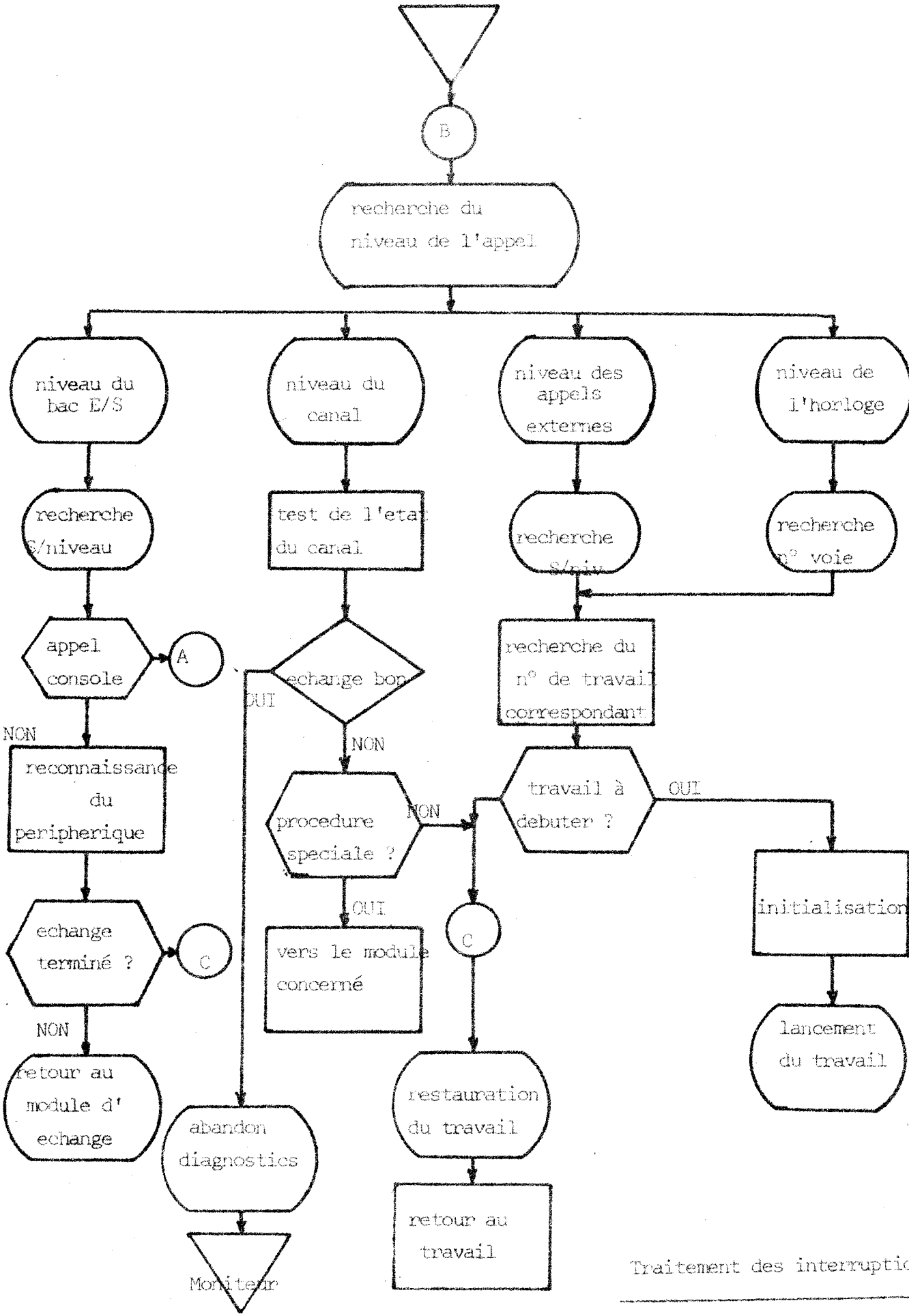
Module de fin de travail



Module d'entree-sortie



Gestion des fonctions du moniteur



Traitement des interruptions

ANNEXE C

UTILISATION DES FONCTIONS DU MONITEUR

SUR UN PROGRAMME

```
        SEC  ETUDE2;
        INIT ()          <MEMOIRES RELAIS >;
        HYBD (1,SP,CL,T1,M6) <INIT DU SYSTEME >;
        NIVAL (9,CADEN,2,TRAHOC H1* 50* );
        NIVAL (11,RDEF,9,) ;
DEBUT:  POT  (1,POTNUM,POTAB) <AFF DE P1 >;
        POT  (1,POTNUM+1,POTAB+1) <AFF DE P7 >;
        POT  (1,POTNUM+2,POTAB+2) <AFF DE P8 >;
        POT  (1,POTNUM+3,POTAB+2) <AFF DE P9 >;
MODEBU: CAV  +1;
        RA   GAMMA       <INIT DE GAMMA >;
AGAMA:  MPV  +40;
        GNE  +4;
        RA   VALEUR;
        POT  (1,POTNUM+4,VALEUR) <AFF DE P2 >;
        CA   GAMMA;
        IRV  RACVF       <SP DE RACINE CARREE >;
        IRV  PEUPAS;
        MPV  +40;
        GNE  +4;
        RA   VALEUR;
        POT  (1,POTNUM+5,VALEUR) <AFF DE P3 >;
        RZ   INFLAM;
        CA   BONINF;
        RA   NOVLAM;
        RA   SUPLAM      <PREP DU CRITERE SUR LAMBDA >;
ALAMDA: RA   LAMBDA      <VALEUR DE T(0) >;
        AD   MILLE;
        RA   TOTO;
        MP   TOTO;
        GNE  +18;
        RA   LAMKRE;
        CA   TREIZ8;
        DNE  +18;
        DV   LAMKRE;
        RA   VALEUR;
        POT  (1,POTNUM+6,VALEUR;
        CA   LAMKRE;
        MP   TOTO;
        GNE  +18;
        RA   TOTO;
        CA   QUAT9;
        DNE  +18;
        DV   TOTO;
        RA   VALEUR;
        POT  (1,POTNUM+7,VALEUR ) ;
ENFIN:  RZ   COMPT;
        RZ   VALEUR;
        CONVER (VALEUR) <CONVERSION N/A >;
        HYBD (1,CI,CL,0,0) <MISE EN CI DU SYSTEME >;
DEPART: HORLOG (S10,VOIE2) <DEPART DE LA PERIODE DE CALCUL>;
        HYBD (1,HOP,CL,0,0) <RESOLUTION >;
AVANT:  CX   AIGUI;
        IR X VAZY        <GENERE SI X=0,TRAVAU SI X=1 >;
GENERE: CHAINE (TABLOC) <ACQUISITION DE DONNEES >;
        CA   TABLEC+1 <VALEUR DE R >;
        MPV  +10;
        ADV  -1;
```

```
        NRV   AVANT;
        CA    TABLEC+4 <X1 >;
        DNE   +18;
        DV    TABLEC+3 <X2 >;
        RA    VALEUR;
        CONVER (VALEUR) ;
        IRV   AVANT;
TRAHOG: CAV   +1          <FIN DE TRAVAIL >;
        RA    AIGUI;
        FINTRA ()         <RETOUR AU PROGRAMME >;
TRAVAU: RZ    AIGUI;
        HYBD (1,SP,CL,0,0) <FIN DE PERIODE ET TESTS >;
        HORLOG (SO,VOIE2) <ARRET DU CADENCEUR >;
        CA    COMPT;
        PRV   APRES       <TRACES DE COURBES >;
        CA    TABLEC+2 <T/1,4 >;
        DNE   +4;
        MPV   +875;
        ST    BONINF;
        NRV   CPABON      <T INF A 1 - LAMBDA TROP PETIT >;
        ST    BONSUP;
        PRV   CPUBON      <T SUP A 1 - LAMBDA TROP GRAND >;
CERON:  CA    TABLEC+5 <CAL CUL DE U(0) >;
        MPV   +10;
        GNE   +7;
        AD    HUN;
        RA    HUZERO;
        POT   (1,POTNUM+8,HUZERO) <AFF DE P5 >;
        HYBD (1,CI,RU,0,0) ;
AVANT:  IC    COMPT       <PREP POUR LES COURBES >;
        CXV   -10;
        CAV   2070;
        SORT (9)          <SORTIE DE SONNERIES SUR TTY >;
        AXV   +1;
        IRV   AVANT+2;
        WAIT (ATTENTE DE LANCEMENT ) ;
        WAIT () ;
        IRV   DEPART;
APRES:  CA    COMPT;
        ADV   -2;
        PRV   AVANT       <CE N'EST PAS FINI >;
        CA    GAMMA;
        AD    ECART;
        RA    GAMMA;
        ADV   -25;
        NRV   AGAMMA      <GAMMA NON TERMINE >;
        FINTRA ()         <FIN DE TRAVAIL >;
CPUBON: CA    NOVLAM;
        RA    SUPLAM;
        CA    INFLAM;
        AD    SUPLAM;
        DNE   +1;
        RA    NOVLAM;
        IRV   ALAMDA;
CPABON: CA    NOVLAM;
        RA    INFLAM;
        IRV   CPUBON+2;
PEUPAS: CAV   "?";
        SORT (9)          <NOMBRE INF A 0 - RACINE IMPOSSIBLE>;
        FINTRA () ;
VAZY:   GENERE BI* BO;
        TRAV
```

```
VAZY:   GENERE BI* BO;
        TRAVAU BI* BO*;
GAMMA:  ;
AIGUI:  ;
        VALEUR;
INFLAM:;
        SUPLAM;
NOVLAM:;
LAMKRE:;
TOTO:   ;
HUZERO:;
COMPT:  ;
BONINF:+9900#;
BONSUP:+200#;
HUN:    +10000#;
TREIZ8:+1388#;
QUAT9:  +4900#;
ECART:  +4;
MILLE:  +1000#;
POTNUM:+10;
        +16;
        +17;
        +18;
        +11;
        +12;
        +13;
        +15;
        +14;
POTAR:  +1000#;
        +71#;
        +714#;
TABLOC:TABLEC BO*;
TABLEC:TAB  +8;
        FIN;
.
```

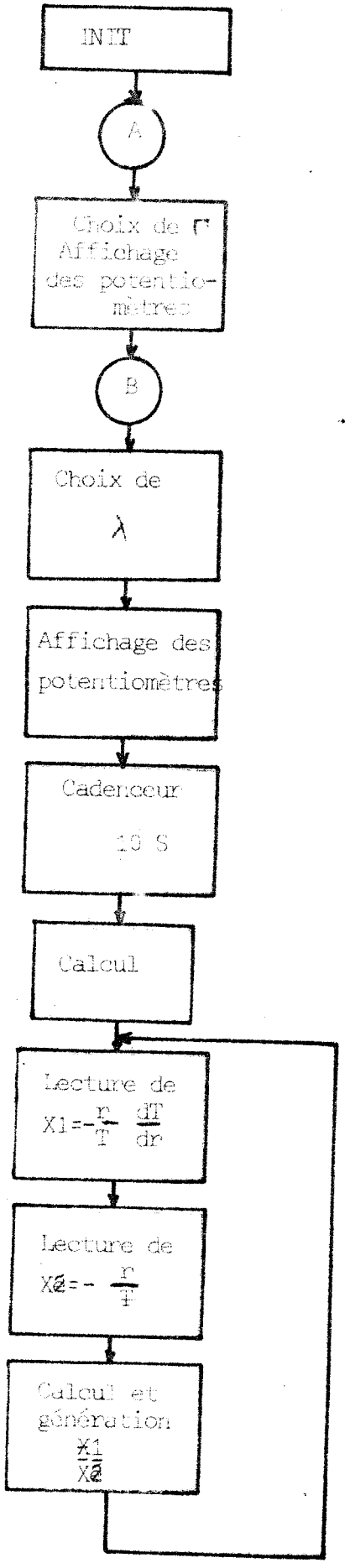
```
         COM              <SP DE RACINE CARREE >;
         SEC   SRACVF;
RACVF:   RT    PTRT;
         NRV   TILT;
         NE    NBNE;
         ZRV   TILT-1;
         RA    NN;
         CA    NBNE;
         ITV   +1;
         ZRV   ***+7;
         CA    NN;
         DLS   +1;
         RA    NN;
         MP    NP1;
         AD    NP2;
         IRV***+4;
         CA    NN;
         MP    NI1;
         AD    NI2;
         CXV   -3;
         IRV   ITRAC+7;
ITRAC:   CA    NN;
         DV    RACIN;
         DLS   +1;
         RA    MANVR;
         CA    RACIN;
         DLS   +1;
         AD    MANVR;
         RA    RACIN;
         MPV;
         AXV   +1;
         IRV   ITRAC;
         CAV   +18;
         AD    NBNE;
         DLS   +1;
         UNV   0400;
         RA    DECDNE;
         CA    RACIN;
DECDNE:  ;
         IC    PTRT;
TILT:    IR    PTRT;
PTRT:    ;
NBNE:    ;
NN:      ;
RACIN:   ;
MANVR:   ;
NP1:     3240 474;
NP2:     1140;
NI1:     2220;
NI2:     1560;
         FIN;
```

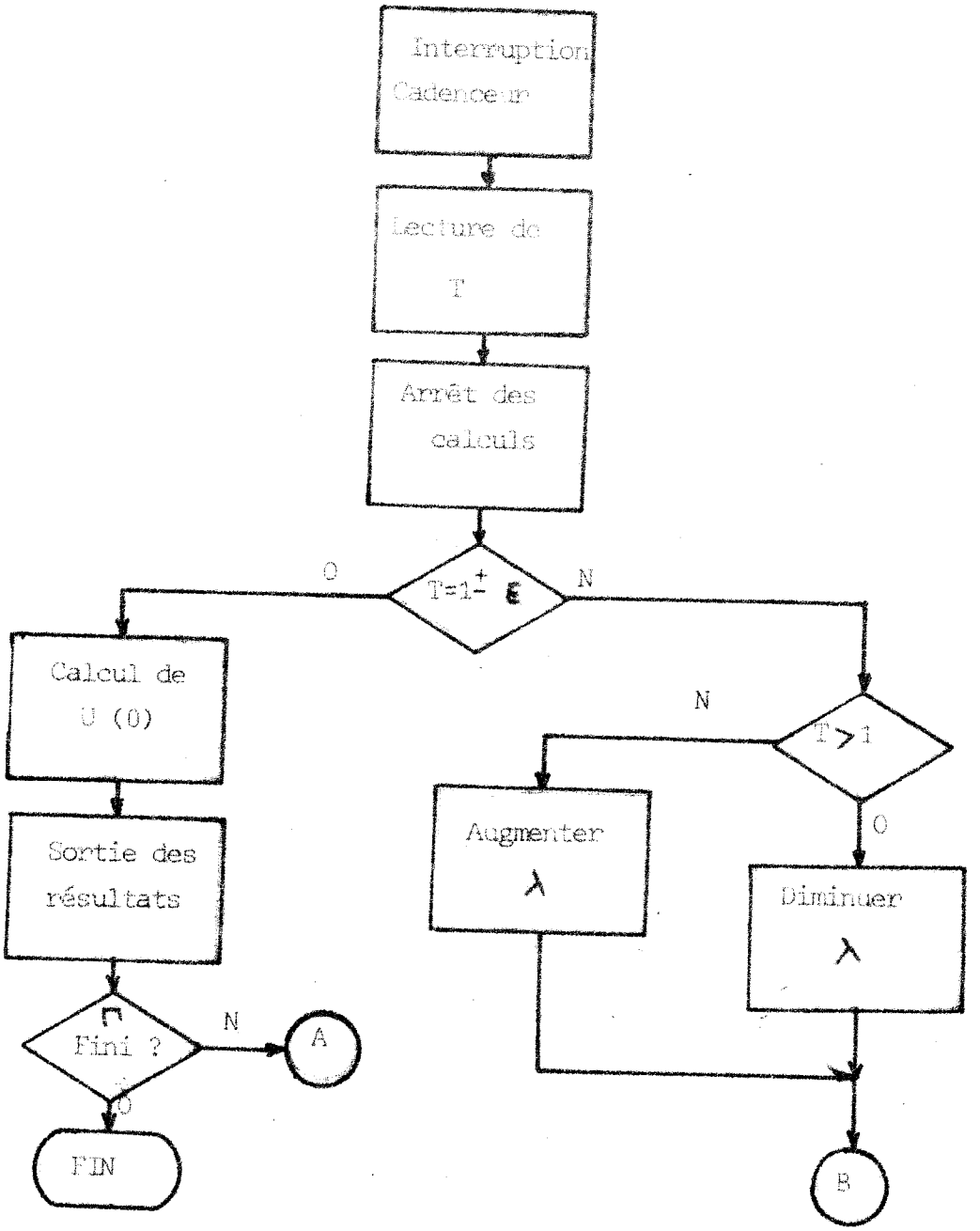
ANNEXE D

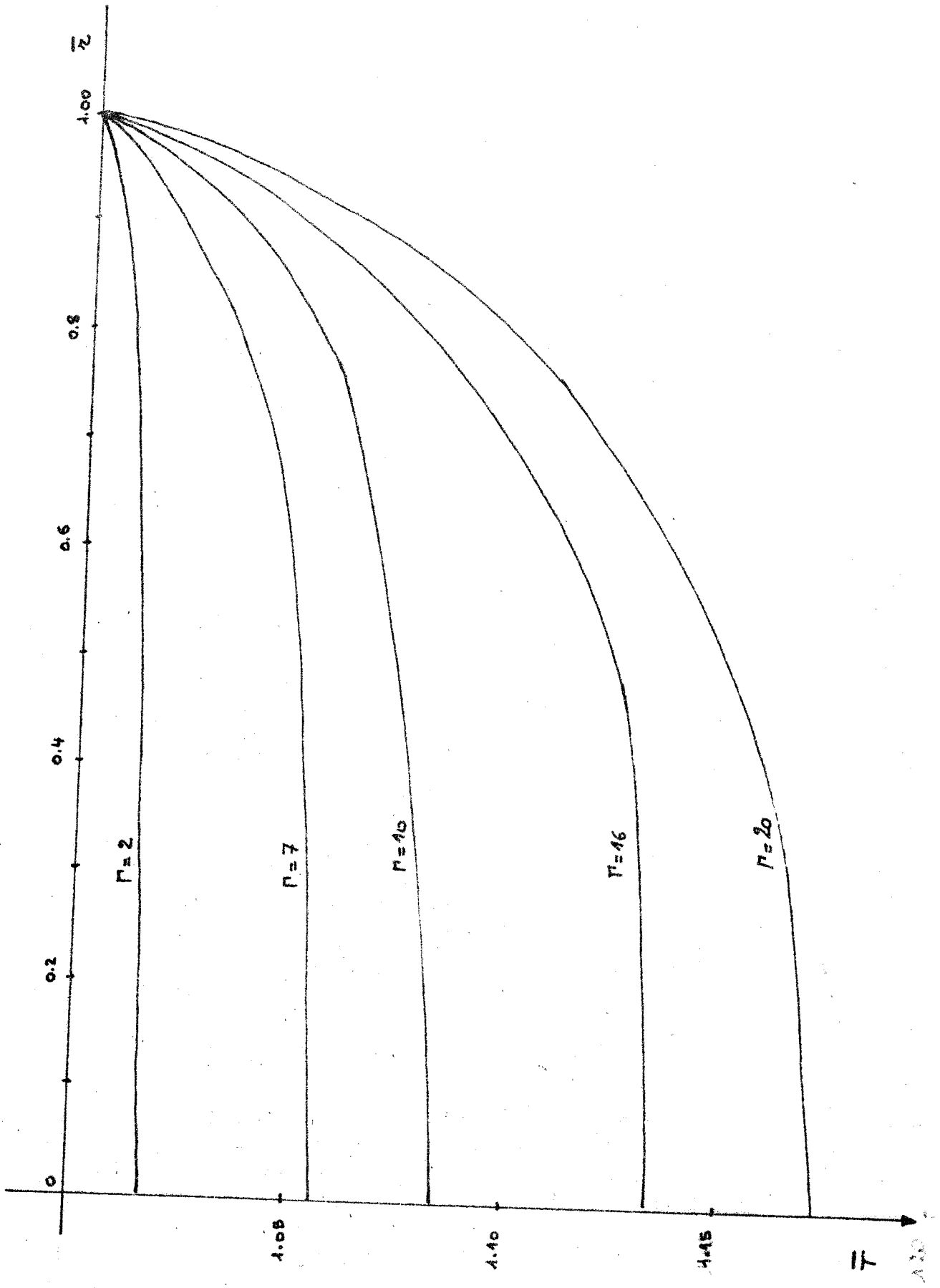
ORGANIGRAMMES ET RESULTATS

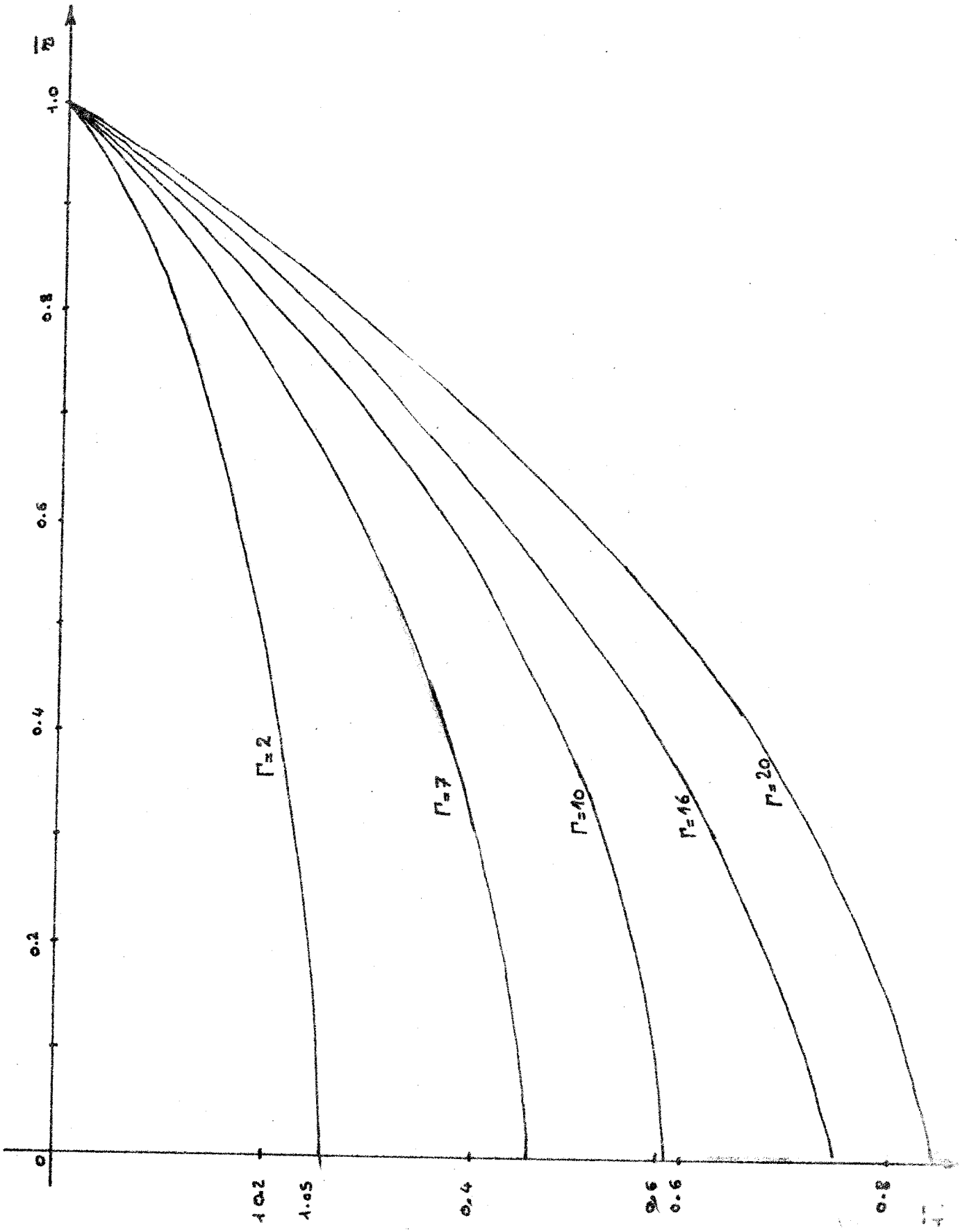
EXEMPLE 2 : ECOULEMENT D'UN FLUIDE

(Equation de POISEUILLE)


INIT
A
Choix de Γ
Affichage
des potentio-
mètres
B
Choix de
λ
Affichage des
potentiomètres
Cadencoeur
10 S
Calcul
Lecture de
X1=-r/T dT/dr
Lecture de
X2=- r/T
Calcul et
génération
X1
X2


Interruption Cadence-m
Lecture de T
Arrêt des calculs
T=1 ± ε
O
N
Calcul de U (0)
Sortie des résultats
Fini ?
N
O
A
FIN
N
T > 1
O
Augmenter λ
Diminuer λ
B

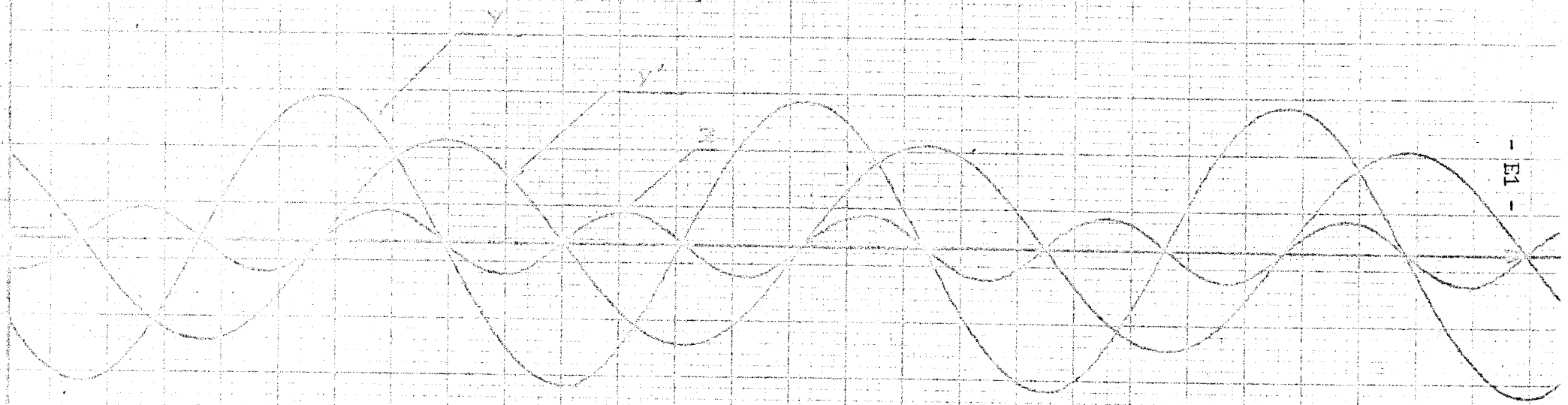

z̄
1.00
0.8
0.6
0.4
0.2
0
Γ=2
Γ=7
Γ=10
Γ=16
Γ=20
1.05
1.10
1.15
T̄


$\overline{\pi_0}$
1.0
0.8
0.6
0.4
0.2
0
Γ=2
Γ=7
Γ=10
Γ=16
Γ=20
1.02
1.05
0.4
0.6
0.6
0.8

ANNEXE E

VERIFICATION GRAPHIQUE ET EVALUATION DE LA PRECISION

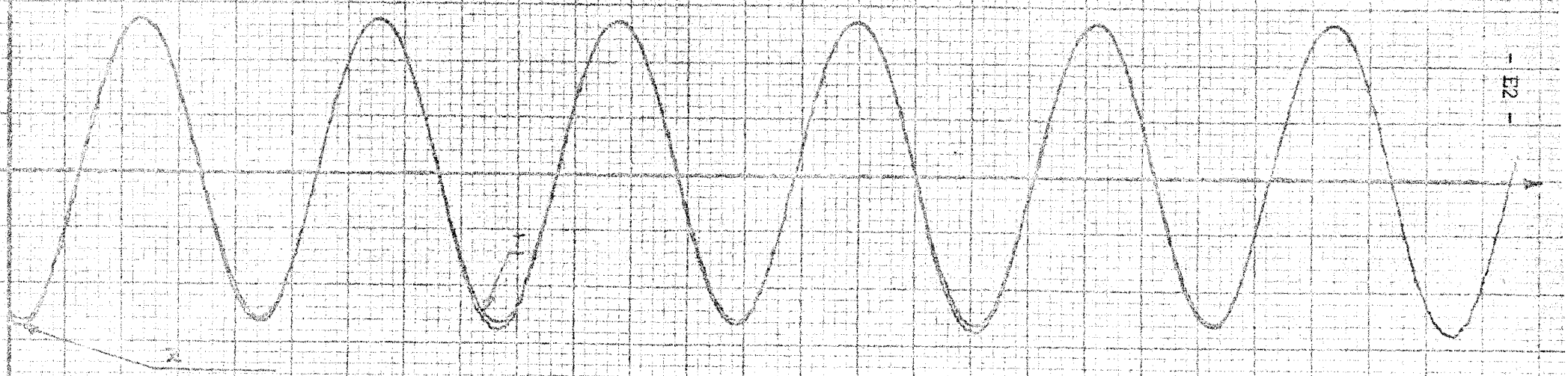
EXEMPLE 3



Echelle : 0.5 mV / Cm

Z = Y. Y'

Figure E-1 : Elaboration des sinusoïdes par une simulation analogique



Echelle : 0.3 mV/cm

$\bar{x}$ Signal venant du multiplieur analogique

T Signal venant du calculateur numérique

Figure E.2 : Signaux de sortie

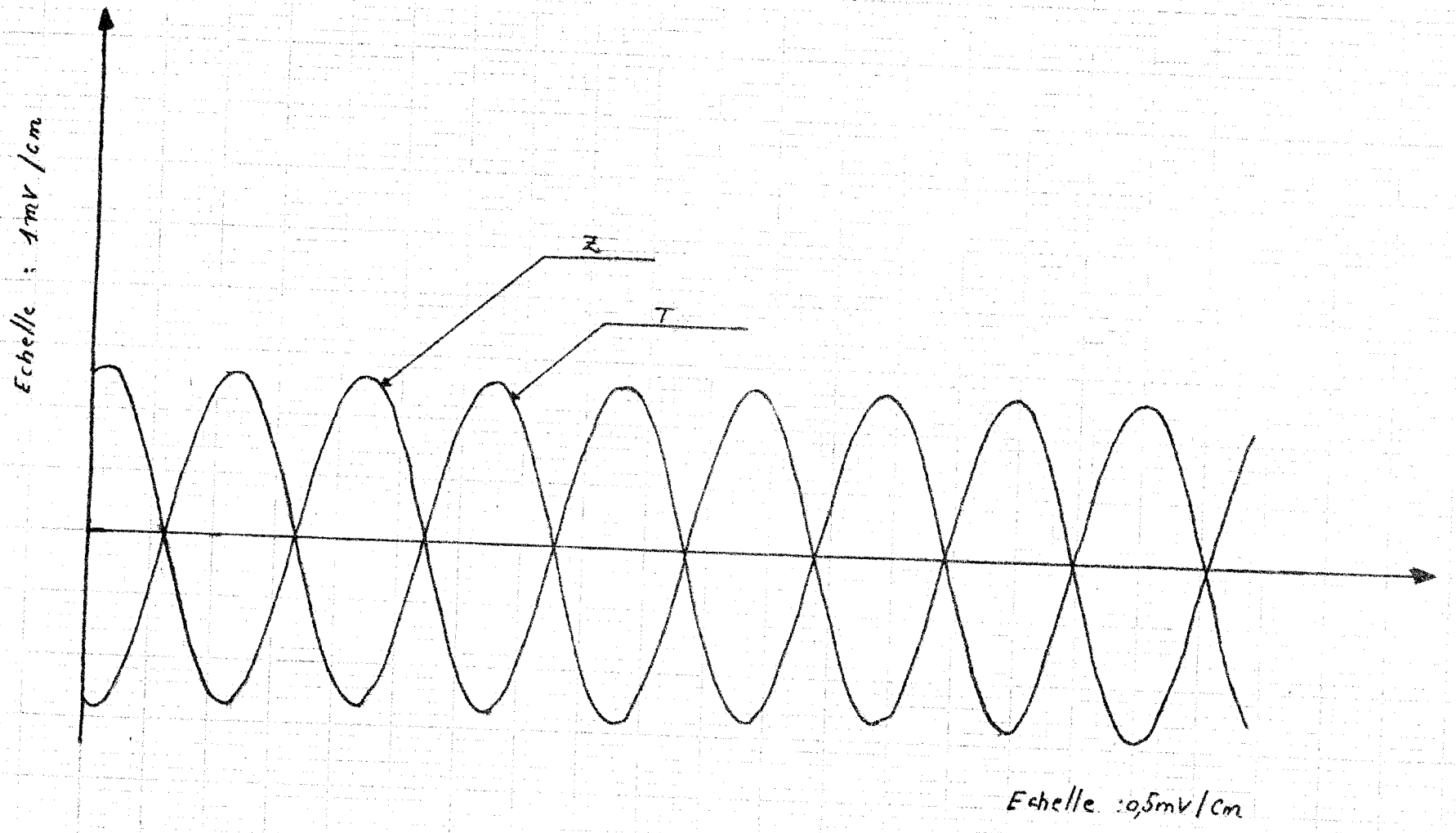


Figure E.3 : Sortie des signaux en opposition de phase.

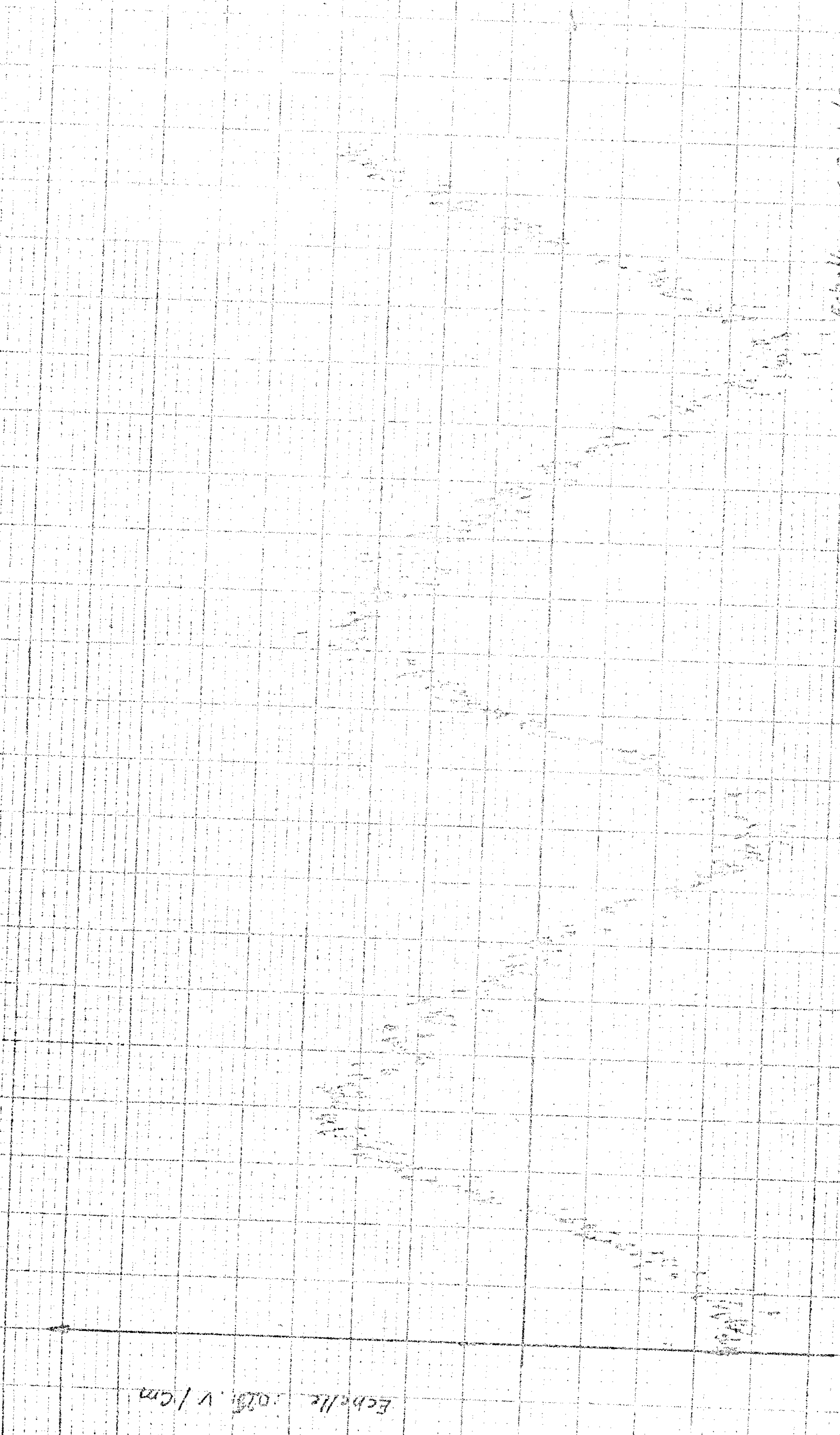


Figure E4 : Signal x 2 aux gain de do ao entrée

BIBLIOGRAPHIE

1 GRANINO A.KORN, THERESA M.KORN
"Electronic - Analog and Hybrid Computers".

2 J.P. NANTET
"Ordinateurs en temps réel - Applications industrielles".

3 SAUL STIMLER
"Real time data - processing systems."

4 S. FIFER
"Analog Computation ". .

5 "EAI - 590 - Hybrid Computing Systems"

6 "EAI 640 - Digital Computing Systems".

7 "Totality new hybrid computing Systems".
Applied Dynamics.

8 J. GAUDFERNAU
"Introduction au traitement de l'information en temps réel".

9 J. ARSAC
"Les systèmes de conduite des ordinateurs".

10 H.Y. CHANG, E.G. MANNING, G.METZE
"Fault diagnosis of digital systems".

11 W.J.KARPLUS, W.W. SOROKA
"Analog Methods".

12 W.S. KARPLUS
"Analog Simulation".

13 C. MELIN - J.M. TOULOTTE
"Les machines hybrides".

14 C.V. FEUVRIER
"La simulation des systèmes".

15 J.C. ANGUE
"Contribution à l'étude des systèmes discrets non linéaires décrits par un modèle recurrent".
Thèse de 3ème cycle - Juillet 1973

16 P. VIDAL
"Systèmes échantillonnés non linéaires".
Gorden and Breach - 1968

17 S.C. CHANG, L.G. BIRTON, A. KUMER
"Porseuille flow with variable physical properties - a simulation approach".
Septembre 1972.

PLAN DU MEMOIRE