Contribution à la réalisation d'un robot manipulant des objets en contact : commande par retour d'efforts

Patrick Baraona

N° d'ordre 2510

Année 1981

22265

THESE

présentée

DEVANT L'UNIVERSITE PAUL SABATIER DE TOULOUSE (Sciences)

en vue de l'obtention

du Titre de DOCTEUR 3e CYCLE

Spécialité : E.E.A. : Option : Automatique

par

Patrick BARAONA

Maître ès-Sciences

CONTRIBUTION A LA REALISATION D'UN ROBOT MANIPULANT DES OBJETS EN CONTACT. COMMANDE PAR RETOUR D'EFFORTS

Soutenue le 6 juillet 1981, devant la Commission d'Examen

MM	Y. SEVELY	*Président*
	Ph. COIFFET	*Examinateurs*
	G. GIRALT	
	A. GIRAUD	
	R. KELLEY	
	J.-L. LACOMBE	

AVANT-PROPOS

Le travail présenté dans ce mémoire a été réalisé au Laboratoire d'Automatique et d'Analyse des Systèmes du C.N.R.S. au sein de l'Equipe "Logique des Systèmes et Robotique".

Je remercie Monsieur le Professeur Y. SEVELY, d'avoir accepter la présidence de notre jury de thèse.

Je remercie :

- Monsieur Ph. COIFFET, Maître de Recherche au C.N.R.S., L.A.M., Montpellier,
- Monsieur G. GIRALT, Directeur de Recherche au C.N.R.S., et Responsable du Programme Robotique au L.A.A.S., Toulouse,
- Monsieur A. GIRAUD, Chargé de Recherche au C.N.R.S., Responsable de l'Opération "Commande et Architecture des Robots" et qui a assuré la direction scientifique de nos travaux,
- Monsieur R. KELLEY, Professeur à l'Université de RHODE-ISLAND, USA,
- Monsieur J.L. LACOMBE, Ingénieur à la Société MATRA, Vélizy,

pour leur participation au jury.

Je remercie également :

- tous les membres de l'Equipe L.S.R. et plus particulièrement Chazi, Michel(s), Abdel, Raja, Bernard, Danièle, François, Jean-Charles, Ralph et Marc(s),
- le personnel des Services S.I.S. et M.C.I. et plus particulièrement G. LARREGOLA,
- Martine LABERGUE pour la dactylographie,
- le personnel des Services Documentation et Imprimerie.

I N T R O D U C T I O N

Ces dernières années ont vu la naissance et le développement d'un nouveau domaine de recherche : la Robotique.
Dans ce contexte, définir ce que recouvre le terme de robotique paraît être une tâche difficile, tant les champs d'application de cette nouvelle science paraissent vastes.

Pourtant, nous pourrons dire que la Robotique vise à conférer à des machines un certain nombre de facultés qui leur permettent d'effectuer, de manière autonome, des tâches relativement complexes. De plus, ces machines, que nous appellerons "robot", doivent être dotées d'une capacité d'adaptation suffisamment grande pour interagir avec un environnement évolutif.

Un robot peut alors se définir comme un système automatique hautement adaptatif, intégrant outre les organes mécaniques et électroniques lui permettant de se déplacer et de manipuler des objets, les organes de perception indispensables pour évaluer l'état de son environnement. Cette intégration est faite à travers un centre de décision coordonnant les organes précédents.

Parmi les différents axes de travail développés en Robotique, les problèmes d'assemblage automatique d'objets prennent une part prépondérante. Nous nous sommes intéressés plus particulièrement à l'aspect "montage" de ce dernier problème.

Dans cette approche, le robot doit travailler dans une configuration où son organe terminal subi des contraintes d'efforts. Les capteurs opto-électroniques (caméra) utiles dans des phases de perception globale de l'environnement, ne nous permettent pas d'accéder à la maîtrise d'un tel problème. Nous utiliserons un type de perception plus locale, réalisé par l'intermédiaire d'un capteur d'effort tridimensionnel, situé sur le bras manipulateur. C'est une fonction équivalente au toucher et une partie des problèmes que nous nous sommes posés, relatifs au traitement et à l'interprétation des signaux issus du capteur (efforts) afin de maîtriser l'action du robot quand il travaille au contact.

Notre contribution porte donc sur la mise en place de primitives permettant la réalisation de manipulations canoniques de contact, par un robot expérimental muni d'un capteur d'effort tridimentionnel, et sur le développement d'une procédure d'asservissement en force et position, paramétrable, pour l'adapter aux caractéristiques de la liaison mécanique à obtenir.

Le premier chapitre, à travers une étude bibliographique non exhaustive, essaie de définir quels sont les choix qui s'imposent pour la réalisation d'un robot expérimental devant travailler en manipulant des objets sous des contraintes de forces (montage).

Le chapitre II décrit les primitives permettant de représenter l'espace de travail du robot.

Dans le chapitre III, nous abordons la maîtrise des efforts engendrés par les phénomènes de contact. Ceci passe par le développement d'asservissements en force et position et d'algorithmes d'interprétation des efforts.

Le chapitre IV décrit le support expérimental dont nous disposons pour mettre en oeuvre les résultats obtenus aux chapitres précédents.

Dans le chapitre V, nous exposons le logiciel que nous avons utilisé, tant au niveau supérieur, le langage de commande et ses primitives implantées sur le minicalculateur, qu'à un niveau plus bas, programmes fonctionnels gérant des tâches spécifiques et implantés sur des microprocesseurs.

Enfin dans le dernier chapitre, nous avons cherché à travers trois expérimentations à démontrer la versatilité de notre approche. La première vise à calibrer en position le capteur d'effort par rapport au robot et montre la possibilité d'utiliser un tel capteur pour simplifier les tâches de mise en oeuvre d'un robot d'assemblage. La seconde montre la possibilité de définir les dimensions et position d'un objet manipulé en aveugle par usage des efforts et d'accéder ainsi à une information géométrique. La troisième illustre que le robot peut faire évoluer un objet dans le cadre des degrés de liberté de la liaison mécanique établie.

CHAPITRE I

GÉNÉRALITÉS SUR LE PROBLÈME POSÉ PAR L'ASSEMBLAGE AUTOMATIQUE D'OBJETS

I.1. INTRODUCTION

Une grande part des travaux effectués dans l'industrie font appel à des assemblages de pièces élémentaires. Ces assemblages peuvent se résumer dans la plupart des cas par "des séquences ou pseudo séquences de cycles élémentaires de montage, entrecoupées éventuellement d'opérations d'inspection ou autres" [GIR$_b$ 81].

Notre objectif au cours de ce travail est d'apporter une contribution aux problèmes posés par le montage dans le cadre de l'assemblage automatique d'objets.

Tout d'abord, nous avons à faire face à un problème qui demande l'utilisation des techniques et des méthodes de l'automatique classique. Sa résolution doit permettre à la machine de se déplacer et d'agir sur son environnement en fonction de la tâche à accomplir, tout en tenant compte des perturbations pouvant survenir. Ce problème dépendra essentiellement des organes sensoriels dont sera doté le manipulateur (vision, laser, ultrasons, capteur de force, peau sensible). Leur rôle est de permettre au robot d'appréhender son état ainsi que celui de son univers d'action [GIR$_a$ 78].

Dans le cadre de l'assemblage automatique orienté vers le montage, le problème central est celui de la maîtrise des efforts et donc la conception d'un système de commande du manipulateur donnant la priorité à cette fonction. C'est ce que nous verrons dans la première partie.

D'autre part, la caractéristique principale des robots devant être l'adaptabilité, il est nécessaire d'avoir un niveau informatique supérieur qui contrôle la tâche à accomplir et qui est capable de choisir en tout ou partie le type de fonctionnement correspondant à une situation nouvelle détectée par les senseurs du robot. Nous rentrons alors dans le domaine de l'intelligence artificielle. L'opérateur commande la machine et communique avec le système par des ordres concis spécifiant les tâches à accomplir et ceci dans un langage de haut niveau.

I.2. LE PROBLEME DU MONTAGE = LA GESTION DES EFFORTS

Un des facteurs déterminant entrant dans la composition d'un robot est le choix des senseurs qui lui sont associés.

Nous pouvons distinguer deux grands types de capteurs intervenant dans la commande d'un robot :

- les capteurs de contact,
- les capteurs à distance.

Les capteurs de la deuxième catégorie sont essentiellement optiques, éventuellement ultrasonores. Une très nombreuse littérature s'y rapporte et les travaux relatifs aux organes de vision et aux traitements des informations obtenues (reconnaissance de forme) constituent un des axes de recherche en robotique [STU 80], [FER 81]. De plus ces capteurs sont essentiels pour un robot mobile [BAU 81] et dans les phases de transport de pièces dans le cas d'assemblage automatique.

Néanmoins, dans le problème de la saisie et du montage, il est fondamental de maîtriser les efforts dus à la mise en contact du robot avec les objets à manipuler afin de contrôler les petits déplacements de l'organe terminal et de pouvoir réaliser des opérations d'insertion.

Dans ce cas, un capteur non tactile n'est pas utile et un capteur d'effort ou de pression donnant une information précise sur les forces et les moments générés lors du contact s'impose comme étant la meilleure solution.

Nous sommes alors dans une configuration où nous disposons d'un robot aveugle qui doit, par le seul sens du toucher, se mouvoir dans un univers partiellement connu et effectuer des tâches nécessitant la mise en contact de pièces mécaniques entre elles.

Nous pouvons citer un certain nombre de travaux utilisant cette approche.

Des chercheurs japonais T. GOTO et K. TAKEYASU ont développé un robot à sept degrés de liberté, muni d'un mécanisme de saisie à mâchoires parallèles, de capteurs de contact (tout ou rien) et de pression (peau sensible) [GOT 72]. Le robot recherche la position d'un objet par toucher, puis place les doigts en fonction de la posture de l'objet à manipuler pour le saisir sans le déplacer. L'ampleur de la saisie permet de reconnaître la forme et l'orientation de cet objet qui est ensuite rangé (en étant poussé par le robot) dans une position définie au préalable. Ce travail démontre qu'il est possible de faire du conditionnement sans l'aide d'une fonction vision.

Les mêmes auteurs [GOT 74] ont poursuivi des recherches en vue de réaliser l'insertion de pièces mécaniques-piston dans un cylindre. Les modifications notables ont été l'adjonction d'un poignet flexible utilisant une "compliance"* [NEV 78] passive couplée à des capteurs tactiles permettant de palper la position ; un contrôleur séquentiel enchainant les tâches : contact, scrutation, insertion. TAKEYASU [TAK 76] a d'ailleurs recherché un algorithme de contrôle tactile adapté à la flexibilité (déplacement dans l plan X-Y suivant l modèle, avec contrôle de la force suivant l'axe vertical).

Dans le même ordre d'idée, Van BRUSSEL [BRU 77] expose un algorithme simple relatif à des choix de rigidité sur certains degrés de liberté du robot, permettant de résoudre des opérations d'assemblage. En particulier, une adpatation active utilisant l'information provenant de la mesure des forces, montre que l'utilisation d'une rigidité contrôlable sur chaque axe au niveau du poignet est une bonne solution de travail (programmation facile) sur le problème de la cheville à insérer dans le trou. Il a abordé également en se servant d'un modèle plan, les conditions du coincement et du blocage dans le cas de l'insertion du goujon dans un trou, cf. [CAL 75]. Cette modélisation a été reprise et élargie aux cas des insertions multiples par OHWOVORIOLE [OHW 78].

---

* compliance : anglicisme désignant la faculté d'une structure flottante élastique, à se déformer de manière à minimiser les efforts auxquels elle est soumise.

Nous retrouvons également au M.I.T. (Charles Stark draper laboratory) les mêmes choix sur la nécessité de connaître et de maîtriser les informations force tactile déplacement, générées au cours de la réalisation d'une tâche d'assemblage [NEV 73]. Dans le cas du montage, la commande de l'action doit être fondée sur l'information issue de la réalisation de la tâche [NEV 74].

En effet, une manipulation de montage utilisant le contrôle des forces et des déplacements nécessite la description de chaque élément et de chaque étape de cette manipulation d'où une modélisation de l'espace. Cette description peut être acquise par le robot en cours de manipulation par connaissance de la position et de l'orientation relatives des pièces entre elles. Les capteurs d'efforts mesurant six composantes, trois forces et trois moments dans un système de coordonnées orthonormées, semblent bien adaptés [WAT 75], [WHI 77].

Parallèlement, pour s'orienter vers l'insertion, la solution la meilleure paraît être celle des poignets "compliants" (déformable ou flexible, voir chapitre IV) actifs ou passifs [NEV 78]. SIMUNOVIC utilise un poignet passif tout en mesurant les forces de manière à commander les déplacements de l'organe terminal ; il obtient alors une compliance active [SIM 75] DRAKE [DRA 77] et GERELLE [GER 78] ont étudié aussi les problèmes théoriques de l'insertion pour réaliser le montage d'un alternateur à l'aide d'un poignet déformable, monté sur un bras commandé par calculateur. Nous sommes en présence d'une compliance passive, le bras réalisant les grands déplacements et l'assemblage complet.

Il apparaît donc à travers les travaux réalisés à l'étranger que l'association à un bras manipulateur d'un poignet flexible muni d'un capteur d'efforts mesurant les forces et les moments dans un repère cartésien est une bonne solution de travail. C'est ce support expérimental que nous avons adapté. Par contre, nous avons tenu à nous placer dans une optique où l'univers du robot est partiellement connu et donc visant à développer un ensemble de primitives permettant à partir des informations issues du contact de donner au robot certaines caractéristiques dimensionnelles sur les pièces saisies ou bien l'environnant. De même, nous avons voulu doté le manipulateur d'une commande associant la mesure des efforts aux déplacements incrémentaux et tenant compte des impératifs de sécurité (cf chapitre III).

Cette approche permet de réaliser une compliance active nécessaire à la réalisation d'un montage, en s'intégrant à la problématique de l'assemblage robotisé dans la mesure où elle esquisse des primitives du langage de commande.

I.3. LES LANGAGES DE COMMANDE POUR LES ROBOTS D'ASSEMBLAGE

Les problèmes rencontrés pour la mise au point d'expérimentations ou la modification, dans le cadre d'un milieu industriel, des tâches que doit accomplir le robot, constituent souvent un frein au développement de la robotique. Il est donc apparu essentiel de développer des langages de commande permettant de communiquer avec le robot et de le contrôler. Nous pouvons citer quelques uns des nombreux langages qui ont été développés ces dernières années.

- WAVE qui fut le premier système flexible construit pour le développement d'algorithmes permettant des manipulations complexes. Des travaux d'assemblage furent réalisés à l'aide de ce langage de commande [PAU 77].
- AL qui est un système de programmation pour la spécification des tâches à effectuer par un manipulateur. Il permet la construction, le test et la mise au point de programmes d'assemblage : il autorise l'intégration des algorithmes de montage (gestion des forces) et de vision [FIN 74] [FIN 75].
- Le système AUTOPASS qui modélise l'univers d'assemblage comme un graphe dont les entités de base sont des points, des droites et des surfaces [GRO 74] [PAR 77]. Ce langage détecte en particulier les erreurs de l'utilisateur à la compilation plutôt qu'à l'exécution [LIE 75].

- le langage LM réalisé par LATOMBE et MAZER dans le cadre du projet robotique Pandore [MAZ 81] [LAT 80] qui est un langage de programmation pour la description de tâches de manipulation et d'assemblage. Il doit permettre la prise de décision automatique, la modélisation des raisonnements, la perception par capteur (force, toucher, vision), la modélisation géométrique de l'univers des robots, la création et l'exploitation de base de connaissances et l'apprentissage.

Bien d'autres langages de commande ont vu le jour, en particulier aux Etats-Unis, et nous pouvons voir à ce sujet les travaux de William T. PARK qui en dresse une liste non exhaustive [PAR 77].

Le langage réalisé au LAAS par A. GIRAUD et en cours de développement, se présente dans son état actuel comme un interpréteur. Ecrit en langage Fortran IV, il facilite la description, la mise au point et la réalisation de tâches d'assemblage, confiées à un robot.

Il se rapproche du RPL (Robot Programming Language) [NIT 79] bien qu'actuellement il ne contienne pas de phase de compilation, mais se présente sous la forme d'un code objet interprétable (cf chapitre V).

I.4. CONCLUSION

L'étude des travaux menés sur le problème de l'assemblage automatique d'objets, plus spécialement centré sur le montage, nous a conduit aux constatations suivantes :

- d'une part la nécessité d'un support matériel incluant :
  - .un bras manipulateur à X degrés de liberté,
  - .un poignet flexible passif ou actif,
  - .un capteur d'efforts fournissant l'information sur les contraintes appliquées à l'organe terminal,
  - .une pince ;
- d'autre part, la réalisation et la conception d'un logiciel de commande et de contrôle du robot comprenant :
  - .la représentation de l'univers de travail,
  - .l'asservissement en force et en position de l'organe terminal du robot,
  - .un langage de commande permettant la communication et la reprogrammation du robot.

C'est ce type de structure que nous avons adoptée dans notre approche et dont nous décrivons la spécificité dans les chapitres suivants.

C H A P I T R E II

MAITRISE DE L'ESPACE

II.1. INTRODUCTION

Nous avons examiné dans le chapitre précédent les problèmes généraux posés par la réalisation d'un robot capable de faire du montage.

Dans celui-ci, nous définirons la représentation de l'univers que nous avons adoptée, et les primitives qui en découlent.

En effet, la description de différents corps et la possibilité de situer les positions relatives des objets entre eux, qu'ils soient fixes ou en mouvement, nécessitent un repérage de chacun des corps. Dans notre approche, le manipulateur sera considéré comme un sous-ensemble de l'univers dans lequel il évolue. Nous le caractériserons par une chaîne de repères ; ces repères représentent, non pas ses articulations, mais les organes fonctionnels qui le composent (capteur, bras, pièce...).

C'est la définition et la gestion des repères qui constituera la première partie.

La deuxième partie fera le lien entre l'asservissement décrit au chapitre III et l'utilisation que nous pouvons en faire suivant la manière dont sont définis les repères associés aux différents corps.

La troisième partie décrira les primitives graphiques développées pour visualiser les diverses étapes de nos expérimentations : repères, forces, déplacements, objets.

II.2. MODELISATION DE L'UNIVERS DU MANIPULATEUR

La description de l'espace englobant le robot et son environnement doit nous permettre :

- de localiser des objets les uns par rapport aux autres,
- de traduire le déplacement d'un corps,
- d'exprimer un quelconque élément constitutif d'un solide par rapport à ses autres éléments ou à d'autres solides,
- d'avoir une référence où tous les ensembles représentant l'univers puissent être exprimés.

Nous définirons donc des corps entre eux en exprimant par le biais de repères qui leur seront liés, leur position et leur orientation dans l'espace à trois dimensions.

II.2.1. Représentation des corps par des repères orthonormés

II.2.1.1. Définition de la position d'un corps

Nous cherchons à définir la position d'un point 0_j par rapport à une référence. Nous pouvons utiliser les coordonnées cartésiennes, cylindriques ou sphériques ; notre choix s'est porté sur les coordonnées cartésiennes.

Le point 0_j est alors défini par le rayon vecteur r égal à 0_i0_j. (Nous le noterons 0_{ij}, Figure II.1). Les coordonnées du point 0_j dans le repère orthonormé R_i $(0_i, X_i, Y_i, Z_i)$ seront définies par $O_{ij} = (r, r_2, r_3)^T$

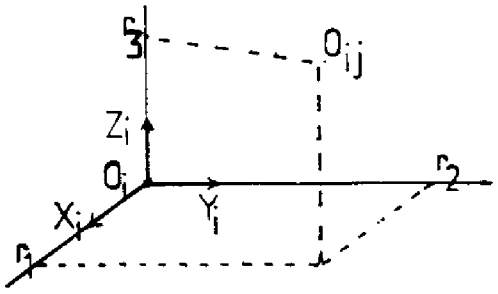


FIGURE II.1 *Définition de la position d'un corps*

II.2.1.2. Définition de l'orientation d'un corps

Nous avons défini la position d'un corps en exprimant les coordonnées d'un point de ce corps dans un repère orthonormé. De même, son orientation sera décrite en attachant à ce point un repère orthonormé de vecteurs unitaires notés X_j, Y_j, Z_j (Figure II.2).

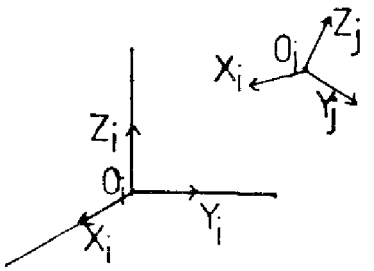


FIGURE II.2 *Définition de l'orientation d'un corps*

Plusieurs méthodes s'offrent à nous pour définir l'orientation du repère R_j par rapport au repère R_i :

- La représentation par les angles d'Euler classiques LAN 69 qui nécessite le choix de trois angles (précession, mutation et rotation propre) ou n'importe quel système d'angle d'Euler mixte. Dans tous ces cas, une direction est privilégiée par rapport aux autres et la symétrie est détruite ;
- Une manière simple de définir l'orientation du repère R_j par rapport à R_i est de donner les cosinus directeurs des vecteurs X_j, Y_j, Z_j dans le repère R_i. C'est cette représentation de base que nous utiliserons ;
- Une troisième solution est d'utiliser les paramètres d'Euler [ABA 63] [REN 80].

Nous définissons ces deux représentations ci-après, car elles nous seront utiles pour la définition des repères.

a) Cosinus directeurs

Les cosinus directeurs $r_{ij}(i,j=1,2,3)$ sont les composantes dans le repère R_i des vecteurs de base X_j, Y_j, Z_j du repère R_i

$$X_{ij} = (r_{11} \quad r_{12} \quad r_{13})^T$$

$$Y_{ij} = (r_{21} \quad r_{22} \quad r_{23})^T$$

$$Z_{ij} = (r_{31} \quad r_{32} \quad r_{33})^T$$

Nous pouvons définir, une matrice 3x3, notée S_{ij} qui sera la matrice de passage du repère R_j au repère R_i. Cette matrice rendra compte de l'orientation des axes de R_j dans le repère R_i

$$S_{ij} = \begin{bmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{bmatrix}$$

b) Paramètres d'Euler

Nous pouvons obtenir le repère R_j en faisant pivoter le repère R_i d'un angle ε autour d'une direction de vecteur unitaire e (si on ne tient pas compte de la translation).

L'angle ε est supposé toujours positif et le sens de rotation est défini par e (règle du tire-bouchon) (Figure II.3).

FIGURE II.3 orientation de R_j/R_i définie par (e,ε)

Les composantes de e dans R_i sont

$$e/R_i = (e_1 \quad e_2 \quad e_3)^T$$

Nous pouvons poser : $E = \sin \frac{\varepsilon}{2} \cdot e$

avec $E = (p \quad q \quad r)^T$

$$p = e_1 \sin \frac{\varepsilon}{2} \qquad q = e_2 \sin \frac{\varepsilon}{2} \qquad r = e_3 \sin \frac{\varepsilon}{2} \quad \text{et} \quad s = \cos \frac{\varepsilon}{2}$$

Nous avons la relation $p^2 + q^2 + r^2 + s^2 = 1$

Le quadruplet (p,q,r,s) représente les paramètres d'Euler.

Le triplet (p,q,r) permet de calculer le quadruplet (e_1,e_2,e_3,ξ) et par conséquent l'orientation de R_j par rapport à R_i [REN 80]

La matrice de passage de R_j à R_i s'exprime alors par :

$$R_{ij} = \begin{bmatrix} 2(p^2+s^2)-1 & 2(pq-rs) & 2(pr+qs) & r_1 \\ 2(pq+rs) & 2(q^2+s^2)-1 & 2(qr-ps) & r_2 \\ 2(pr-qs) & 2(qr+ps) & 2(r^2+s^2)-1 & r_3 \end{bmatrix}$$

translation

$$s = (1 - p^2 + q^2 + r^2)^{1/2}$$

Nous pouvons identifier cette matrice avec celle des cosinus directeurs, il vient alors :

$$p = 1/2\,(\text{sign}\,(r_{32} - r_{23}))\,(r_{11} - r_{22} - r_{33} + 1)^{1/2}$$

$$q = 1/2\,(\text{sign}\,(r_{13} - r_{31}))\,(-r_{11} + r_{22} - r_{33} + 1)^{1/2}$$

$$r = 1/2\,(\text{sign}\,(r_{21} - r_{12}))\,(-r_{11} - r_{22} + r_{33} + 1)^{1/2}$$

Nous obtenons donc à partir de la matrice de passage (cosinus directeurs) l'axe et l'angle de rotation permettant de passer de R_j à R_i.

Ces résultats nous serviront pour définir les déplacements d'un solide dans l'espace.

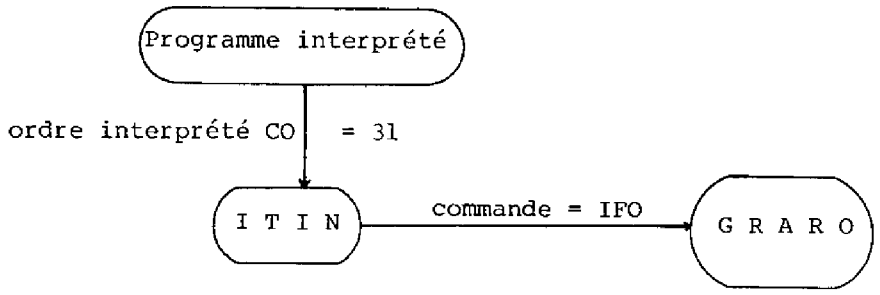
Pour nous, la représentation de base dans le calculateur, exprimant la position et l'orientation d'un repère par rapport à un autre, sera représentée par la matrice 3x4 $R_{ij} = [S_{ij} \mid O_{ij}]$

$$R_{ij} = \begin{bmatrix} \underbrace{r_{11} & r_{12} & r_{13}}_{S_{ij}} & r_1 \\ r_{21} & r_{22} & r_{23} & r_2 \\ r_{31} & r_{32} & r_{33} & r_3 \end{bmatrix} \quad O_{ij}$$

Ceci quelque soit la manière dont l'opérateur définira ces repères.

II.2.2. *Primitives de gestion des repères*

L'ensemble des primitives que nous allons décrire est réalisé par le sous-programme Fortran GRARO ; nous y avons accès par l'intermédiaire du module ITINC appelable par le programme interprété (voir Figure II.4). Tous les sous-programmes (GRARO, ITINC...) sont écrits en Fortran IV, ainsi que l'interpréteur.



II.2.2.1. *Généralités*

Nous générons en fonction de la manipulation envisagée et donc des éléments la constituant (bras manipulateur, capteur, poignet compliant, pièce, plan de contact), un graphe orienté dont l'ensemble des sommets X constituera les différents repères et la famille des arcs U les relations matricielles de passage R_{ij}.

A partir de ce graphe, nous pourrons créer de nouvelles liaisons entre des repères ou rechercher des liaisons déjà existantes. Ceci afin de décrire l'univers robot-environnement et son évolution dans le temps.

L'utilisateur définira suivant ses besoins les sommets et les arcs du graphe orienté, c'est-à-dire les différents repères et matrices de passage.

Nous pouvons classer ces repères par catégories. Suivant la fonction qu'ils occupent, ils peuvent évoluer dans le temps ou rester figés.

.liaisons invariantes :

- physiques : ces repères sont définis comme des points d'un même solide ; ils sont liés entre eux par des arcs invariants. Ils constituent une composante connexe du graphe, ayant les variables du type repère pour sommets et les liaisons entre ces variables pour arcs. Un repère n'appartient qu'à un solide ;
- étalonnées : repère du capteur d'efforts par rapport au repère de la tête du robot, dans lequel sont traduits les déplacements.

.liaisons variables :

- variables, mais mesurables par des capteurs praprioceptifs ;
  exemple : déplacement du repère de contact après mesure des efforts,
  déplacement du robot ;
- variables, mais mesurables par des capteurs exteroceptifs ;
  utilisation d'une caméra, de capteur de proximité ;
- variables étalonnables ou calibrables : création de repères par mesure d'un certain nombre d'informations (détermination de points de contact...).

Pour gérer convenablement le graphe des repères et avoir un outil souple de développement, le calculateur doit gérer dynamiquement le graphe.

En général, le graphe est conçu comme un arbre de changement de repères.

La racine est le repère de référence, noté REF, par rapport auquel tous les autres peuvent être exprimés.

Un repère R_j défini par rapport à un repère R_i est représenté par une matrice R_{ij} (cf II.2.1.2.), à laquelle nous associons un arc orienté.

A l'arc inverse correspondra la matrice R_{ij} (R_i défini par rapport à R_j).

Certaines étapes dans une manipulation demandent la création momentanée d'un circuit dans le graphe. Dans ce cas, nous devons garder la cohérence au niveau du produit des matrices de passage S_{ij} qui doit être égal à la matrice unité : $S_{ii}=S_{ij} \times S_{jk} \times S_{ki} = I_E$. La translation doit être nulle: $0_{ii} = (0\ 0\ 0)^T$.

Au fur et à mesure que l'opérateur crée des repères, le graphe s'implémente.

Cette génération de l'arbre demande la mise en place de plusieurs tableaux mémoires :

- une table contenant des repères R_i = origine d 'un arc,
- une table contenant les repères R_j = extrémité d'un arc,
- une table correspondant aux arcs créés (R_{ij}) définissant un repère R_j par rapport à R_i.

* Recherche de chemin dans le graphe

Quand nous recherchons un chemin entre deux sommets du graphe, afin de créer automatiquement une liaison, nous tenons compte de l'orientation des arcs. Si l'arc existe, nous pouvons l'emprunter ; il est marqué positif s'il est de sens direct,et négatif s'il est de sens inverse.

Dans le cas du graphe sans circuit, le chemin s'il existe est unique. S'il y a des circuits, nous choisirons le chemin qui emprunte le minimum d'arcs pour aller du sommet origine au sommet but.

Nous allons maintenant décrire les primitives de GRARO permettant de définir le graphe et de manipuler des objets vectoriels (point, vecteur, droite, torseur).

II.2.2.2. Définition de repères par l'opérateur

Pour définir un repère R_j par rapport à un repère R_i, l'utilisateur se contente de sélectionner son mode de définition par une commande appropriée de GRARO. Le programme attendra alors l'information correspondant à la définition souhaitée.

Nous pouvons créer un repère R_j par rapport à R_i de différentes façons:

a) par un vecteur à neuf composantes

Ce vecteur est noté E et se décompose comme suit :

. E (1 à 3) coordonnées de 0_{ij}

. E (4 à 6) vecteur représentant un des axes X_{ij} du repère R_j (respectivement Y_{ij} et Z_{ij})

. E (7 à 9) un vecteur du plan associé à l'axe précédent plan ZX ou XY (respectivement YZ ou XY, ZX ou YZ)

La matrice R_{ij} est construite à partir de ces informations, en rendant les vecteurs donnés unitaires et en utilisant le produit vectoriel. Nous obtenons alors les cosinus directeurs du repère R_j dans le repère R_i (cf II.2.1.1.).

b) par un déplacement

Nous nous servons alors des paramètres d'Euler. L'opérateur donne un vecteur E à sept composantes caractérisant le déplacement (cf II.2.1.1). Le programme génère la matrice R_{ij} donnant l'expression du repère R_j dans le repère R_i.

. E (1 à 3) direction de l'axe de rotation e dans R_i

. E (4 à 6) translation subie par 0_i l'amenant en 0_j

. E (7) angle de rotation ε autour de e

La matrice de passage R_{ij} est donnée, à la translation près, par :

$$R_{ij} = I^3 - 2 \sin^2 \frac{\varepsilon}{2} M_e + \sin\varepsilon . \Lambda_e$$

avec

$$\Lambda_E = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}$$

I^3 = matrice identité (3x3)

(p,q,r,s) = quadruplet d'Euler

$$M_e = \frac{1}{1-s^2} \begin{bmatrix} 1-p^2 & -pq & -pr \\ -qp & 1-q^2 & -qr \\ -rp & -rq & 1-r^2 \end{bmatrix}$$

c) par recopie d'une matrice déjà existante

Ceci correspond à la duplication d'un arc du graphe.

Le repère R_j étant défini par rapport à R_i, nous créons le repère R_l par rapport à R_k par une transformation identique à celle passant de R_j à R_i

$$R_{kl} \equiv R_{ij}$$

d) par une transformation identité

Nous exprimons le fait que le repère R_j est identique au repère R_i, en rendant la matrice R_{ij} telle que :

$$R_{ij} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

I^3

La matrice de passage R_{ij} est identique à la matrice identité pour l'orientation, vecteur nul pour la position.

La création de la matrice R_{ij} exprimant le repère R_j dans R_i peut s'obtenir également par la recherche automatique du chemin menant du sommet i au sommet j (§ II.2.2.1.).

II.2.2.3. Opérations sur les repères

Nous avons vu qu'un repère R_j/R_i est représenté par une matrice (3,4) notée R_{ij}.

Nous adopterons dans ce qui suit, la notation suivante :

$$R_{ij} = \left[S_{ij} \;\vdots\; 0_{ij} \right]$$

Toutes les primitives que nous décrivons dans ce chapitre sont écrites en Fortran IV. Un argument X sera noté EX en entrée et SX en sortie.

a) composition de repères : $R_{ik} = R_{ij}\ 0\ R_{jk}$

Nous supposons connus, deux repères

R_{ij} repère j exprimé dans i

R_{jk} repère k ———————— j

Nous cherchons le repère k défini dans i

$$S_{ik} = S_{ij} * S_{jk}$$
$$0_{ik} = S_{ij} * 0_{jk} + 0_{ij}$$

Cette fonction est réalisée par la primitive PRRE (ERIJ,ERJK,SRI)

b) symétrie de la représentation entre deux repères

Nous connaissons R_{ij} (R_j exprimé dans R_i) ; nous cherchons R_{ji} (R_i exprimé dans R_j).

$$S_{ji} = S_{ij}^{T}$$
$$0_{ji} = -(S_{ji} * 0_{ij})$$

$$R_{ji} = [\ S_{ji}\ \vdots\ 0_{ji}\ ]$$

La primitive correspondante est REPI (ERIJ,SRJI)

c) changement de repère d'un point

P_j point P exprimé dans le repère R_j

R_{ij} repère R_j exprimé dans R_i

Nous cherchons P_i, point P exprimé dans R_i

$$P_i = S_{ij} * P_j + 0_{ij}$$

Primitive correspondante : CHRP (EP_j,ER_{ij},SP_i)

d) changement de repère d'un vecteur

Connaissant la matrice R_{ij}, un vecteur V_j du repère R_j aura pour expression dans le repère R_i

$$V_i = S_{ij} * V_j$$

Primitive correspondante : CHRV (EV_j,ER_{ij},SV_i)

e) changement de repère d'une droite

La droite est représentée par le doublet (point, vecteur).

Une droite DR_j dans R_j sera donc exprimée dans R_i en utilisant les deux transformations précédentes.

$$DR_j = (P_j , V_j)$$

$$DR_i = (P_i , V_i)$$

avec $P_i = S_{ij} * P_j + 0_{ij}$

$V_i = S_{ij} * V_j$

Primitive correspondante : CHRD (EDR_j,ER_{ij},SDR_i).

f) changement de repère d'un torseur de force

Un torseur de force T_j est défini dans R_j par :

- un point P_j
- un vecteur F_j représentant les composantes de la force dans R_j
- un vecteur M_j (P_j) représentant les composantes du moment dans R_j par rapport à P_j

Nous exprimerons T_i, torseur T_j du repère R_j, dans le repère R_i, de la manière suivante :

.le point d'application du torseur T_i est ramené au centre du repère R_i ; $P_i = 0_i$ identiquement nul,

.le vecteur de force est transformé comme un vecteur classique

$F_i = S_{ij} * F_j$

.le moment a l'expression suivante

$$M_i(P) = S_{ij} * M_j - \underbrace{\Lambda_{F_i} \cdot (S_{ij} * P_j + 0_{ij})}_{P_i \wedge F_i}$$

REMARQUE

$$\Lambda_{F_i} = \begin{bmatrix} 0 & -F_{iz} & F_{iy} \\ F_{iz} & 0 & -F_{ix} \\ -F_{iy} & F_{ix} & 0 \end{bmatrix}$$

c'est l'écriture matricielle du produit vectoriel

$$A \wedge B = \begin{bmatrix} a_y & b_z - a_z & b_y \\ a_x & b_z + a_z & b_x \\ a_x & b_y - b_x & a_y \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}}_{\Lambda_A} \times \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$$

$$A \wedge B = \Lambda_A \cdot B = -\Lambda_B \cdot A$$

Nous utiliserons par la suite cette notation du produit vectoriel.

Le torseur s'exprime alors

$$T_i \left[0_i , F_i , M_i \right]$$

II.2.2.4. Opérations sur les déplacements

Les primitives définies sur les repères et les opérations qui leur sont associées, ne suffisent pas toujours à décrire toutes leurs évolutions ultérieures.

Nous pouvons introduire la notion de déplacement lié à un repère.

Nous définirons ce déplacement de deux façons :

- par les éléments : axe de rotation, vecteur translation, angle de rotation,
- par une matrice (3x4) définissant un repère.

Les primitives développées à cet égard sont au nombre de quatre.

a) Représentation d'un déplacement par un repère

Les éléments du déplacement (e, ξ , T) étant donnés, nous cherchons le repère correspondant à ce déplacement (Figure II.5).

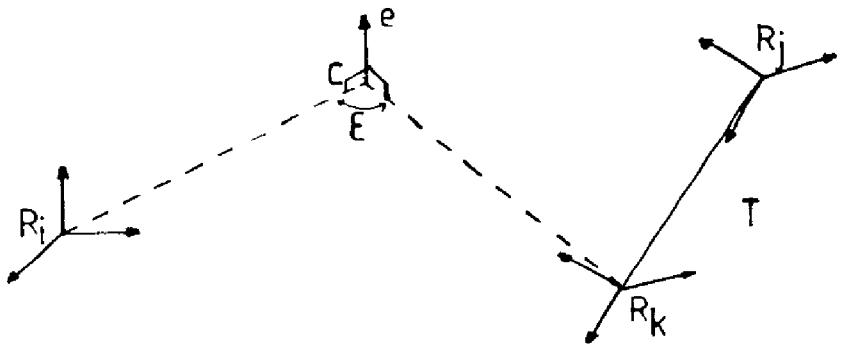


FIGURE II.5 Représentation d'un déplacement par un repère

Le repère R_i est transformé en R_k par la rotation , puis amené en R_j par la translation T.

La primitive correspondante est DFDE (EDE,SRDE)

L'argument d'entrée DE représente le déplacement.

DE (1 à 3) = coordonnées du point D dans R_i : C_i

DE (4 à 6) = vecteur e

DE (7 à 9) = translation

DE (10) = angle ξ

RDE = R_{ij} = $\left[S_{ij} , 0_{ij}\right]$

b) définition d'un repère par un déplacement

C'est l'inverse de la transformation précédente. Le déplacement n'est malheureusement pas unique et nous devons faire un choix dans la définition de ses éléments.

Nous avons choisi le déplacement dont la translation est minimum (parallèle à l'axe de rotation).

Primitive correspondante : DFDI (ERDE,SDE)

RDE = R_{ij} = $\left[S_{ij}, 0_{ij}\right]$

DE : même signification que précédemment.

c) définition d'un déplacement sous forme matricielle à partir d'un repère

Le repère R_i est connu dans R_j par R_{ij}. Nous recherchons le déplacement permettant de passer de R_i à R_j.

L'axe de rotation passe par le centre du repère R_i, la rotation se fera autour de cet axe.

Primitive correspondante : DFRI (ER_{ij},SRE)

RE (1 à 3) = vecteur e dans R_i

RE (4 à 6) = translation dans R_i

RE (7) = ξ , angle de rotation autour de e

Nous pouvons donc à l'aide de tous ces opérateurs, modéliser l'univers englobant le robot et son environnement par des repères orthonormés représentant un graphe orienté (arbre).

Nous étudions ci-dessous les différentes possibilités d'utilisation des repères intervenant dans la définition de l'asservissement du bras manipulateur et donc la représentation de ces déplacements.

II.3. REPRESENTATION DES DEPLACEMENTS DU ROBOT

Tous les repères que nous définirons seront exprimables par rapport à un repère, noté REF, qui joue le rôle de référence absolue. C'est le sommet origine du graphe orienté.

Le système bras manipulateur-poignet "compliant"-capteur d'efforts-pièce saisie, constitue un même solide. Nous lierons donc les repères attachés à chacun de ces éléments par une arborescence.

Référence absolue : noté REF

Repère bras : noté RTRO

Repère capteur : noté RJAU

Repère pièce

(repère asservissement) : noté RCO

Le repère lié au bras manipulateur se situe au sommet du porte poignet, nous le noterons RTRO. C'est dans ce repère que nous explicitons tous les mouvements à générer. Il est parfaitement connu par rapport à la référence.

Le repère de l'asservissement, noté RCO, que nous décrirons dans le paragraphe suivant peut se situer indifféremment sur le solide ou à l'extérieur, selon les besoins de l'utilisateur.

Tout déplacement du bras manipulateur se traduira par un déplacement de RTRO, changement d'orientation compris. Il entraînera en cascade, le déplacement des autres repères qui lui sont solidaires.

L'expression de RTRO par rapport à REF est connue par le modèle cinématique. Nous pouvons donc après chaque mouvement réactualiser la liaison entre ces deux repères ; cette réactualisation étant répercutée sur les repères qui lui sont liés.

Il existe d'autres référentiels transparents pour l'utilisateur, mais qui transforment le déplacement à faire dans un système de coordonnées, propre au manipulateur utilisé, ici, coordonnées cylindriques. Ensuite, il reste l'opération qui consiste à passer des coordonnées cylindriques aux coordonnées généralisées (nombre de pas-moteur).

II.3.1. Repères propres à l'asservissement

Dans le chapitre III, nous décrivons le mode de fonctionnement de l'asservissement. Il nécessite essentiellement la définition de deux repères fondamentaux :

- le repère dans lequel se fait le calcul des coefficients donnant le mouvement sur chaque degré de liberté. Ce repère, RCO, est en général porté par le manipulateur ;
- le repère de consigne de l'asservissement, noté RCCO, qui définit la position et l'orientation que doit satisfaire RCO.

II.3.1.1. Repère de calcul de l'asservissement

Le repère RCO est par définition variable, il sert au calcul de l'asservissement. Dans la plupart des cas, il se situe sur le solide Bras-poignet-pièce.

Néanmoins, nous pouvons le placer à l'extérieur du manipulateur pour des utilisations spéciales.

Nous distinguerons plusieurs modes de définition :

- le cas le plus général est celui où il est variable par rapport à la référence et immobile par rapport au poignet (aux déformations près de celui-ci). Après chaque incrément de déplacement généré par l'asservissement, le repère RTRO est redéfini par rapport à la référence, et il en est de même pour RCO. Il suit donc les déplacements du bras manipulateur ;

- nous pouvons aussi faire varier le repère RCO, non pas en fonction des déplacements du robot, mais après analyse des efforts de contact produits par l'organe terminal. Ceci nous permet de définir le repère de l'asservissement au point où se fait le contact, alors que nous ne connaissons pas les coordonnées de ce point.

Méthode de convergence de RCO vers le point de contact

Rappel

Au contact, nous mesurons un effort de réaction comparé d'un vecteur force et d'un vecteur moment que nous pouvons exprimer dans RCO. Nous représentons le Torseur de force par une droite sur laquelle se trouve le point de contact.

L'idée est de projeter le centre du repère RCO sur la droite représentative du torseur de force ; puis de recréer RCO en gardant les orientations de ses axes, à partir de cette nouvelle origine (Figure II.6). Le précédent repère est détruit. Nous faisons un certain nombre de tests d'appui, au même point, donnant des droites non parallèles entre elles pour faire converger le repère de l'asservissement vers le point de contact.

notation = RCO_i position du repère à l'itération i

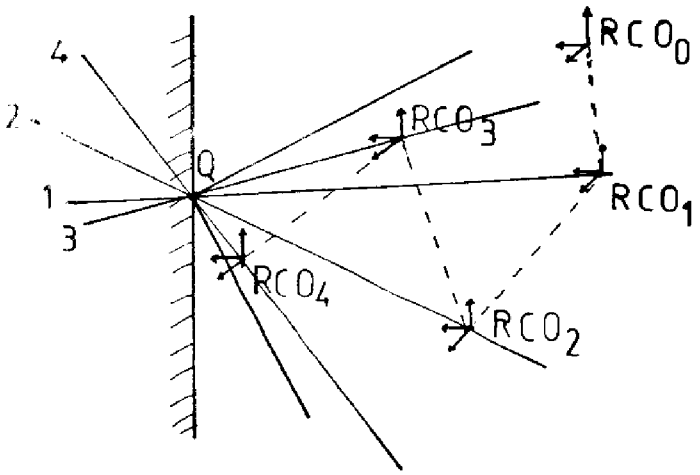


FIGURE II.6 *Convergence du repère RCO vers le point de contact*

Il existe un critère pour vérifier si l'algorithme converge. Il s'agit d'observer l'évolution des moments, exprimés dans le repère RCO. S'ils décroissent jusqu'à devenir très petits, nous pouvons considérer que le point de contact est atteint.

- Dernier cas, le repère RCO est situé à l'extérieur du solide formé par le bras, le poignet et la pièce. Ceci permettrait d'asservir une des extrémités de la pièce à une trajectoire définie suivant les valeurs des coefficients de l'asservissement.

II.3.1.2. Le repère de consigne de l'asservissement

C'est lui qui définit la position et l'orientation finales que doit atteindre RCO.

Nous pouvons le définir de plusieurs manières :
- variable,
- constant,
- étalonnable.

1) Constant

Nous le définissons par rapport à la référence absolue REF. A chaque mouvement du robot, RCO se déplace et recalcule l'expression de RCCO par rapport à lui-même. Ce calcul se fait en utilisant la recherche automatique du chemin menant d'un sommet du graphe à un autre (cf II.2.2.1.).

2) Etalonnable

La détermination des coordonnées d'un point de contact peut être utilisée pour définir RCCO. Supposons que le manipulateur produise le contact sur un plan inconnu. Après une série de tests de mesures, il en déduit les coordonnées du point où s'est fait l'accostage. Le repère RCCO peut alors être construit en ce point et exprimé par rapport à REF.

3) Variable

Dans ce cas, RCCO est défini par rapport à un des repères portés par le bras manipulateur ; le plus souvent, c'est RCO. Quand le bras bouge, sa position par rapport à la référence est réactualisée. Tous les repères qui lui sont liés le sont également, et en particulier RCCO. Le repère RCO ne coïncidera jamais avec le repère RCCO, la liaison entre eux restant constante.

Une autre possibilité est de définir une liaison rigide entre RCO et RCCO et de faire décrire au repère de consigne de l'asservissement une certaine trajectoire.

Les différents types de définition de ces deux repères seront utilisés dans la partie expérimentation pour la mise en oeuvre de diverses manipulations.

II.4. REPRESENTATION GRAPHIQUE

Nous avons développé un logiciel graphique afin de visualiser les évolutions des repères liés au manipulateur, les forces, les déplacements et même la géométrie des pièces en contact.

Nous décrivons ci-après les primitives réalisées et les différents modes de représentation envisagés.

II.4.1. Définition de base

Les primitives de base se regroupent en trois modules Fortran, eux-même appelés par le sous-programme ITIN appelable par le programme interprété (cf Figure II.7).

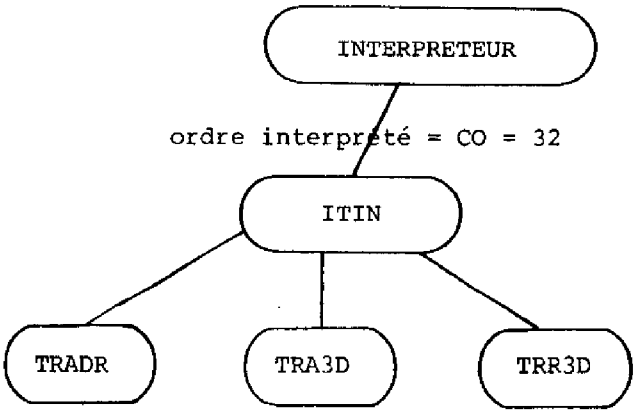


FIGURE II.7

Nous allons exposer le principe de fonctionnement du tracé graphique.

II.4.1.1. Principe

Nous devons définir sur l'écran de la console, une fenêtre de visualisation affectée d'un repère de tracé et des échelles appropriées. L'opérateur doit donc définir un certain nombre de paramètres qui correspondent aux différentes échelles nécessaires au tracé.

1) Définition du repère de tracé

Bien que disposant d'une représentation plane (plan de l'écran de la console), nous avons laissé la latitude à l'utilisateur de définir des repères bidimensionnels ou tridimensionnels orthonormés.

Pour cela, il suffit de donner pour chaque vecteur représentant un axe, ses cosinus directeurs dans le plan.

D'autre part, l'opérateur doit fournir les coordonnées de l'origine du repère de tracé en coordonnées bidimensionnelles et tridimensionnelles.

L'utilisateur pourra après définition du repère de tracé, noté R3D, visualiser en relatif ou en absolu, différents objets vectoriels tels que : points, droites,repères, déplacements et forces.

Les variables à définir par l'opérateur ou à initialiser par programme seront donc :

* R3D : numéro du repère de tracé dans la table de gestion des repères
* orientation du repère R3D
  - X32D (1 à 2) = cosinus directeur dans le plan pour l'axe X
  - Y32D (1 à 2) = ——————————————————— l'axe Y
  - Z32D (1 à 2) = ——————————————————— l'axe Z
* position du repère R3D
  - 033D (1 à 3) = origine des axes tridimensionnels à tracer
  - 032D (1 à 2) = position de 033D dans le plan utilisateur

2) Définition du cadre et des échelles de tracé

Le cadre sera défini par la donnée de trois valeurs, représentant les limites de la fenêtre de visualisation.

Représentation dans le plan.

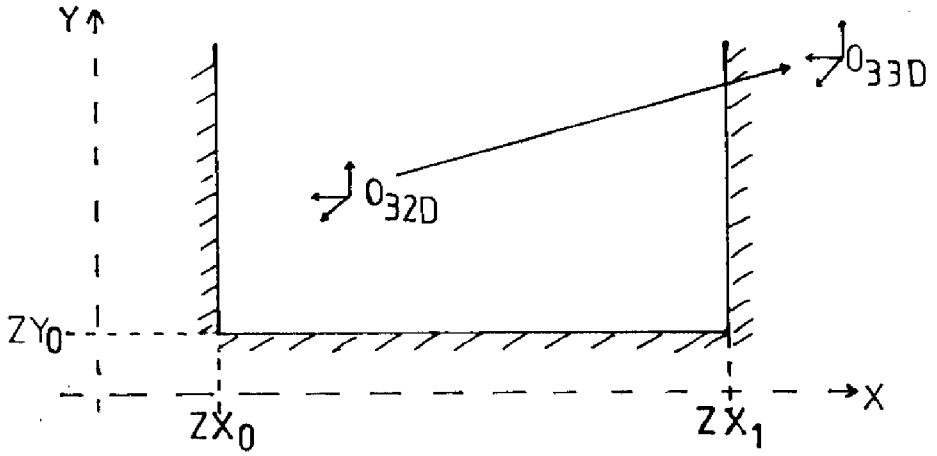


FIGURE II.8

$Z\ X_0$: valeur sur l'axe horizontal de l'extrémité gauche du cadre

ZX_0 0

$Z\ X_1$: valeur sur l'axe horizontal de l'extrémité droite du cadre

$Z\ Y_0$: valeur sur l'axe vertical de l'extrémité inférieure du cadre

.Définition échelle longueur axe et forces

Nous définissons deux variables entières représentant :

- XLAXE : facteur d'échelle donnant la longueur des axes à tracer, suivant les échelles définissant le cadre ;
- ELOF : facteur d'échelle permettant de représenter la longueur d'un vecteur force suivant la valeur de son intensité (ou module).

Exemple : Définition du repère de tracé.

```
NR3D   =  20
XLAXE  =  10          ELOF  =  10
X32D   =  -1     0
Y32D   = -0,5   -0,5
Z32D   =  0      1
033D   =  0      0    0
032D   =  0      0
ZX0 = -100      ZX1 = 100       XY0 = -80
```

La figure II.9 représente l'écran de la console graphique.

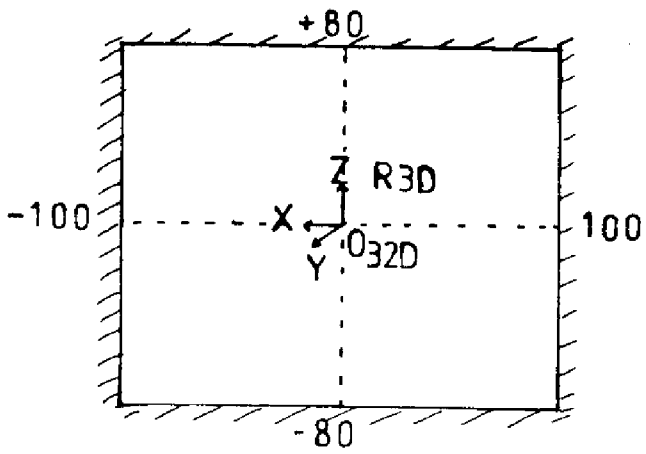


FIGURE II.9 Représentation de l'écran de la console

II.4.2. Description des commandes de tracé

La primitive Fortran ITIN (programme utilisateur) est accessible à l'aide du programme interpréré par la macrocommande C_0=32 suivi de deux arguments IFO, IFA qui sont des nombres entiers.

Nous allons décrire les commandes propres au tracé contenues dans ⟨ITIN⟩ .

ITIN [IFO,IFA]

II.4.2.1. Définition des échelles

- échelle manuelle

  (IFO,IFA) = (41,4)

  L'opérateur définit à la télétype les variables décrites au § II.4.1.1.

- échelle interprétée

  (IFO,IFA) = (41,7)

Les variables de définition des échelles et du repère R3D ont été précédemment définies par le programme interprété, à l'aide de l'écriture directe dans le COMMON / CTRAC.

Cet ordre déclenche l'activation de ces variables sans nécessiter de demande d'information à l'opérateur.

II.4.2.2. Tracé du cadre et effacement

- En fonction des échelles données, nous pouvons tracer le cadre, définissant sur l'écran de la console le domaine de visualisation.

  (IFO,IFA) = (41,3)

- L'effacement total de l'écran est réalisé par :

  (IFO,IFA) = (41,5)

II.4.2.3. Tracé d'un repère N

(IFO,IFA) = (46,N)

N = numéro du repère à tracer

Le repère de numéro N est tracé par rapport à R3D (orientation et position).

Si IFA $<$ 0, le tracé des axes du repère se fait avec temporisation, de telle sorte que l'opérateur puisse les voir se tracer l'un après l'autre.

Nous pouvons grâce à cette commande, visualiser l'évolution d'un corps ou d'une manipulation en traçant systématiquement certains repères (RTRO, RCO....).

II.4.2.4. Tracé des supports des vecteurs de force

Nous pouvons représenter un torseur de force par une droite. Cette commande permet de tracer toutes les droites relatives aux torseurs mesurés.

(IFO,IFA) = (42,N)

N = numéro du repère dans lequel sont tracées les droites.

II.4.2.5. Tracé des forces et des bras de levier

(IFO,IFA) = (47,N)

Nous traçons dans le repère N, le vecteur de force, affecté du facteur d'échelle ELOF, ainsi que son bras de levier comme le montre la figure II.10.

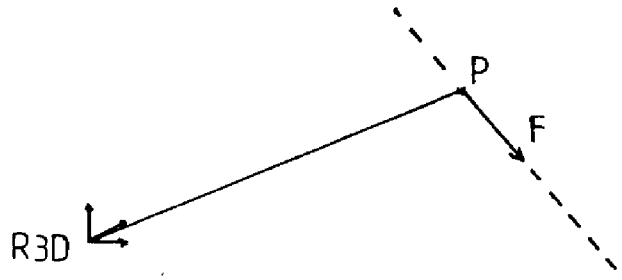


FIGURE II.10 Tracé d'une force et de son bras de levier

II.4.2.6. Tracé propre aux manipulations de contact

Dans la représentation du torseur de force par une droite, nous rangeons après plusieurs mesures les droites dans un tableau.

Puis à partir de la représentation par des droites, nous en déduisons l'informations sur la géométrie de la pièce.

Pour pouvoir visualiser ces opérations, nous avons construit des modules qui permettent dans un repère donné N de tracer les droites discriminantes (essais valides), les points découverts, ainsi que les arêtes.

a) Tracé des droites valides dans un repère N

(IFO,IFA) = (48,N)

N = numéro du repère où sont exprimées les droites.

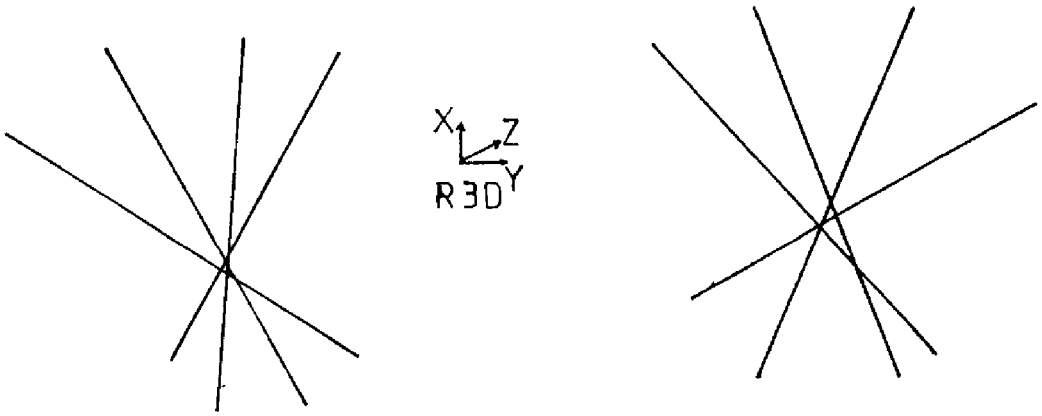


Figure II.11 Tracé des groupes de droites

Nous visualisons après classification, les groupes de droites discriminantes qui conduisent à la détermination du point de contact.

b) Tracé des "points de contact" et des normales aux droites

(IFO,IFA) = (43,N)

N = numéro du repère

L'information comme pour les commandes précédentes, est donnée dans le repère N ; nous la transformons dans R3D, puis nous traçons les points de contact trouvés, ainsi que les normales aux faisceaux de droites auxquelles ils appartiennent (voir Figure II.12).

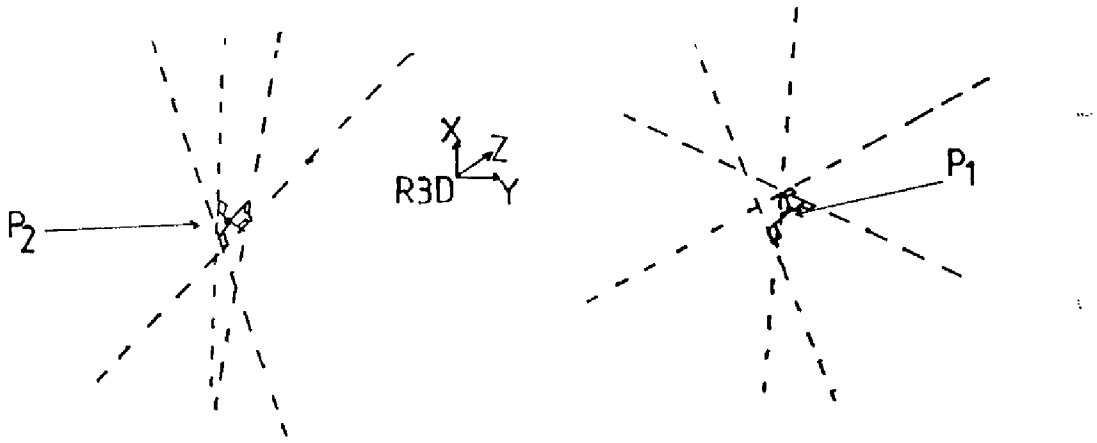


FIGURE II.12 Tracé "des points de contact" et des normales aux droites

c) Tracé des arêtes

(IFO,IFA) = (44,N)

Les arêtes sont des segments de droite (vecteur) connus dans le repère N. Nous les exprimons dans le repère R3D pour les tracer. C'est ce

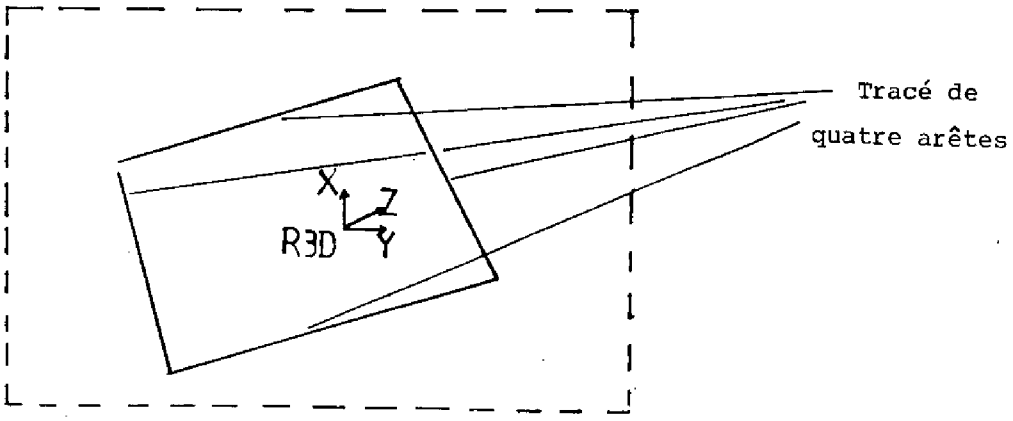


FIGURE II.13 Tracé des arêtes d'une pièce reconnue par contact

II.4.2.7. Tracé des déplacements du robot

(IFO,IFA) = (2,Ø)

Les déplacements du bras manipulateur, exprimés par rapport à la référence (REF) sont convertis dans le repère graphique R3D, puis tracés (cf Figure II.14).

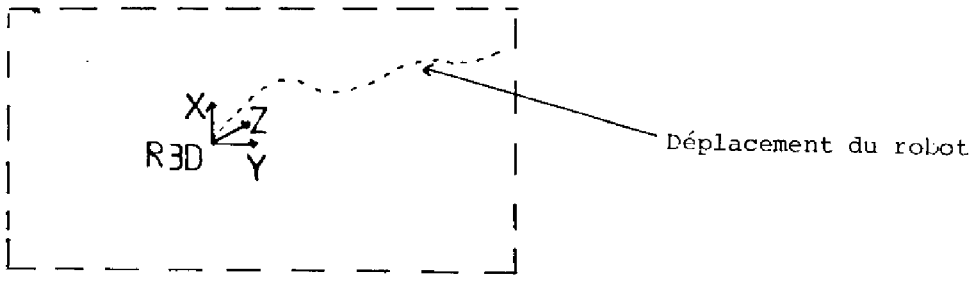


FIGURE II.14 Tracé du déplacement du robot

Ce logiciel graphique rend compte du retour d'information provenant du capteur d'efforts (d'autres capteurs pourraient être utilisés) BEJ 80 . Il permet à l'opérateur de contrôler l'initialisation et le déroulement correct des tâches exécutées par le calculateur.

II.5. CONCLUSION

Les primitives de définition et de gestion dynamique des repères nous ont donné une grande souplesse dans la recherche et la description des caractéristiques géométriques des objets. De plus, cette approche, au niveau des repères de consigne et de calcul de l'asservissement (respectivement RCCO et RCO), offre une bonne maîtrise de l'espace de travail du bras manipulateur.

De plus, l'adjonction du tracé graphique, nous a permis de disposer d'un outil sérieux pour la mise au point et le développement des manipulations.

Le logiciel informatique, relatif à ces primitives, a nécessité une importante programmation ; écrit en Fortran IV et actuellement implanté sur le minicalculateur Mitra 15, sa structure modulaire laisse envisager un transport facile sur un autre calculateur ainsi que son extension dans les domaines suivants :

- le développement de la maîtrise de la gestion des vitesses, dans la commande des mouvements du robot,
- la définition de matrices de rigidité, relatives au système bras-manipulateur-poignet-capteur.

Ceci devrait nous permettre d'améliorer la maîtrise de l'action du robot sur son environnement quand il est asservi en force et en position, notamment dans les manipulations de montage.

C H A P I T R E III

MAITRISE DES FORCES

III.1. INTRODUCTION

Nous allons définir dans ce chapitre les bases mathématiques et conceptuelles de la stratégie qui permettra à un robot aveugle une appréhension d'un univers, dont il ne prend connaissance qu'en se déplaçant et en recueillant l'information issue d'un capteur d'efforts. Le capteur, monté à l'extrémité du robot, délivre après traitement par un microprocesseur, un torseur de force, exprimé dans un repère de référence.

Le robot ne connaissant que l'expression des forces et des moments produits au contact d'un obstacle, il est primordial de bien connaître les propriétés du contact.

Le contact ponctuel, point sur plan, est de loin le plus riche en information, mais aussi le plus facile à mettre en oeuvre.

Nous allons donc décrire ce type de contact et expliciter les renseignements qu'il peut apporter. A partir de cette étape, nous parlerons de l'utilisation de ces propriétés pour acquérir une information, non plus forces-moments, mais sur l'espace. Comment décrire par exemple les caractéristiques dimensionnelles de la pièce étudiée, à partir d'une séquence de tests mettant en oeuvre une série de contacts.

Bien entendu, la mesure de ces efforts et les tests envisagés, nécessitent des déplacements du robot ; nous exposerons donc le modèle d'asservissement choisi, qui nous paraît le mieux adapté à notre problème.

III.2. LE CONTACT PONCTUEL

Nous disposons d'un capteur d'effort, nous permettant de connaître les forces et les moments appliqués à l'organe terminal du robot, dans un repère parfaitement connu, situé sur le robot et noté RJAU.

Quand une pièce tenue par le bras manipulateur est mise en contact avec un plan extérieur, nous mesurons dans RJAU un torseur de force rendant compte de l'état de ce contact.

Dans le cas plus particulier du contact ponctuel, l'exploitation de ses propriétés doit nous permettre d'accéder à la connaissance des coordonnées du point où se fait le contact. Si la pièce saisie par le robot est tout ou partie inconnue, cette approche nous fournira des caractéristiques géométriques sur cette pièce. De plus, dans le cadre de l'insertion d'un goujon dans un trou, il s'avère intéressant de connaître le point où la pièce est en contact avec la paroi du trou.

III.2.1. Caractérisation

Soit comme le montre la figure III.1, un des sommets Q du corps A en contact avec un point du plan B.

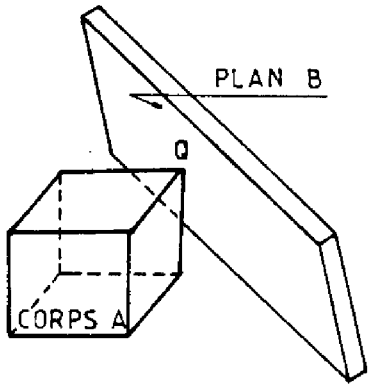


FIGURE III.1 Contact ponctuel

En ce point Q où il y a contact, nous ne connaissons pas la direction de la force résultante due à la réaction du plan. Mais si le contact est parfaitement ponctuel, les moments produits par la force F au point Q sont nuls. Pour nous, cette propriété caractérisera le contact ponctuel (cf Figure III.2).

Nous chercherons donc à partir de la connaissance du torseur de force dans un référentiel donné, s'il existe un ou plusieurs points d'application du torseur précédent par rapport auquel les moments auraient une valeur nulle.

Nous allons formaliser le problème dont la résolution doit nous conduire à l'obtention des coordonnées du point de contact Q.

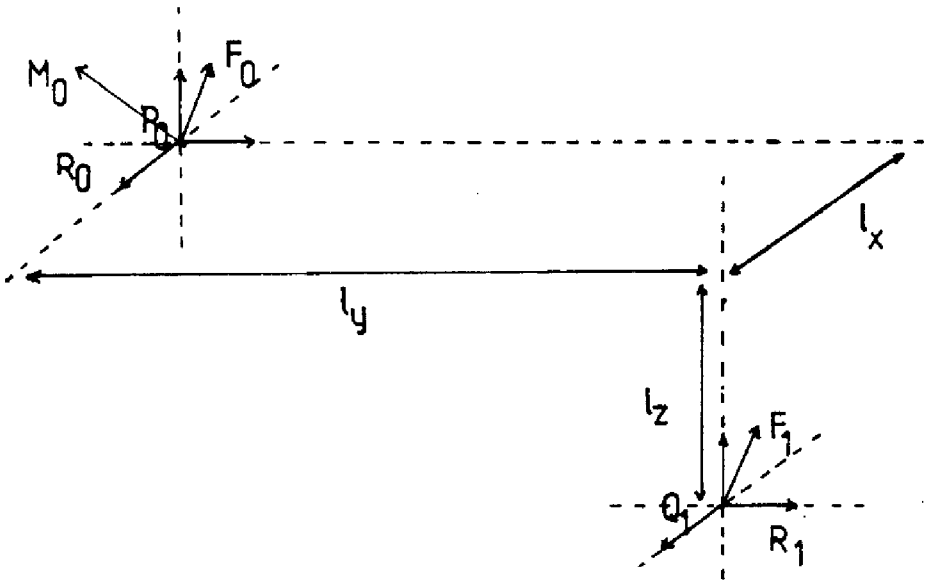


FIGURE III.2

R_1 = référentiel où se produit le contact

R_0 = référentiel où nous connaissons la valeur des efforts

$$M_1(Q_1) = \begin{bmatrix} M_{1x}\ (Q) \\ M_{1y}\ (Q) \\ M_{1z}\ (Q) \end{bmatrix} \equiv 0 \qquad M_0(P_0) = M_1(Q_1) + P_0Q_1 \wedge F_1$$

III.2.2. *Formalisme du problème*

* Hypothèse de départ

Nous connaissons dans un référentiel donné, l'expression du torseur de force, engendré par la mise en contact du corps A avec le corps B.

Nous travaillons bien sûr, avec des repères de type orthonormé.

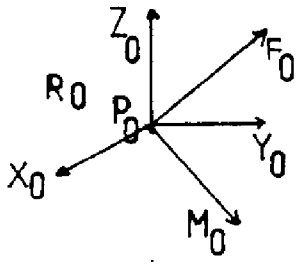


FIGURE III.3 Référentiel de mesure

Soit dans le repère de mesure des efforts, R_0, (cf Figure III.3) le torseur T_0 défini par le triplet :

$$T_0 \left[P_0 , F_0 , M_0\right]$$

.P_0 est le point d'application du torseur, c'est aussi l'origine du repère R_0

.F_0 est le vecteur de force exprimé dans le repère R_0

$$F_0 = (F_{0X} , F_{0Y} , F_{0Z})^T$$

.M_0 est le vecteur moment, exprimé par rapport au point P, dans le repère R_0

$$M_0 = (M_{0X} , M_{0Y} , M_{0Z})^T$$

Nous cherchons le torseur T_1, dont le point d'application Q_1 est tel que son moment est nul, et son vecteur de force identique en module à celui de T_0. Nous associerons l'origine du repère R au point Q calculé.

$$T_1 \left[Q_1 , F_1 , M_1\right] \quad \text{tel que} \quad \begin{matrix} F_1 \equiv F_0 \\ M_1(Q) \equiv 0 \end{matrix}$$

Nous en déduisons l'équation vectorielle (1)

$$M_1(Q_1) = M_0(P_0) + P_0Q_1 \wedge F_1 \equiv 0 \qquad (1)$$

La condition nécessaire d'existence des solutions de cette équation vectorielle est telle que =

Le produit scalaire du vecteur de force par le moment en P_0 soit nul.

$$F_0^T \cdot M_0(P_0) = 0$$

a) $F_0^T \cdot M_0(P_0) = 0$ avec $F_0 = 0$

Si $M_0(P_0) = 0$ tout vecteur P_0Q_1 est solution

$M_0(P_0) \neq 0$ nous sommes en présence d'un couple. Pas de solution ponctuelle

Le cas a) ne se produit pas dans la pratique, car nous recherchons un contact point sur plan ; donc le vecteur de force n'est jamais identiquement nul.

b) $F_0^T \cdot M_0(P_0) = 0$ avec $F_0 \neq 0$

Supposons qu'il existe une solution particulière P_0Q_0. Nous aurons :

$(P_0Q_1 - P_0Q_0) \wedge F_0 = 0 \longrightarrow \exists \alpha \in R / P_0Q_1 - P_0Q_0 = \alpha . F_0$

qui exprime que les vecteurs F_0 et $(P_0Q_1 - P_0Q_0)$ sont colinéaires. Nous pouvons déterminer cette solution particulières de la manière suivante.

Nous remplaçons P_0Q_1 par $F_0 \wedge M_0(P_0)$

d'où $(F_0 \wedge M_0(P_0)) \wedge F_0 = (F_0^T . F_0) \cdot M_0(P_0) - (M_0(P_0) \cdot F_0) \cdot F_0$

$= 0$

$$(F_0 \wedge M_0(P_0)) \wedge F_0 = F_0^T F_0 \cdot M_0(P_0)$$

$$\frac{(F_0 \wedge M_0(P_0) \wedge F_0}{F_0^T F_0} = M_0(P_0)$$

or $M_0(P_0) = - P_0Q_0 \wedge F_0$ d'après (1)

Nous obtenons alors la solution particulière :

$$P_0Q_0 = \frac{M_0(P_0) \wedge F_0}{F_0^T F_0} \qquad (2)$$

La solution générale de l'équation s'écrit alors

$$P_0Q_1 = \frac{M_0(P_0) \wedge F_0}{F_0^T F_0} + \alpha . F_0 \qquad (3)$$

Les point Q_i cherchés, décrivent une droite parallèle au vecteur F_0 et issue de Q_0.

Nous avons donc plusieurs solutions possibles à notre problème. Mais il nous faut un complément d'information pour aboutir à l'obtention des coordonnées du point où se fait le contact. En l'occurrence pour déterminer les coordonnées du point, il nous faut au minimum connaître deux droites non colinéaires.

III.2.3. Méthode d'obtention des coordonnées du point de contact

III.2.3.1. Représentation du contact par une séquence de tests

L'idée de départ est de produire, au même point Q, des efforts de directions différentes, de manière à obtenir plusieurs droites représentant chacune les lieux des points par rapport auxquels les moments mesurés sont nuls. Nous noterons ces droites DR_i.

En faisant plusieurs tests (contact ponctuel) pour un même point et en enregistrant à chaque fois les forces et les moments réactifs, respectivement pour chaque test , nous pourrons en déduire les équations des droites DR_i, propres à chaque essai.

Nous trouverons les coordonnées du point cherché, en faisant l'intersection de ce faisceau de droites. Théoriquement, les droites DR_i doivent toutes se couper en Q, et deux droites suffiraient pour la solution du problème. Malheureusement, il est difficile d'avoir expérimentalement un contact ponctuel parfait. De plus, le capteur d'efforts fournit un signal entaché d'erreur, aussi faible soit-elle.

Tout ceci nous conduit à obtenir, après calcul, un système d'équations linéaires représentant les droites DR_i, qui n'a pas forcément une solution.

Nous aurons, dans le cas général, des droites DR_i non sécantes. Nous sommes donc amenés à rechercher l'intersection de droites non concourrantes. Ce problème trouve sa solution en mettant en oeuvre une méthode du type des moindres carrés.

III.2.3.2. Calcul du point de contact

III.2.3.2.1. Représentation d'un torseur de force par une droite

Nous avons vu que les points Q décrivent une droite, donnée par l'équation (3).

Les efforts sont mesurés dans le repère R_0, qui a le point P_0 comme origine. Nous connaissons donc P_0, F_0 et $M_0(P_0)$; il nous est facile de calculer les coordonnées du point Q_0 (2).

Nous pouvons passer, à partir de la donnée du torseur dans un repère R_0, à une représentation avec des droites. Pour un torseur donné $T_0 \left[P_0, F_0, M_0\right]$, nous génèrerons la droite D_0, qui sera définie par

.un point $Q_0 = \dfrac{M_0(P_0) \wedge F_0}{F_0^T \cdot F_0}$ dans R_0

.un vecteur $F_0(F_{0X}, F_{0Y}, F_{0Z})$ dans R_0.

Quand nous travaillerons sur des droites, nous garderons ce type de représentation.

III.2.3.2.2. Description de la méthode choisie

Nous savons donc représenter le contact ponctuel par un faisceau de droites, dont la propriété est qu'elles passent toutes dans un domaine voisin du point cherché.

Les différents tests mis en oeuvre pour un même point Q seront choisis de telle manière que les droites DR_i obtenues soient suffisamment discriminantes. C'est-à-dire, qu'elles convergent bien vers la région où se situe le point de contact, comme le montre la figure III.4. Deux essais identiques conduisant à deux droites parallèles ne seront pas discriminants.

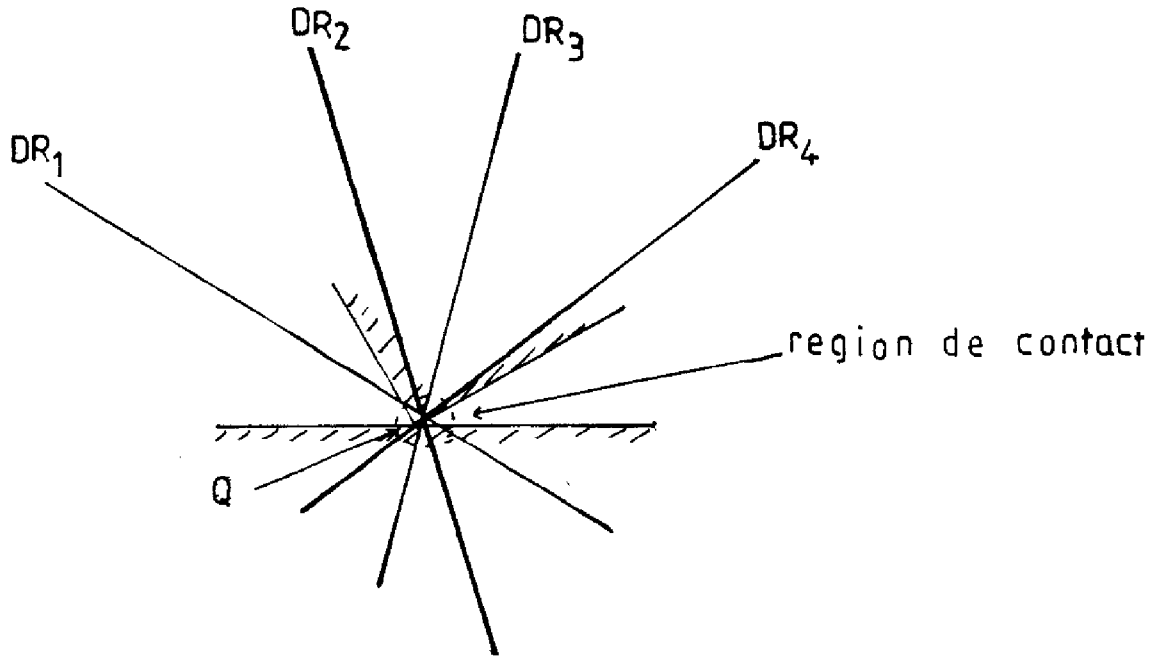


FIGURE III.4 Représentation des tests permettant d'obtenir les coordonnées du point Q

Le problème est en fait de trouver à partir de cette famille de droites, les coordonnées du point qui minimise les distances entre toutes ces droites.

Nous avons besoin pour ce calcul de quelques propriétés mathématiques telles que :

- Distance d'un point à une droite

La droite DR étant définie par un point et un vecteur DR(P,V), la distance d'un point Q à cette droite sera donnée par :

$$d^2 = (Q-P)^T \; M_V (Q-P) \qquad (3)$$

avec $M_V = I - \frac{VV^T}{V^T V}$ qui est une matrice (3x3)

I étant la matrice identité (3x3)

. Coordonnées d'un point Q, qui minimise la somme des carrés des distances à une famille de droites $DR_i \; (P_i, V_i)$

$$Q = \left(\sum_i M_{V_i} \right)^{-1} \cdot \left(\sum_i M_{V_i} \cdot P_i \right) \qquad (4)$$

M_V a la même signification que précédemment

. Calcul de la somme D^2 des carrés des distances d'un point Q à une famille de droites $DR_i \; (P_i, V_i)$

$$D^2 = \sum_i P_i^T M_{V_i} P_i - \left(\sum_i P_i^T M_{V_i} \right) \left(\sum_i M_{V_i} \right)^{-1} \left(\sum_i M_{V_i} Q_i \right) \qquad (5)$$

Nous utiliserons ces différents résultats dans la recherche des coordonnées du point de contact.

Nous avions fait l'hypothèse que tous les essais générant les droites d'une même famille étaient entachés d'erreur. Nous allons donc construire un algorithme de classification, qui nous permettra de rejeter des droites aberrantes appartenant à une même famille ; nous pourrons alors approcher au plus près les coordonnées du point de contact que nous cherchons à déterminer.

Algorithme de classification

Chaque droite représentative d'un test de contact est décrite par DR.

A l'initialisation toutes les droites correspondant aux essais effectués sont dans la classe 0.

.Etape 1

Nous calculons les coordonnées du point Q qui minimise la somme des carrés des distances aux différentes droites contenues dans la classe 0.

.Etape 2

Les coordonnées du point Q ayant été obtenues, nous calculons pour chaque droite la valeur de la distance du point Q à la droite considérée.

.Etape 3

A chaque droite est affecté un numéro dans la classe. Nous cherchons le numéro de la droite dont la distance au point Q est maximum. Si cette distance est inférieure à une distance seuil, nous décidons d'éliminer la droite incriminée en la transférant dans la classe -1 (resp.0). Nous revenons alors à l'étape 1.

Dans le cas où $D_{max} < D_{seuil}$, nous créons une nouvelle classe qui contiendra les droites restant dans la classe 0.

Cette classe est représentative d'un point de contact, dont nous connaissons les coordonnées. L'ensemble est gardé en mémoire dans le calculateur et réutilisable à tout moment.

La classe -1 contenant les droites éliminées est alors prise comme classe de départ, et nous retournons à l'étape 1.

La distance seuil qui permet de faire la discrimination entre "bonnes" et "mauvaises" droites est réglable par l'utilisateur.

.Etape 4

Si il n'y a qu'une seule droite dans la classe nouvellement créée, ou bien dans la classe 0, nous la détruisons.

.Etape 5

Arrêt = L'algorithme s'arrête quand il n'y a plus de droites à classer (classe 0 est vide).

Cet algorithme ne présente pas un caractère général dans la mesure où il ne permet pas de retrouver les groupes de droites appartenant à différents tests relatifs à plusieurs points de contact. Néanmoins, il trouve son utilité dans l'élimination d'essais non valides pouvant entraîner des erreurs non négligeables dans le calcul du point de contact.

Nous avons décrit comment à partir d'un contact ponctuel, en appliquant une séquence de tests, nous pouvons obtenir les coordonnées du point de contact et par-là même une information sur la géométrie de la pièce tenue par le robot.

Nous pouvons par la même procédure, décrire tous les points anguleux de la pièce, en faisant par exemple subir des rotations à celle-ci, puis en revenant faire le contact sur un plan de travail. Nous nous retrouvons alors dans la phase d'apprentissage des coordonnées du point où se fait le contact. Ces tests peuvent être systématisés, mais ne disposant d'un manipulateur n'ayant qu'un seul degré de liberté en rotation, les applications sont limitées pour pouvoir reconnaître une pièce de forme compliquée.

Nous verrons dans le chapitre expérimentation (cf VI.3.) quel type de séquence de test nous avons choisi.

D'autre part, à partir de la connaissance des points anguleux de la pièce tenue par le robot, nous pouvons en déduire un certain nombre d'arêtes (en reliant les points entre eux). Malheureusement, toutes les arêtes trouvées ne sont pas représentatives, seules celles limitant l'objet sont à prendre en considération.

Nous pouvons donc envisager une séquence de tests complémentaires, fondée sur la détection non plus d'un point, mais d'une arête toute entière. Néanmoins, le contact effectué sur l'arête à détecter doit rester ponctuel, pour garder la cohérence avec les calculs développés.

III.3. DETERMINATION D'UNE ARETE

Nous avons mis en oeuvre cet opérateur à partir de l'exploitation des propriétés du contact ponctuel.

En effet, nous pouvons représenter l'information sur le torseur de force par une droite, ayant la direction de la force et passant par le point de contact.

Dans le cas du contact sur une arête, nous devons appuyer sur celle-ci en différents endroits et enregistrer après chaque essai la valeur des forces et des moments. Pour délimiter convenablement l'arête, nous prenons deux mesures aux extrémités de celle-ci dans les points d'angles.

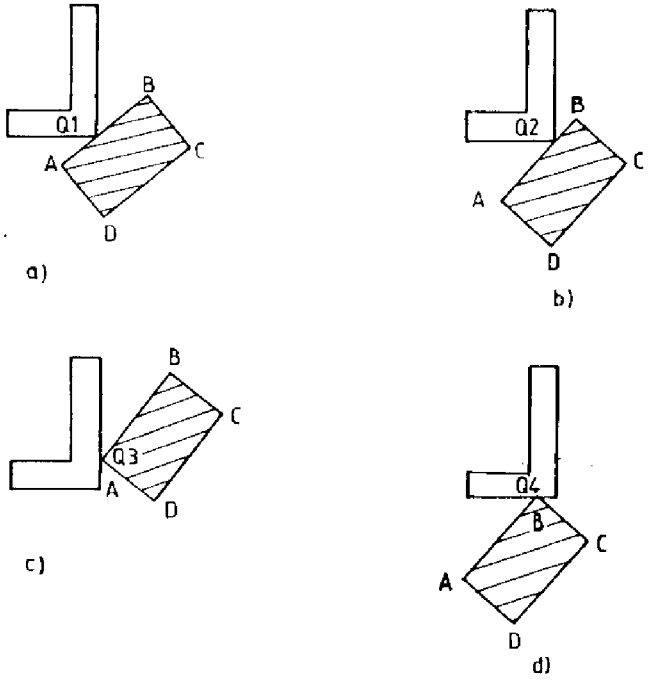


FIGURE III.5 Tests de contact pour la détermination d'une arête

Les tests relatifs à la figure III.5.a et b sont effectués en des points quelconques de l'arête, mais suffisamment espacés. Les tests relatifs à la figure III.5.c et d délimitent les points extrémums de l'arête. Il est nécessaire d'obtenir au moins quatre droites donc quatre tests.

Les droites obtenues s'appuient théoriquement sur une droite qui est l'arête cherchée. Les droites représentatives des essais (c) et (d) nous donnent en fin de clacul, les bornes de l'arête.

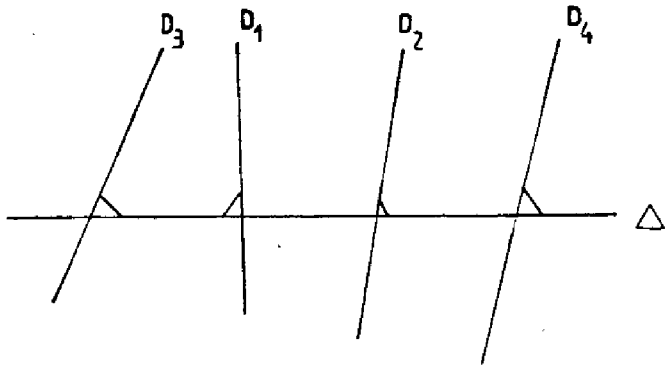


FIGURE III.6 La droite Δ représente la direction de l'arête

Pratiquement, nous devons résoudre le problème de la recherche de la droite Δ passant le plus près d'une famille de droites D_i et minimisant donc la somme des carrés des distances à ces droites D_i. Nous n'avons pas pu mettre en oeuvre une méthode analytique de résolution du type "méthode des moindres carrés" et nous avons dû utiliser une méthode itérative.

.Principe

Nous définissons la droite cherchée par un point et un vecteur unitaire V. Le vecteur V peut donc être défini à l'aide de deux angles A et B. IL définit alors un plan E passant par l'origine de l'espace.

En projetant les droites D_i dans le plan E, nous obtenons un "treillis" de droites D_i^P. Le point minimisant la somme D des carrés des distances entre ce point et ses projetés sur les droites D_i^P est appelé centre. Cette valeur D est une fonction de la direction de projection V et donc D = f(A,B) en minimisant D par rapport aux deux variables A et B nous obtenons un couple (A,B) donc une direction V et un point CE (centre) obtenu dans le plan E associé à cette direction le point CE et le vecteur V définissent l'arête cherchée.

La fonction f(A,B) est minimisée par une méthode de Newton couplée à une méthode de Partan. En effet, pour minimiser la fonction, il faut s'assurer pas à pas de la diminution de la valeur de cette fonction car la méthode peut mener à un maximum. D'autre part, l'incrément (DX,DY) calculé par cette méthode peut atteindre des valeurs soit très petites, soit très grandes qui peuvent entacher les calculs, d'erreurs non négligeables. Nous devons donc nous assurer de l'amplitude de ces incréments.

Dans le cas où la méthode de Newton conduit vers un maximum, nous utilisons donc la méthode de Partan avec Division du pas.

La droite obtenue doit être limitée. Nous projetons alors les droites D_i sur Δ ; les points les plus éloignés déterminent les extrémités de la droite.

Ces calculs peuvent compléter efficacement, ceux effectués pour la détermination d'un point de contact.

De même, nous pouvons envisager la détermination d'un plan, par toucher successif de l'organe terminal du robot sur ce plan inconnu.

III.4. CALCUL DES DEPLACEMENTS DE L'ORGANE TERMINAL DU ROBOT

Les calculs précédents ont fait apparaître qu'à partir d'une information de contact (force - moment) nous pouvions obtenir des caractéristiques dimensionnelles d'un objet à l'origine inconnue. Par exemple, détermination des points anguleux, des arêtes ou bien d'un plan.

La mise en pratique de ces contacts puisqu'ils sont fait par le robot lui-même, nécessite que ce dernier génère un certain nombre de déplacements. Nous allons donc expliciter la manière dont devra se mouvoir le robot pour parvenir à réaliser les fonctions complexes que nous désirons [KEN 80].

III.4.1. L'asservissement

Tel que nous l'avons conçu, l'asservissement, en fonction de la valeur des efforts qu'il mesure par l'intermédiaire du capteur, générera des petits déplacements qui nous amèneront à la consigne désirée. Deux modes sont possibles :

1) Cinématique

Si l'asservissement ne tient pas compte des forces d'inertie, la commande des degrés de liberté est réalisée en mode cinématique. Au cours d'un déplacement, l'automate ne peut alors demander que le passage du manipulateur par des positions successives avec une vitesse moyenne d'évolution.

2) Dynamique

Si l'asservissement tient compte des forces d'inertie, la commande peut être réalisée en mode dynamique. Nous imposons alors non seulement le passage du manipulateur par des positions successives, mais aussi les vitesses instantanées des degrés de liberté à ces positions.

Avantages et inconvénients des deux méthodes

L'avantage du mode cinématique est la simplicité ; son inconvénient est l'incompatibilité entre la rapidité et la précision d'un déplacement.

L'avantage du mode dynamique est la précision, mais ce mode pose des problèmes de modélisation et de calcul en temps réel qui sont encore mal résolus.

La commande en mode cinématique est celle que nous avons retenue. Nous détaillerons dans le chapitre V le passage des déplacements calculés par l'asservissement à leur réalisation effective par l'organe terminal du manipulateur.

III.4.1.1. Description générale

L'asservissement développé permettra de contrôler en force et en position les déplacements d'un robot à six degrés de liberté, comprenant trois translations suivant les axes d'un repère orthonormé et trois rotations autour de ces mêmes axes . Il se décompose en asservissements élémentaires s'appliquant à chaque degré de liberté. Nous réalisons alors une commande indépendante sur chaque axe.

Nous définissons un repère orthonormé, noté RCO, qui correspondra au référentiel dans lequel seront exprimés les mesures d'efforts et le calcul des déplacements que le robot doit effectuer. Ces mouvements feront évoluer le manipulateur vers la consigne à atteindre qui peut être définie comme suit :

a) Force-moment

La consigne sur les forces et les moments sera décrite par un vecteur à six composantes, noté $F^*(i)$

$$F^*(i) = \begin{bmatrix} F^*_x \\ F^*_y \\ F^*_z \\ M^*_x \\ M^*_y \\ M^*_z \end{bmatrix} = \begin{bmatrix} F^*(1) \\ F^*(2) \\ F^*(3) \\ F^*(4) \\ F^*(5) \\ F^*(6) \end{bmatrix}$$

L'indice i variant de un à six indique le degré de liberté correspondant.

Nous représenterons de même les forces et les moments mesurés dans le repère RCO par un vecteur à six composantes noté F(i)

$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} \\ f \\ \\ \text{----} \\ \\ m \\ \\ \end{bmatrix}$$

Nous utiliserons comme notation $F^*(i)$ et $F(i)$ pour bien souligner que l'asservissement est appliqué sur chacun des axes.

b) Position et orientation

La consigne de position et d'orientation sera représentée par un repère orthonormé RCCO (cf II.3.1.2.). Le vecteur de consigne est obtenu en exprimant RCCO par rapport à RCO. Il sera noté $X^*(x,y,z,\theta,\psi,\Phi)$.

La satisfaction d'une de ces deux consignes (a ou b) entraînera l'arrêt de l'asservissement sur l'axe considéré.

Les repères RCO et RCCO sont définis par l'utilisateur suivant le problème envisagé. L'organisation générale de ces repères et leur lien avec les autres référentiels ont été décrits au chapitre II.

Suivant la valeur des forces et des moments mesurés, ainsi que celle des différentes consignes données par l'opérateur ou initialisées par programme, nous exprimons le déplacement de la tête du robot, dans le repère RCO, de la manière suivante.

$$\Delta D(i) = \alpha(i) \cdot \delta d_m(i) \qquad \text{pour le degré de liberté } i$$

$$\delta d_m = (\delta dx_m , \delta dy_m , \delta dz_m , \delta d\theta_m , \delta d\psi_m , \delta d\phi_m)^T$$

$$\Delta D = (\Delta DX, \Delta DY, \Delta DZ, \Delta D\theta , \Delta D\psi , \Delta D\phi)^T$$

$$\alpha = (\alpha_x, \alpha_y, \alpha_z, \alpha_\theta , \alpha_\psi , \alpha_\phi)^T$$

$\Delta D(i)$ est l'incrément de déplacement calculé sur l'axe i

$\delta d_m(i)$ est l'incrément maximum de déplacement sur l'axe i

$\alpha(i)$ sont les coefficients qui sont calculés par l'asservissement en force et position

Nous pouvons schématiser l'asservissement de la manière suivante :

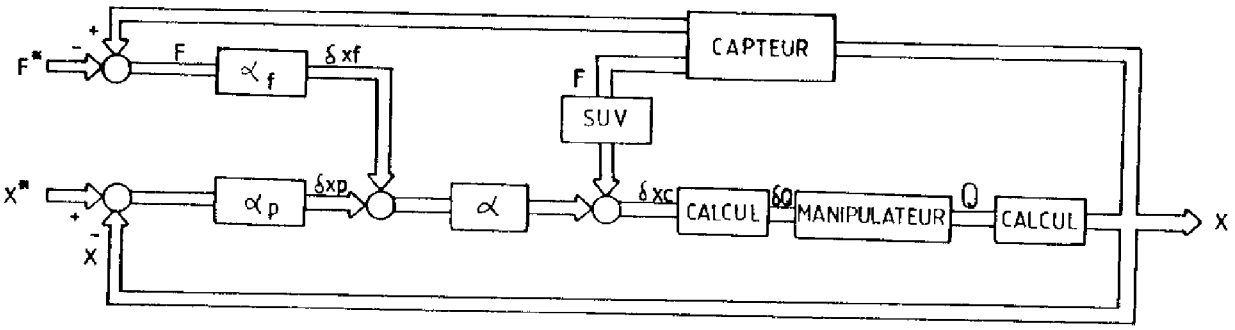


FIGURE III.7 *Schéma de l'asservissement en force et position*

Les incréments de déplacement étant calculés dans le repère RCO, ils doivent être exprimés dans le repère RTRO. Nous obtenons alors la position à atteindre.

$$x_{n+1} = x_n \pm \Delta D$$

La position ainsi obtenue est reconvertie dans l'espace des coordonnées généralisées et soustraite de la position précédente gardée en mémoire. Nous obtenons alors l'incrément de déplacement δQ des coordonnées généralisées. Cet incrément est transmis au microprocesseur qui gère la commande des moteurs du bras manipulateur. Il renvoie alors sa position actuelle Q et exécute la commande.

Le vecteur $\delta d_m(i)$ sera à définir par l'utilisateur (écrit par programme).

Nous avons défini plusieurs niveaux de priorité dans l'asservissement, afin d'éviter, puisque nous nous déplaçons en aveugle, une détérioration éventuelle du matériel ; ils nous donneront également une bonne modularité et une grande souplesse.

III.4.1.1.1. Niveau 1 = Surveillance du module des forces et moments

Nous définissons à l'initialisation, deux grandeurs en module :

- une pour la force, notée FFSUV,
- une pour le moment, notée FMSUV.

Après acquisition du torseur de force dans le repère de mesure (capteur) et transformation dans le repère de l'asservissement RCO, nous comparons le module de la force mesurée à FFSUV, puis le module du moment à FMSUV. Si l'un de ces modules dépasse les valeurs fixées, aucun déplacement n'est généré. Les coefficients α_i (i=1 à 6) sont tous nuls.

Nous positionnons également une variable appelée état de l'asservissement et notée I ETAT, à une certaine valeur pour que l'utilisateur puisse déceler l'origine de l'arrêt.

Si $\exists\ f(i) \;/\; |f(i)| > FFSUV \Longrightarrow \forall \alpha(i),\ \alpha(i) = 0$, I ETAT = 3

Si $\exists\ m(i) \;/\; |m(i)| > FMSUV \Longrightarrow \forall \alpha(i),\ \alpha(i) \equiv 0$ (i=1 à 6) , I ETAT = 4

La priorité est donnée au module de la force ; elle sera donc examinée en premier ; si le test n'est pas valide, nous passons à l'examen du module des moments. Dans le cas où les deux tests ne sont pas satisfaits, nous passons au niveau 2 de l'asservissement.

III.4.1.1.2. Niveau 2 : Surveillance par axe

Nous définissons à l'initialisation du programme six grandeurs correspondant aux trois composantes d'une force et aux trois composantes d'un moment exprimés dans un repère orthonormé.

Cette grandeur est notée FSUV

.FSUV (1 à 3) = F_X, F_Y, F_Z

.FSUV (4 à 6) = M_X, M_Y, M_Z

Remarque

Ces valeurs peuvent être modifiées par l'utilisateur, en mode conversationnel, grâce au programme interprété comme nous le verrons au chapitre V.

Après passage par le niveau 1, le torseur de force exprimé dans le repère RCO est comparé aux valeurs de FSUV.

La comparaison se fait pour chaque axe.

Compte tenu de la valeur mesurée pour les forces, nous aurons soit une translation, soit aucun déplacement, respectivement sur chaque axe. De la même façon, compte tenu de la valeur des moments, nous aurons, soit des rotations, soit aucun déplacement respectivement suivant chaque axe.

Nous pouvons représenter cette fonction par le diagramme ci-dessous. Il est identique pour les trois translations et les trois rotations.

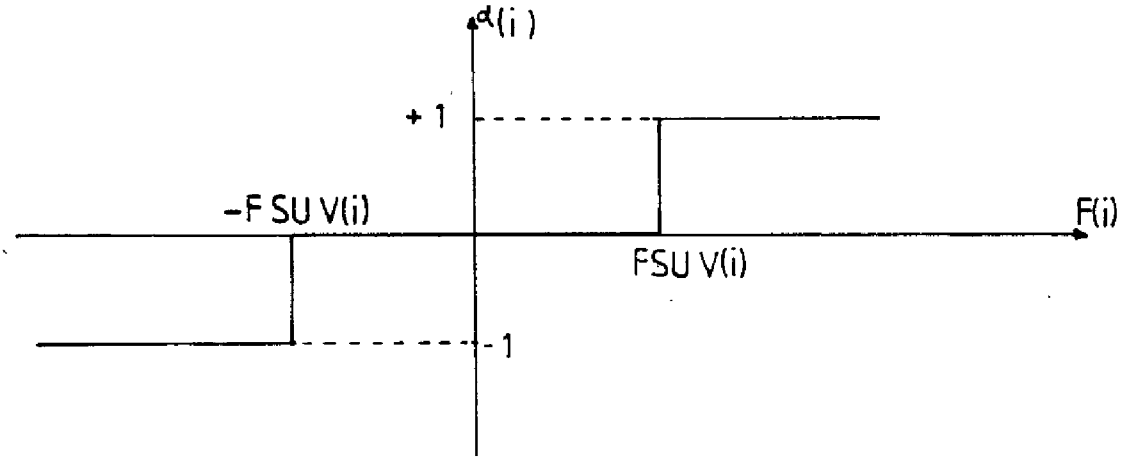


FIGURE III.9 Fonction de surveillance des efforts

$\forall i, \exists i \;/\; F(i) \leq -FSUV(i) \longrightarrow \alpha(i) = -1$

$\forall i \quad \exists i \;/\; F(i) \geq FSUV(i) \longrightarrow \alpha(i) = +1$

$\forall i, \exists i \;/\; F(i) \leq FSUV(i)$

$\forall i \quad \exists i \;/\; F(i) \geq -FSUV(i) \longrightarrow \alpha(i) = 0$

Pratiquement, cette surveillance par axe autorise à faire du "suivi" avec le robot si ce dernier détecte une force ou un moment.
(i) supérieur à FSUV(i) il se translatera, on tournera suivant l'axe représentant le degré de liberté numéro i, d'un incrément $Sd_n(i)$, de manière à annuler l'effort détecté.

III.4.1.1.3. Niveau_3_:_Consignes_force-moment-position

Nous abordons la partie fondamentale de l'asservissement. En effet, travaillant en permanence avec une information sur les réactions des forces au contact, il est indispensable de maitriser la valeur de ces forces et moments produits en les liant aux déplacements.

D'autre part, la consigne en position est nécessaire pour se déplacer dans la direction que nous voulons ou pour orienter le manipulateur angulairement suivant les tâches qu'il doit accomplir.

Ainsi, les deux notions, consignes efforts et position doivent toujours être prises en compte simultanément pour le calcul du déplacement à effectuer.

Contrairement aux deux fonctions précédentes, l'analyse faite nous fournira deux coefficients :

- asservissement en force $\alpha_F(i)$
- asservissement en position $\alpha_p(i)$

Le coefficient résultant $\alpha(i)$ sera une fonction des deux précédent s.

$$\alpha(i) = f(\alpha_F(i), \alpha_p(i))$$

Nous explicitons cette fonction un peu plus loin. i représente toujours le degré de liberté considéré.

A) Asservissement des forces

L'asservissement de force consiste en la génération de déplacements du bras du robot, visant à produire des efforts de réaction égaux ou proches, à un certain pourcentage près, de valeurs imposées.

Ceci revient à rendre minimum, l'erreur égale à la différence entre la force mesurée et la force de consigne.

Le vecteur représentant les forces et moments de consigne est noté F^* (cf III.4.1.1.).

Le seuil tolérable d'erreur sur les forces et moments de consigne est noté $\delta F^*(i)$

$$\delta F^* = (\delta F_X^*, \delta F_Y^*, \delta F_Z^*, \delta M_X^*, \delta M_Y^*, \delta M_Z^*)$$

Nous définissons une zone d'approche de la consigne en donnant un facteur de pondération, qui diminuera l'incrément de déplacement si nous nous rapprochons de la consigne ou qui l'augmentera si nous nous en éloignons.

Le facteur de pondération est noté K_f ; il représente la pente de la droite de la figure III.9 ou la tangente de l'angle .

$$K_f(i) = \frac{1}{FL(i) - \delta F^*(i)} \qquad i \text{ variant de 1 à 6}$$

Le coefficient $K_f(i)$ est initialisé en début de programme mais l'utilisateur peut le modifier s'il le désire.

.Effet sur la stabilité de l'asservissement

Le coefficient K_f joue un rôle prépondérant dans la stabilité de l'asservissement. En effet, suivant la rigidité du poignet utilisé ou des objets en contact, un déplacement incrémental δx aura des conséquences fort différentes sur la variation des forces de contact. Si nous ne connaissons pas les matrices d'élasticité ou de rigidité, relatives à chaque montage, nous risquons d'entrer dans des zones instables où il y aura apparition de phénomènes dits de "pompage". C'est pourquoi nous avons tenu à avoir un réglage possible par l'utilisateur de ce coefficient K_f, qui rend compte de l'élasticité du système. De plus, l'adjonction d'une erreur tolérable sur la force de consigne facilite la réalisation de la consigne.

Cette approche permet de résoudre ponctuellement le problème de la stabilité de notre asservissement. Nous pensons toutefois qu'une étude plus approfondie doit être menée pour résoudre ce problème dans le cas général. Une solution serait l'introduction de matrices de rigidité.

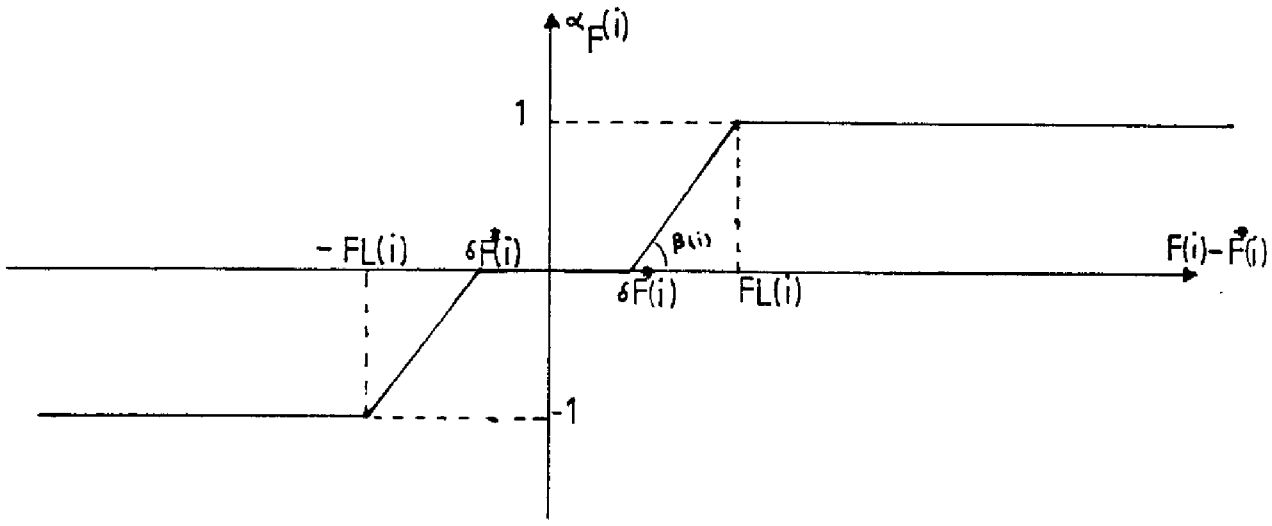


FIGURE III.9 Fonction réalisant l'asservissement en force

$$-\delta F^*(i) \leqslant (F(i) - F^*(i)) \leqslant \delta F^*(i) \longrightarrow \alpha_F(i) = 0$$

$$F(i) - F^*(i) \geqslant FL(i) \longrightarrow \alpha_F(i) = +1$$

$$F(i) - F^*(i) \leqslant -FL(i) \longrightarrow \alpha_F(i) = -1$$

$$-FL(i) \leqslant (F(i) - F^*(i)) \leqslant -\delta F^*(i) \longrightarrow \alpha_F(i) = K_F(i)(F(i)-F^*(i))+K_F(i).\delta F^*(i)$$

$$\delta F^*(i) \leqslant (F(i) - F^*(i)) \leqslant \delta F^*(i) \longrightarrow \alpha_F(i) = K_F(i)(F(i)-F^*(i))-K_F(i).\delta F^*(i)$$

B) Asservissement de position

Le principe de cet asservissement est d'amener le manipulateur dans une position donnée, à partir de sa position initiale.

La consigne en position que le bras manipulateur devra atteindre est représentée par un repère orthonormé, RCCO, exprimé par rapport à un repère de référence.

Nous rappelons que le référentiel dans lequel se fait l'asservissement (calculs forces, déplacements) est noté RCO.

La consigne sera donc obtenue en exprimant le repère RCCO par rapport au repère RCO à chaque échantillonnage. Nous en déduisons alors la translation (T_X, T_Y, T_Z) et la rotation (R_X, R_Y, R_Z) que le robot doit effectuer. C'est sa consigne. Après exécution de chaque déplacement, la consigne est recalculée en fonction de l'actualisation du repère lié au bras manipulateur.

1) Description fonctionnelle

Le repère RCO est défini en début de manipulation par l'utilisateur. Le signe de la consigne à atteindre pour chaque degré de liberté représentera le sens du déplacement sur ces mêmes degrés de liberté (i).

Le calcul de RCCO par rapport à RCO est fait automatiquement par la fonction asservissement qui définit alors la consigne par un vecteur à six composantes noté

$$X^* = (x^*, y^*, z^*, \theta^*, \psi^*, \phi^*)^T$$

Le coefficient $\alpha_p(i)$ traduit l'amplitude de l'incrément de déplacement sur l'axe i.

La courbe ci-dessous définit $\alpha_p(i) = f(X^*(i))$

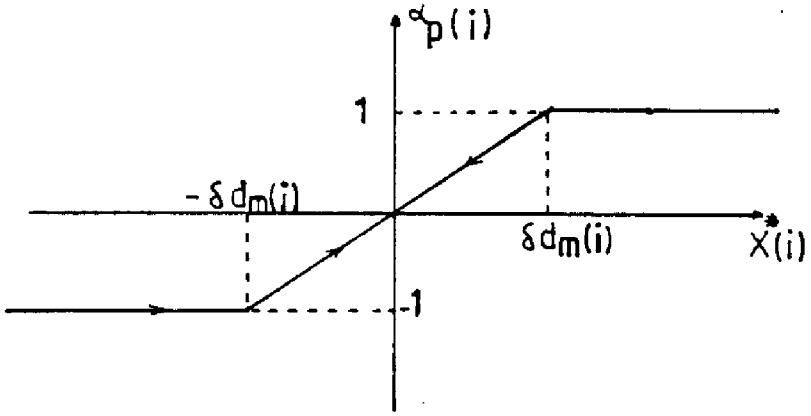


FIGURE III.10 Fonction réalisant l'asservissement en position

si $X^*(i) \geq \delta d_m(i)$ $\quad \alpha_p(i) = 1$

$X^*(i) \leq -\delta d_m(i)$ $\quad \alpha_p(i) = -1$

$-\delta d_m(i) \leq X^*(i) \leq \delta d_m(i)$ $\quad \alpha_p(i) = \frac{1}{\delta dm(i)} \cdot X^*(i)$

Tant que la consigne à atteindre est supérieure en valeur absolue à l'incrément de déplacement $\delta d_m(i)$, nous générons un déplacement égal à la totalité de l'incrément.

Dans le cas contraire où la consigne est inférieure en valeur absolue à cet incrément, nous nous rendons directement au point désiré, avec l'orientation voulue, en se déplaçant de la valeur de cette consigne $X^*(i)$.

C) Analyse des priorités-forces ou positions

Nous avons vu au début de cette étude, que nous pouvions décomposer cet "asservissement" en plusieurs niveaux. Le niveau le plus prioritaire étant le niveau 1 puis le niveau 2 et ainsi de suite. Un niveau est activé que si le précédent ne l'a pas été.

Ainsi, si nous avons $|FSUV(i)| < |F^*(i)|$ nous n'examinerons pas pour l'axe i le niveau 3.

Dans le cas où les conditions sont remplies pour rentrer dans la boucle de calcul correspondant au niveau 3, nous devons examiner la "dualité" entre l'asservissement de force et de position.

1) Conflit entre force et position

Le sens du déplacement que nous calculons est déterminé par le signe de la consigne de force $F^*(i)$ et le signe de la consigne de position $X^*(i)$.

L'utilisateur doit alors prendre la précaution, au moment de la définition de son problème de garder la compatibilité en ne définissant pas deux consignes de signe opposé. Néanmoins, si cela se produit le programme génère un signal d'erreur (état) et demande une redéfinition.

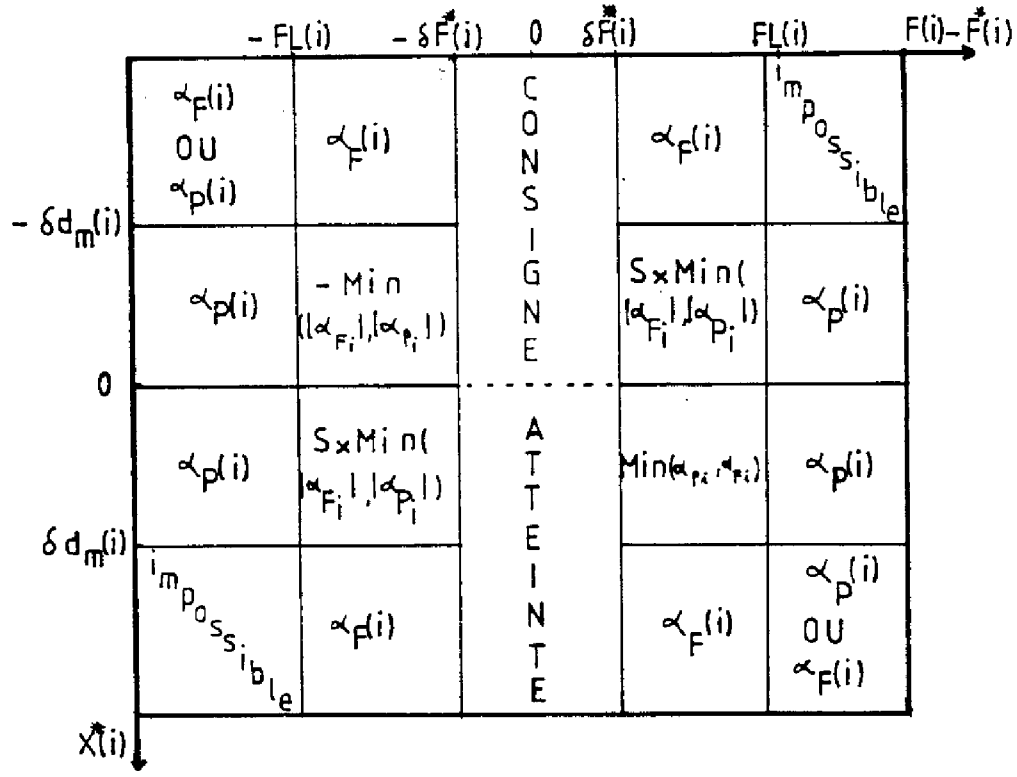
Dans les autres cas, le déplacement généré par les deux types d'asservissement étant de même signe, nous faisons le choix de prendre le minimum des deux déplacements calculés, de manière à satisfaire la consigne la plus proche.

La priorité est donc donnée à la consigne atteinte la première, sans tenir compte de sa qualité.

Ceci implique que l'utilisateur devra tenir compte de ce choix, au moment de la définition des variables (de l'asservissement) afin de privilégier la position ou bien la force.

Nous pouvons schématiser ce conflit entre force et position par le tableau suivant.

S = signe de la variable α_{p_i} ou α_{F_i} la plus petite en valeur absolue



Le déplacement maximum étant égal à $Sd_m(i)$, nous pouvons transformer le tableau de la manière suivante en prenant systématiquement le minimum de la valeur absolue de α_{P_i} et α_{F_i}. Le coefficient S a toujours la même signification.

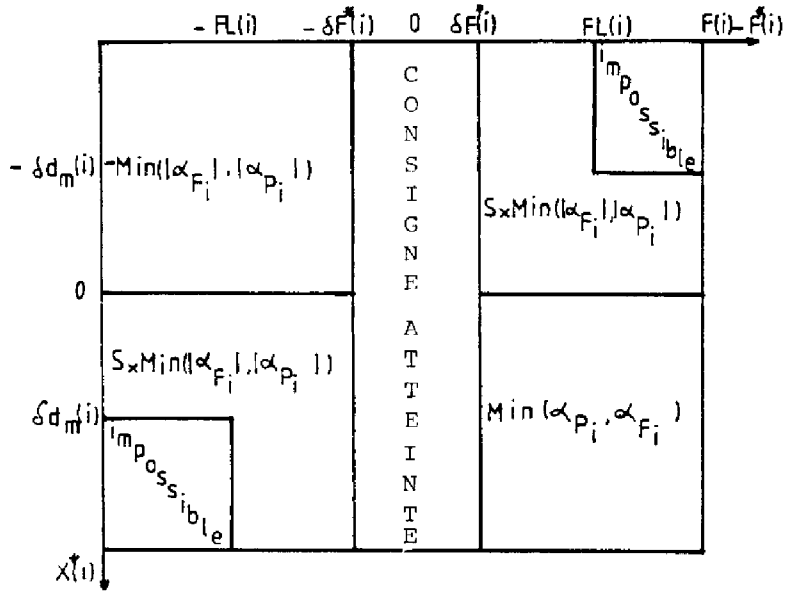


FIGURE III.11 *Tableau rendant compte de la dualité entre les asservissements de force et de position*

2) Interconnexion entre les asservissements par axe

Nous avons pensé, dans le cas de manipulation d'objets nécessitant une mise en contrainte permanente de l'organe terminal du robot, créer un mode de fonctionnement supplémentaire. Celui-ci entraîne une interdépendance entre les axes et tend à privilégier la consigne de force sur un axe au détriment des consignes de position sur les autres axes.

Nous associons une variable entière à chaque degré de liberté. Cette variable est dite active quand elle est égale à deux. Son activation doit entraîner le blocage du déplacement généré par l'asservissement sur l'axe correspondant, et ceci tant que les consignes de force sur les autres axes ne seront pas satisfaites à un certain pourcentage pris.

Exemple

Prenons le cas d'un asservissement en force-position sur l'axe des X et en position sur l'axe des Y. Les forces et moments sur les autres axes sont surveillés afin d'éviter une détérioration physique du matériel.

axe des X = asservissement en force
axe Y = asservissement de position
axe Z = surveillance de la force
rotation autour Z = surveillance du moment

Les deux axes sur lesquels sont générés des déplacements, sont les axes X et Y. En activant la variable correspondant au degré de liberté suivant l'axe Y, nous assujettirons le robot à produire la force de consigne demandée en X avant tout déplacement en Y. Le robot pourra alors se mouvoir tout en maintenant la pièce ou l'outil saisie par son organe terminal, en contact avec le plan de travail.

III.4.1.1.5. Etat de la fonction asservissement

Nous avons pensé ajouter à la fonction asservissement un indicateur d'état du système, noté I ETAT. C'est une variable, prenant certaines valeurs entières correspondant à un mode de fonctionnement de l'asservissement.

Nous donnons ci-dessous le code utilisé.

.I ETAT = 1	Fonctionnement normal, la ou les consignes ne sont pas atteintes.
.I ETAT = 2	Pour chacun des axes, une au moins des consignes (force ou position) est satisfaite. Plus de mouvements.
.I ETAT = 3	Arrêt sur dépassement du module de la force FFSUV. (envoi message opérateur)
.I ETAT = 4	Arrêt sur dépassement du module du moment FMSUV.
.I ETAT = 5	Mouvement impossible. Consignes mal définies.

L'écriture de cet indicateur, après chaque passage dans la fonction asservissement, permettra à l'utilisateur de vérifier si le fonctionnement est conforme au but fixé.

III.5. CONCLUSION

Nous avons fait le choix de ce type d'asservissement, pour avoir une grande facilité d'utilisation et d'adaptation aux différentes tâches que nous voulons réaliser.

La source essentielle d'information est bien sûr la mesure des efforts, qui nous permet après traitement de faire l'analyse du contact et de générer les mouvements du manipulateur.

L'asservissement tel qu'il est conçu nous autorisera à mettre en contact une pièce A, tenue par le manipulateur, avec un plan B. Nous pourrons aussi pour un même point de contact produire des petits déplacements qui nous donneront l'information suffisante pour la détermination des coordonnées de ce point.

Nous envisageons aussi de découvrir les contours d'une pièce, de mettre un plan en contact avec un autre plan, de décrire les degrés de liberté de la pièce extrémale du robot.

Ces différentes applications (voir chapitre VI) prouvent la versatilité de l'asservissement que nous avons développé. La mise en oeuvre d'une nouvelle manipulation ne se résumant qu'en la redéfinition appropriée des variables de cette fonction.

C H A P I T R E IV

LE MATÉRIEL MIS EN OEUVRE

IV.1. INTRODUCTION

L'application pratique des résultats obtenus au chapitre précédent nécessite la présence d'un support physique, électronique et mécanique.

Ce chapitre s'articulera donc autour des différents organes matériels à notre disposition et de leur intégration en vue de réaliser diverses expérimentations.

Nous pouvons différencier trois grandes parties :

- La description du manipulateur, sa conception, ses degrés de liberté, l'électronique de commande qui lui est associée. En particulier, le microprocesseur de commande des moteurs.
- L'organe terminal du robot, ou poignet déformable qui nous fournit l'information sur la mesure des efforts. Le traitement de cette information par un microprocesseur spécialisé.
- Le mini calculateur gérant les échanges avec les microprocesseurs, la console graphique et dialoguant avec l'opérateur. C'est sur cet ordinateur que sont implantés les programmes de base pouvant nous donner la souplesse suffisante pour les expériences envisagées.

Le schéma général de l'organisation de ces différents éléments est représenté sur la figure IV.1.

IV.2. LE MANIPULATEUR ET SA COMMANDE

Nous disposons, comme support physique expérimental, d'un bras manipulateur, construit par la société JAZ et dont la commande a été développée au LAAS[GIR_b 79].

Ayant été conçu pour manipuler des pièces situées dans un vrac planaire, il n'est doté que de quatre degrés de liberté.

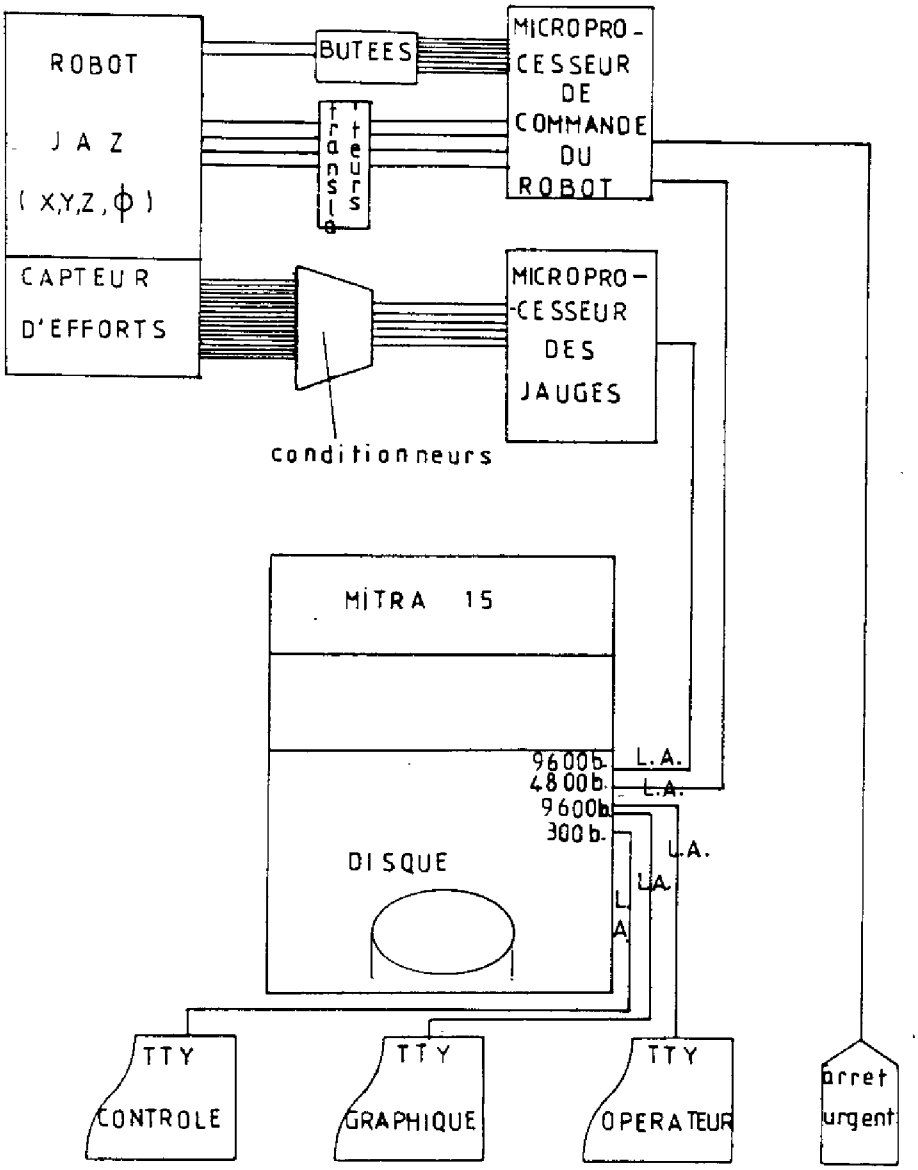


FIGURE IV.1 *Configuration générale du dispositif expérimental*

IV.2.1. Architecture du bras manipulateur

Pour diverses raisons -grande précision, répétabilité de positions, grande rapidité, efforts importants à fournir- les constructeurs ont retenu des actionneurs hydrauliques linéaires, asservis en position par moteur pas à pas.

IV.2.1.1. Vérins hydrauliques linéaires asservis en position

Le vérin comporte un piston, solidaire de la tige. Il est soumis de part et d'autre à la pression du fluide. Cette tige creuse est munie d'un écrou recevant une vis solidaire de la valve située dans le fond du vérin. L'ensemble vis-valve est entraîné en rotation par le moteur pas à pas au moyen d'un accouplement parfait sans jeu. La figure IV.2 représente le principe de cet actionneur.

A tout angle de rotation $d\theta$ de la vis correspond un déplacement linéaire dL de la tige du piston déterminé simplement par le pas de vis de commande.

Le déplacement linéaire de la tige est alors asservi à la position angulaire de la vis de commande.

L'avantage de ces vérins hydrauliques est la précision de positionnement $\pm$ 0.01 mm, la répétabilité, la grande rigidité pour les déplacements rapides, la fiabilité et la séparation des fonctions ; puissance, huile fournie par la centrale hydraulique,commande par moteur pas à pas.

IV.2.1.2. Moteurs pas à pas

Ils ont été choisis pour assurer des déplacements rapides aux vérins. Voulant atteindre un déplacement de 200 mm/s, les moteurs les plus courants dans le commerce ayant 200 pas par tour, la fréquence maximale de fonctionnement est de 10 000 Hertz avec une résolution de 0.02 mm par pas.

Ceci a conduit les constructeurs à utiliser des moteurs pas à pas à structure hybride qui combine avantageusement les caractéristiques des moteurs à aimant permanent et à rélectance variable.

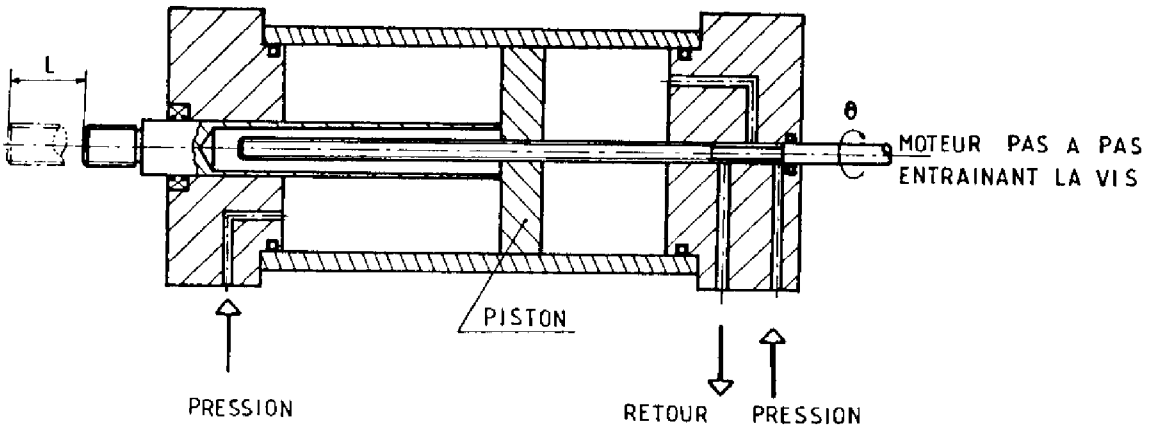


Fig IV.2 VUE EN COUPE D'UN ACTIONNEUR DU ROBOT JAZ

Cette structure permet d'assurer un verrouillage magnétique du moteur en position de repos, même en l'absence de courant dans les bobinages. Ceci assure un maintien de la position de la tige des vérins, même en cas de coupure accidentelle de la tension secteur.

Ces moteurs sont alimentés par une électronique de puissance adaptée.

IV.2.1.3. Structure du robot

Les constructeurs ont choisi une structure plaçant les vérins le plus près possible du centre de rotation du corps du robot, afin de rejeter les fréquences propres du système vers les valeurs élevées.

Pour réduire au minimum l'inertie des deux vérins assurant la commande du bras, ils ont été placés verticalement. Le troisième vérin commandant la rotation du corps du robot est disposé horizontalement au-dessus du bâti.

Cette configuration nous permet de disposer de trois degrés de liberté, que nous pouvons définir comme trois translations selon les axes d'un repère orthonormé classique (l'axe des Z est l'axe vertical).

Le dernier degré de liberté est fourni par un moteur pas à pas, entraînant par courroie la rotation dans le plan horizontal,du poignet fixé à l'extrémité du bras. Notons qu'un système de parallélogramme maintient le poignet vertical quelle que soit la position du bras.

La précision obtenue en bout de bras avec une telle structure est de $\pm$ 0.06 mm. Sa capacité de préhension est de 30 DaN.

La représentation du robot est fournie par la figure IV.3.

IV.2.2. L'électronique de commande

IV.2.2.1. Les translateurs

Ces organes correspondent à l'électronique de puissance qui alimente les moteurs pas à pas.

Ils offrent la possibilité de travailler en demi-pas. Ainsi, un moteur à deux cents pas par tour passe à quatre cents pas, avec une résolution de 0.01 mm par pas. Nénamoins, nous travaillerons avec une fréquence maximale de 10 000 Hz d'où une résolution de 0.02 mm par pas.

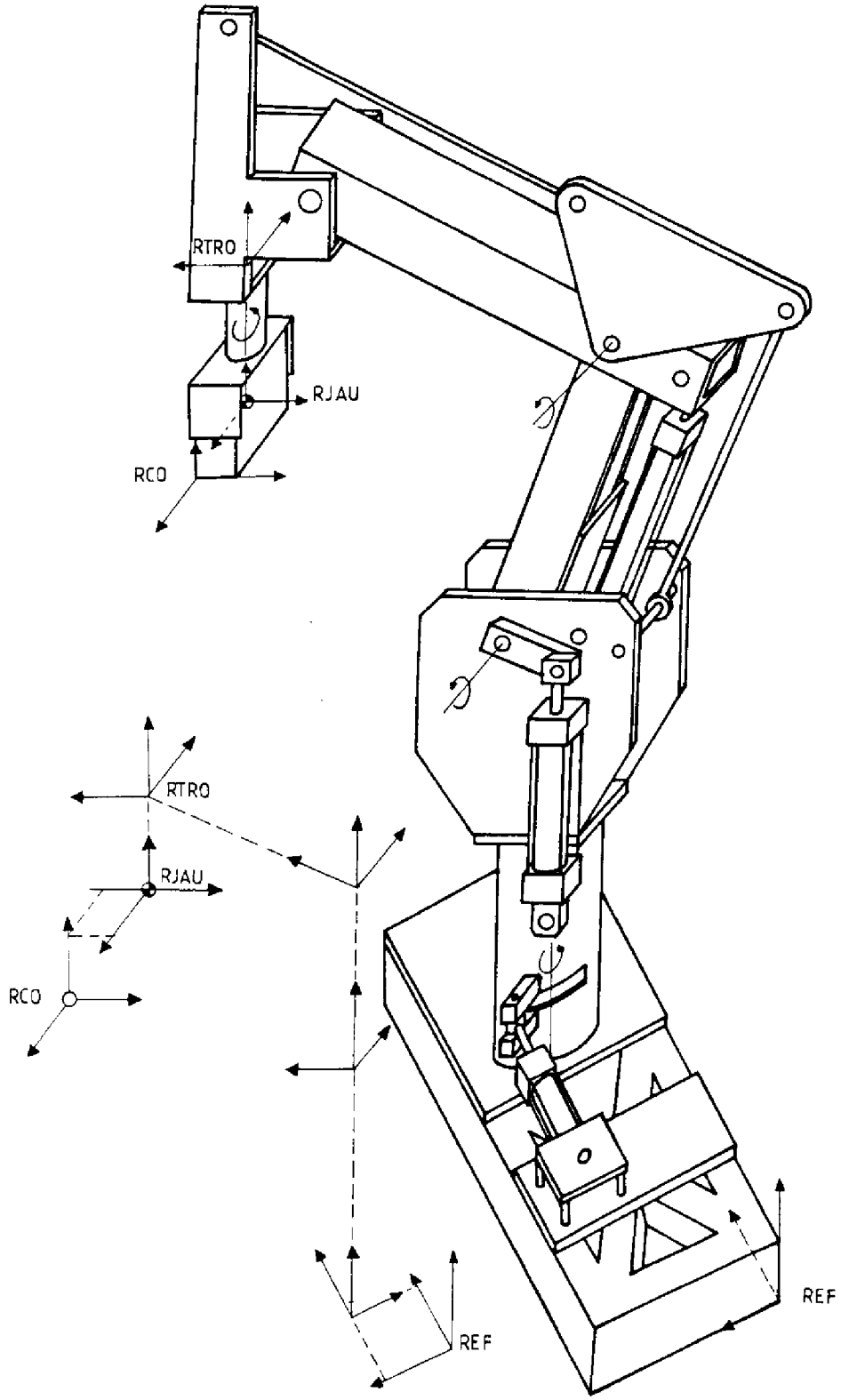


FIGURE IV.3 *Représentation du bras manipulateur*

Le branchement des bobinages est en parallèle pour permettre le fonctionnement jusqu'à la fréquence de 20 000 Hz. C'est une alimentation bipolaire à deux niveaux, de coût modéré.

IV.2.2.2. Les butées de fin de course

Par définition, la proprioception est la perception de l'information relative à l'état du robot lui-même indépendamment de son environnement. Il est important de noter l'absence de proprioception informant sur la position réelle du robot.

Les seules informations proprioceptives existantes sont celles fournies par les capteurs de proximité utilisés en détecteurs de fin de course. Leur rôle est cependant très important car ils présentent les deux avantages suivants :

a) Précision

Ils permettent un positionnement du robot, aussi reproductible que possible, dans sa position dite de repos. Cette position sert alors de référence absolue, et nous pouvons connaître la position exacte du robot par rapport à la référence, en fonction des mouvements générés ;

b) Sécurité

Ils protègent l'équipement mécanique et électromécanique contre toutes détériorations provoquées par des erreurs de commande tendant à entraîner le robot dans des positions pour lesquelles il n'est pas conçu.

IV.2.2.3. Le microprocesseur de commande

C'est un microprocesseur INTEL 8085. Il a été développé au LAAS par A. GIRAUD pour assurer la commande simultanée de quatre moteurs pas à pas en mode coordonné ou cinématique.

Il est chargé de fournir les impulsions aux translateurs afin de générer les déplacements pour chaque degré de liberté.

Il reçoit l'information en provenance des butées de fin de course.

Il jouera pour nous le rôle de capteur proprioceptif ; c'est-à-dire qu'en l'interrogeant, nous pouvons connaître à tous moments la position et l'orientation de la partie terminale du bras, par rapport à une origine définie en butée.

La commande se fait dans l'espace des pas moteurs ; le microprocesseur reçoit de la part du minicalculateur le nombre de pas à générer pour chaque moteur, ainsi qu'un code opératoire qui lui permet de sélectionner le mode de fonctionnement choisi.

Le logiciel réalisé sur ce micro sera détaillé au chapitre V.

Notons qu'un interrupteur, arrêt d'urgence, relié directement au microprocesseur de commande, permet à l'opérateur s'il détecte un mauvais fonctionnement, d'arrêter le mouvement du robot. Des interrupteurs de sécurité placés sur le poignet (cf IV.3.2.) peuvent aussi déclencher cet arrêt urgent.

Ce microprocesseur est relié au mini calculateur Mitra 15 par une ligne asynchrone. Nous pouvons travailler à des vitesses différentes, mais dans le cas général, nous choisissons la vitesse de transmission la plus rapide. Actuellement, l'échange des données entre le microprocesseur et le minicalculateur s'effectue à 4800 bauds ; une vitesse supérieure (9600) entrainerait une perte d'information aléatoire dans certains modes de fonctionnement.

IV.3. LE POIGNET-CAPTEUR D'EFFORT

Conçu et réalisé au LAAS par Ghazi CHAOUI, le poignet est l'organe terminal du bras manipulateur ; c'est à son extrémité que sera fixée ou saisie la pièce.

Il réalise deux fonctions simultanées :

- la mesure des forces et des moments qui lui sont appliqués ; il s'identifie alors à un capteur d'effort ;
- la "compliance", anglicisme désignant la faculté du poignet à se déformer de manière à minimiser les efforts qui prennent naissance quand une force est exercée à son extrémité.

IV.3.1. Généralités

Plusieurs poignets ont été développés, leur utilisation est fonction des tâches à accomplir. En effet, les problèmes d'insertion compliqués requièrent une déformabilité du système, assez importante, afin d'éviter les coincements et blocages.

Par contre, certains types d'insertion, en force, ne demandent pas de telles qualités.

Pour les problèmes de contact pur et de déplacement du bras asservi par les efforts, c'est la partie sensibilité aux efforts extérieurs qui doit être privilégiée.

La première structure mise en place possédait une bonne déformabilité ("compliance") mais donnait lieu à une hystéresis dans la mesure des efforts et plus précisément des moments.

Dans les manipulations envisagées, comme la reconnaissance du point de contact, nous devions tenir compte de cette déformation relativement importante, pour que les résultats n'en soient pas affectés. Ceci bien sûr, entrainait un alourdissement des calculs. De même, l'hystéresis sur la mesure des moments introduisait des erreurs non négligeables ; cette hystéresis était due au fait que la couronne inférieure du poignet portant les jauges de contraintes sensibles aux moments, ne revenait pas exactement au même point d'équilibre après application d'un effort à l'extrémité du poignet, en raison de la présence de frottements secs. Le poignet construit avait donc plusieurs points d'équilibre, d'où la raison de l'hystéresis constatée.

Pour combler ce problème, un second modèle a été développé cherchant à découpler la déformabilité de la mesure des efforts.

La figure IV.7 rend compte de la solution adoptée pour le capteur d'effort. Les jauges de contrainte résistives sont collées sur des poutres composant un "U". Le capteur est formé de trois de ces "U" disposé à 120 degré les uns des autres.

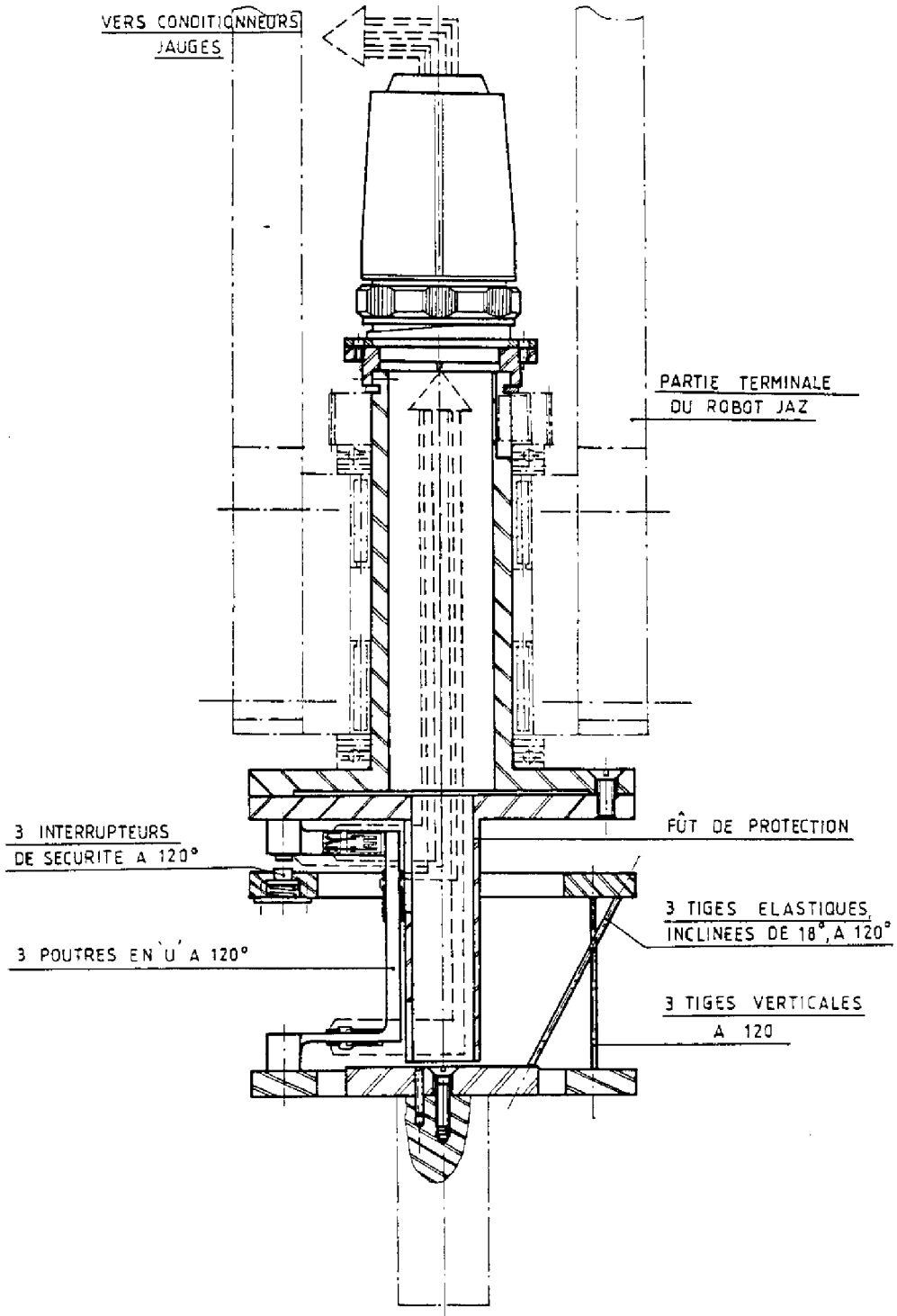


Fig IV.4 CAPTEUR D'EFFORT TRIDIMENTIONNEL + POIGNET COMPLIANT PASSIF SUR ROBOT JAZ

En ce qui concerne la partie "compliante" (Figure IV.5), elle est constituée de trois barres verticales disposées à 120 degrés, assurant la déformation en translation et de trois barres verticales positionnées de la même manière que les précédentes permettant les déformations en rotation. La figure IV.4 donne une vue d'ensemble du poignet capteur d'effort.

IV.3.2. La déformabilité ou "compliance"

La compliance ou déformabilité du poignet utilisé est passive. C'est-à-dire qu'au cours d'une manipulation, quand l'organe terminal du poignet vient en contact avec le milieu extérieur, la structure du poignet aura tendance à se déformer spontanément afin de minimiser les efforts qui y ont pris naissance. Cette déformation doit rester en-dessous du seuil de déformabilité permanente, limite élastique du capteur. A cet effet, des sécurités (interrupteurs) sont prévues pour déclencher l'arrêt urgent du robot (cf Figure IV.4). Néanmoins, nous pouvons avec un poignet passif, réaliser une compliance active. Les déplacements de l'organe terminal sont générés par la commande des degrés de liberté du manipulateur après analyse des efforts mesurés par le poignet, capteur de forces.

La modélisation de la structure du poignet, faite en langage A.P.L., a permis de caractériser sa déformabilité.

Notre propos n'est pas ici de décrire les méthodes employées pour parvenir à la réalisation de ce poignet CHA 81 , nous nous bornerons à donner les résultats obtenus.

En effet, la déformation de la structure, dans un repère de référence, sous l'action d'effort extérieurs, peut être caractérisée de la manière suivante :

Nous noterons : $F = (F_X, F_Y, F_Z, M_X, M_Y, M_Z)^T$ les efforts appliqués,

$U = (X, Y, Z, \Theta, \Psi, \Phi)^T$ les déplacements.

Les unités utilisées sont le millimètre, le decanewton et le degré. Le vecteur U est obtenu en multipliant la matrice de rigidité K par le vecteur des efforts F.

$$U = K \cdot F$$

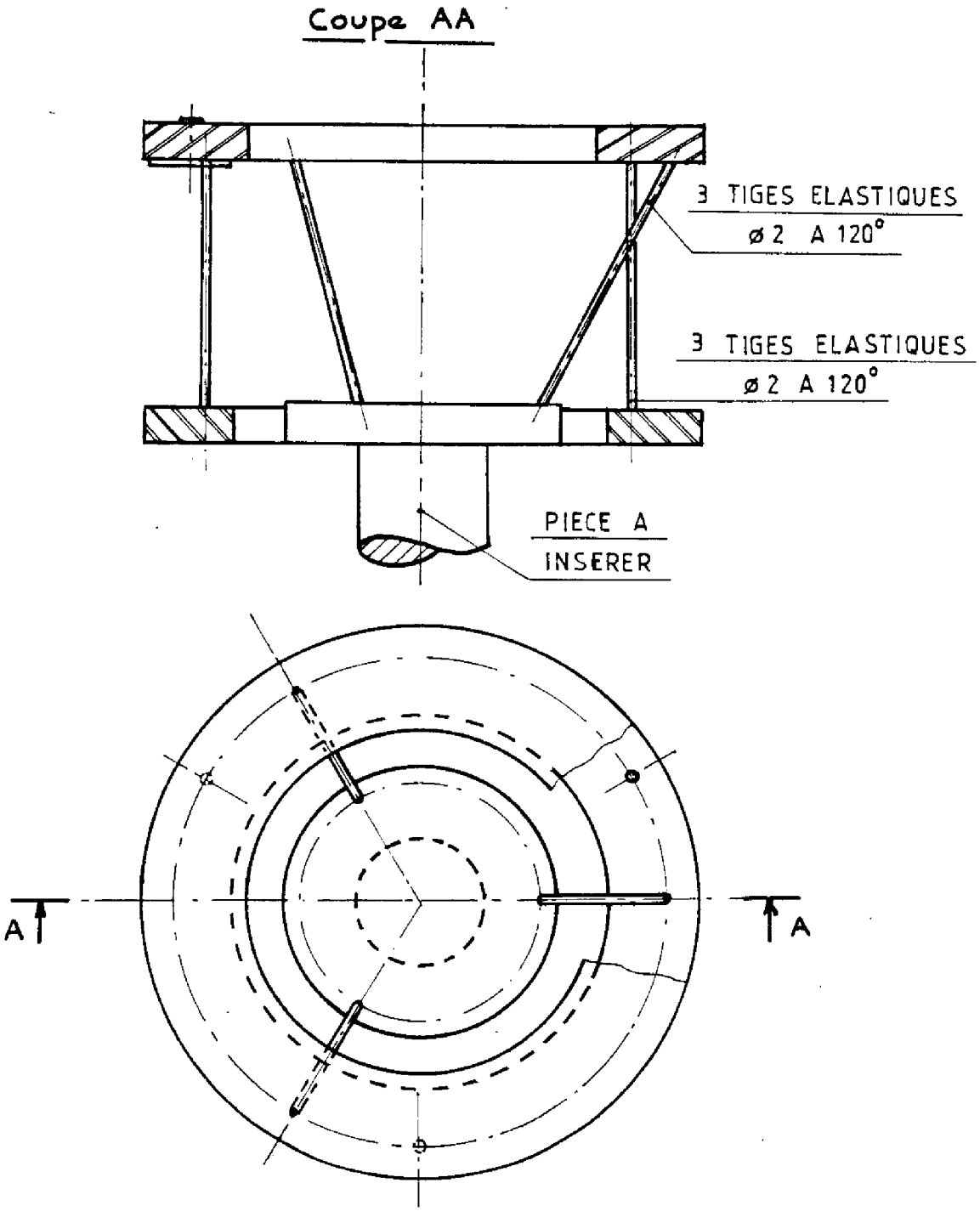


Fig IV.5 STRUCTURE COMPLIANTE PASSIVE

K est une matrice six lignes, six colonnes, inversible . Nous noterons $E=K^{-1}$ matrice d'élasticité, nous aurons alors

$$F = E \cdot U$$

Nous donnons ci-après les matrices K et E obtenues par la modélisation du poignet, dans un repère situé au bout d'une pige de longueur 75 mm vissé à l'extrémité du poignet.

$$K = \begin{bmatrix} 1.746 & 0 & 0 & 0 & 0.0133 & 0 \\ 0 & 1.746 & 0 & 0.0133 & 0 & 0 \\ 0 & 0 & 0.04 & 0 & 0 & 0 \\ 0 & 0.794 & 0 & 0.0034 & 0 & 0 \\ 0.794 & 0 & 0 & 0 & 0.0034 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0357 \end{bmatrix}$$

$$E = \begin{bmatrix} 2.2646 & 0 & 0 & 0 & 3.7205 & 0 \\ 0 & 2.2646 & 0 & 3.7205 & 0 & 0 \\ 0 & 0 & 25 & 0 & 0 & 0 \\ 0 & 214.06 & 0 & 470.72 & 0 & 0 \\ 214.06 & 0 & 0 & 0 & 470.72 & 0 \\ 0 & 0 & 0 & 0 & 0 & 28.011 \end{bmatrix}$$

La connaissance du torseur de force, nous permettra d'obtenir les déplacements d'un point de la structure, exprimé dans un repère donné. Nous pourrons alors prendre en compte la déformation du poignet dans le calcul des coordonnées du point de contact.

IV.3.3. Le capteur d'effort

IV.3.3.1. Principe

Le principe est de mesurer des micro-déformations ε en certains points de la structure ; la déformation est une fonction linéaire des forces appliquées.

Les types de capteur qui nous permettent de réaliser cette fonction sont des jauges de contrainte résistives. Elles sont de faibles dimensions sensibles et peu coûteuses.

Elles sont disposées dans le sens longitudinal des poutres comme le montre la figure IV.6. Elles ne sont alors sensibles qu'à la flexion de la poutre par rapport à l'axe.

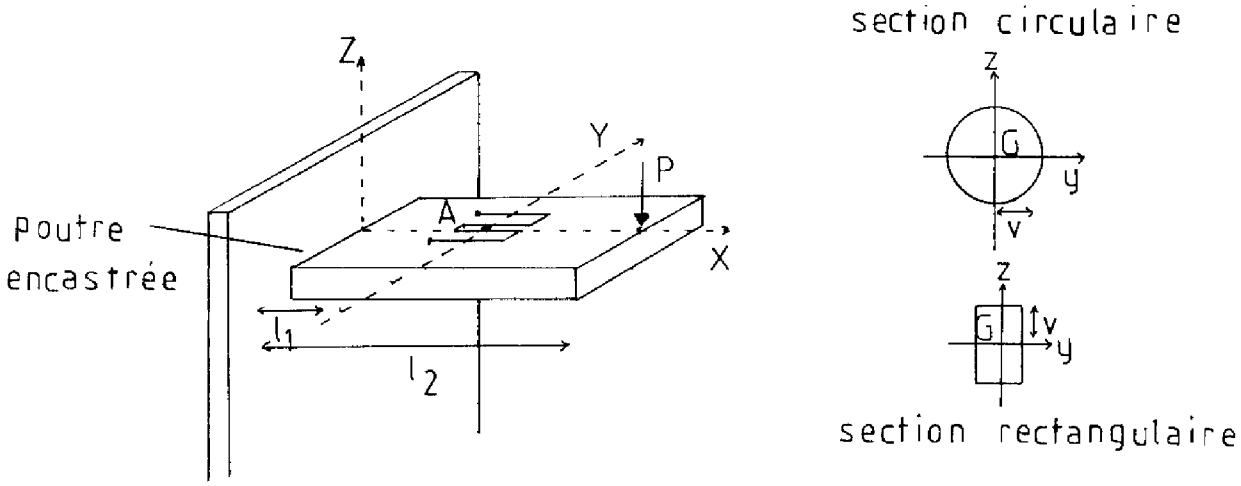


FIGURE IV.6 Jauge travaillant à la flexion

D'après la figure IV.6, pour un effort P appliqué à l'extrémité de la poutre, nous aurons la relation :

$$\sigma = \varepsilon \cdot E = \frac{\Delta L}{L} \cdot E = \frac{m_f \cdot V}{E \cdot I_y} \qquad m_f = P(l_2 - l_1) \qquad (1)$$

V = mi-hauteur de la section de la poutre
E = module de Young du matériau
I_y = moment quadratique par rapport à G_y
m_f = moment de flexion au point A où est collée la jauge
ε = variation relative de l'élongation de la jauge

La relation (1) permet de relier les efforts appliqués aux micro-déformations de la jauge. Nous devons maintenant mesurer ces micro-déformations. En fait, quand un effort est appliqué à l'extrémité du capteur, il est transmis à la structure sur laquelle sont collées les jauges, qui se déforment. Leur résistance varie alors, en raison de l'allongement ou de la contraction des surfaces des poutres où elles sont placées. C'est cette variation de résistance qui contient l'information sur les micro-déformations.

Nous devons donc mesurer la variation de résistance des jauges, puis nous obtiendrons les micro-déformations par la relation (2)

$$\frac{\Delta R}{R} = K.\varepsilon = K\frac{\Delta L}{L} \qquad (2)$$

K = facteur de jauge, il caractérise la sensibilité du phénomène d'élastorésistance (K=2)

$\frac{\Delta R}{R}$ = variation relative de la résistance de la jauge

IV.3.3.2. Réalisation

Le capteur d'effort décrit par la figure IV.7 recevra au total neuf paires de jauges notées - (J_{i_1}, J_{i_2}) i variant de 1 à 6
- (J_{i_3}, J_{i_4}) i variant de 1 à 3

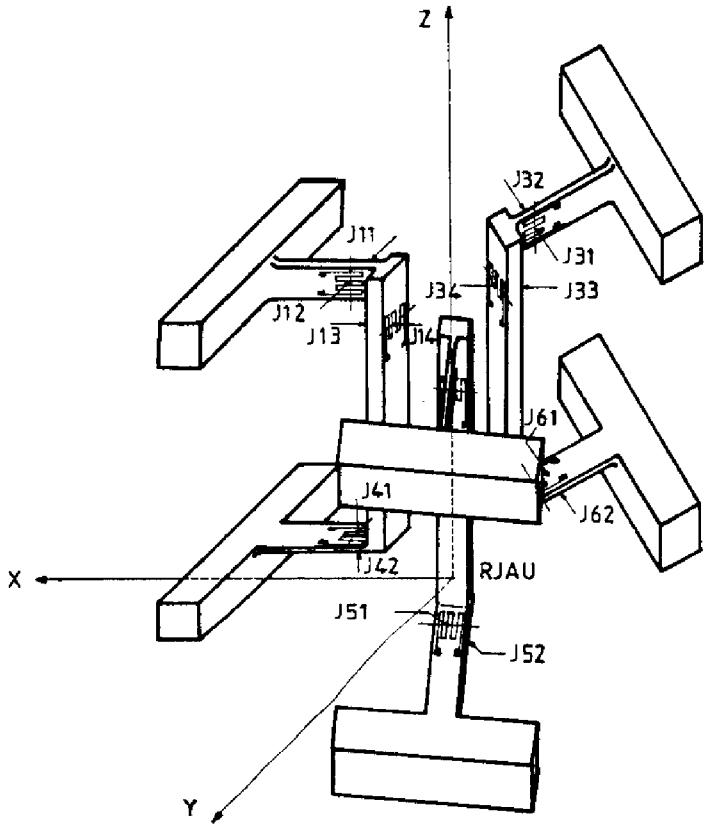


FIGURE IV.7 Schéma décrivant la disposition des jauges sur le capteur d'effort

Les jauges $(J_{i1},J_{i2})_{i=1 \text{ à } 6}$, $(J_{i3},J_{i4})_{i=1 \text{ à } 3}$, sont collées en vis à vis dans le sens des poutres de flexions (Figure IV.8).

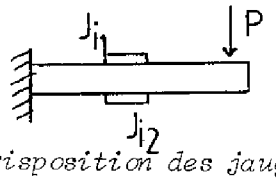
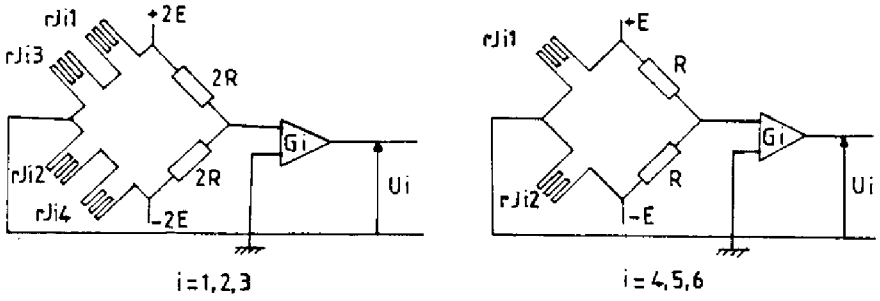


FIGURE IV.8 Disposition des jauges sur une poutre

Les jauges sont collées par paires pour augmenter la sensibilité d'une part, et pour compenser les dérives dues à la température, d'autre part.

Pour mesurer les variations de résistance des jauges quand elles sont sollicitées, elles seront associées par paires à des résistances pures pour former un pont. Le montage adopté, décrit par la figure IV.9, permettra d'enregistrer la variation de résistance des jauges quand elles travailleront en flexion, mais restera insensible à une sollicitation en traction ou compression.



E = 4 volts

R = rJ_i = 12 ohms (i = 1 à 6

G_i = 2 000 ; i = 1 à 3

G_i = 1 000 ; i = 4 à 6

FIGURE IV.9 Montage électrique des jauges de contrainte

Nous avons utilisé pour réaliser ces fonctions des conditionneurs.

.Les conditionneurs des jauges de contrainte résistives

Les six conditionneurs sont des cartes 2B31J de chez Analog Devices. Ils permettent pour chaque pont de jauge d'avoir une amplification réglable de 1 à 2000.

En fait, nous avons deux réglages de gain en cascade. Le premier est fixe ; il est déterminé par la valeur d'une résistance satilisée. Le second est variable ; il permet le réglage à $\pm$20% de la valeur du gain précédent.

Nous aurons en sortie de chaque conditionneur une tension exprimée en millivolt, qui rendra compte du déséquilibre du pont. A l'état d'équilibre quand les jauges ne sont pas sollicitées (pas d'efforts appliqués sur la structure) nous devons avoir une tension nulle en sortie des conditionneurs. Un réglage est prévu à cet effet pour parfaire cet équilibrage de zéro.

La figure IV.10 représente un de ces conditionneurs.

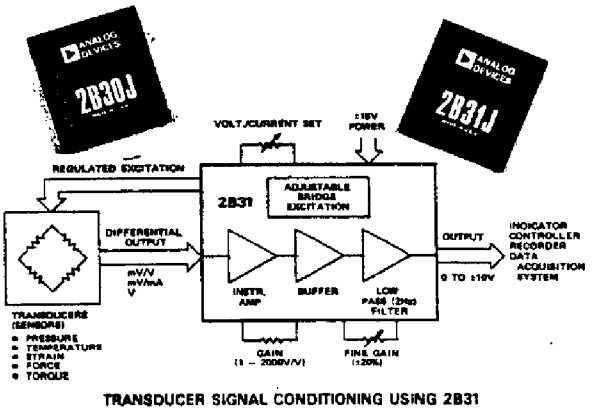


FIGURE IV.10 Schéma d'un conditionneur Jauge

IV.3.3.3. Exploitation des mesures

La modélisation du capteur a permis d'obtenir une matrice (6x6) qui multipliée par le vecteur des micro-déformations, donnera le vecteur des efforts dans un repère, noté RJAU, (cf chapitre II.3), situé dans le plan de contact.

Nous avons dû, par la suite, appliquer une méthode d'étalonnage automatique du capteur, utilisant les propriétés du contact ponctuel (cf chapitre III) de manière à calibrer correctement la matrice précédente.

La matrice définitive, notée SIGTO, obtenue après étalonnage, nous donnera à partir des tensions analogiques issues des conditionneurs, le vecteur des efforts, exprimé dans le repère RJAU. Ce repère est identique au repère global ayant servi à la modélisation du capteur à une translation en $\bar{Z}$ près (cf Figure IV.7).

Matrice SIGTO

0.0056914	-0.0036041	-0.0020841	-0.0041012	0.0024101	0.00108
0.0007441	0.0044754	-0.0054217	-0.0008079	-0.0031194	0.003910
-0.0000373	0.0000432	-0.0000693	0.011264	0.010707	0.010373
-0.058997	-0.3744	0.4482	0.068766	0.53013	-0.55216
0.49495	-0.3744	-0.19146	-0.6786	0.3582	0.26617
0.040661	0.036819	0.04732	-0.022254	-0.025722	-0.033008

TOSIG $\simeq$ SIGTO $^{-1}$

240.93	-0.880	20.786	-0.2404	-1.3726	7.2644
-141.27	230.45	18.126	1.5427	0.7531	8.644
-135.66	-202.08	22.026	-1.296	1.0182	8.4324
196.36	-34.691	28.367	-0.2854	-2.1369	-1.2271
-98.91	180.63	29.537	2.1147	1.0655	0.6648
-110.58	-151.09	35.259	-1.8887	1.2193	0.6928

Nous pouvons représenter la matrice TOSIG de la manière suivante :

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{bmatrix} = \begin{bmatrix} \delta_{11} & \delta_{12} & c_{13} & c_{14} & c_{15} & \delta_{16} \\ \delta_{21} & \delta_{22} & c_{23} & c_{24} & c_{25} & \delta_{26} \\ \delta_{31} & \delta_{32} & c_{33} & c_{34} & c_{35} & \delta_{36} \\ c_{41} & c_{42} & \delta_{43} & \delta_{44} & \delta_{45} & c_{46} \\ c_{51} & c_{52} & \delta_{53} & \delta_{54} & \delta_{55} & c_{56} \\ c_{61} & c_{62} & \delta_{63} & \delta_{64} & \delta_{65} & c_{66} \end{bmatrix} \times \begin{bmatrix} F_X \\ F_Y \\ F_Z \\ M_X \\ M_Y \\ M_Z \end{bmatrix}$$

Les termes contribuant à la sensibilité sont repérés par Δ_{ij}, les termes de couplage c_{ij}.

Les dimensions actuelles du capteur, ainsi que les points où sont collées les jauges tendent à trouver un compromis entre l'augmentation de la sensibilité des termes Δ_{ij} et la minimisation des termes c_{ij}. Ceci a fait l'objet d'une optimisation [CHA 81].

Nous n'avons pas pu à partir des signaux fournis par les six conditionneurs obtenir une matrice diagonale. Il existe des termes de couplage entre les différents ponts pour l'expression des forces et des moments.

Néanmoins, nous pouvons différencier deux groupes de mesures.

Le premier,comprenant les ponts de jauges PJ_1,PJ_2,PJ_3, permet d'obtenir par une combinaison linéaire les grandeurs F_X,$F_Y$et M_Z. Les termes de couplage liés aux autres ponts étant faibles.

Le second, composé de PJ_4,PJ_5,PJ_6, nous servira à la mesure de M_X, M_Y et F_Z avec des termes de couplage liés aux autres ponts plus importants que dans le premier groupe surtout pour la mesure des moments.

Le vecteur des efforts, noté $F = (F_X\ F_Y\ F_Z\ M_X\ M_Y\ M_Z)^T$ s'obtiendra par la multiplication matricielle de SIGTO par le vecteur représentant les six signaux issus des conditionneurs.

$$F = SIGTO \cdot SIG$$

$SIG = (S_1\ S_2\ S_3\ S_4\ S_5\ S_6)^T$ vecteur des tension analogiques exprimé en niveaux

S_1,S_2,S_3 ——— PJ_1,PJ_2,PJ_3

S_4,S_5,S_6 ——— PJ_4,PJ_5,PJ_6

Les S_i sont exprimées en niveaux. 1 niveau = 10 millivolts.
Le vecteur des déplacements s'exprimera alors par :

$$U = K \cdot SIGTO \cdot SIG$$

Remarque sur la sensibilité du capteur

Si nous considérons qu'une variation de dix niveaux à la sortie des conditionneurs est facilement mesurable, nous obtiendrons pour la sensibilité minimale du capteur, les résultats suivants :

Le gain G_i est fixé à 2000 pour les ponts PJ_1, PJ_2, PJ_3 et G_i est égal à 800 pour PJ_4, PJ_5, PJ_6. Nous obtenons :

Mesure F_X, F_Y

- Sensibilité identique pour F_X et F_Y ; nous pourrons détecter un effort de 0.05 DaN (c'est-à-dire 50 grammes) suivant les axes X et Y ;

Mesure F_Z

- Le capteur est moins sensible suivant l'axe vertical, puisque pour un signal égal à 60 millivolt (6 niveaux) nous aurons une force F_Z de 0.15 DaN (c'est-à-dire 150 grammes) ;

Mesure M_X, M_Y

- Pour les moments autour des axes X et Y, nous avons la même sensibilité qui est de l'ordre de 5 à 10 mm DaN, toujours pour 10 niveaux ;

Mesure M_Z

- La mesure du moment autour de l'axe vertical Z, est de loin la plus sensible ; quand nous avons 10 niveaux en sortie des ponts PJ_1, PJ_2, PJ_3 nous obtenons 1 mm x Dan comme valeur de M_Z.

Les valeurs maximales des efforts que nous mesurerons sont de l'ordre de 2 DaN pour les forces F_X, F_Y, F_Z, 200 mm x DaN pour M_X et M_Y et 70 mm x DaN pour M_Z, avec les gains ci-dessus.

IV.3.3.4. Le traitement de l'information

C'est un micro-processeur 8085 de chez INTEL, qui assurera cette fonction.

Le langage et les programmes qui y sont implantés (voir chapitre V) permettent le traitement de l'information venant du capteur d'effort et son transfert vers le minicalculateur.

Il est muni d'un dispositif d'acquisition de tensions analogiques sur huit voies. Nous utilisons six de ces voies pour acquérir les tensions provenant des conditionneurs des jauges.

Les tensions sont exprimées en niveaux. 1 niveau = 10mv. Le seuil maximal d'acquisition est de ± 5,12 Volts ou ± 512 niveaux. Les valeurs dont nous disposons sont des entiers relatifs.

de 0 mV à 5 mV	0 niveau
de 5 mV à 15 mV	1 niveau

Le micro-processeur peut procéder au calcul du vecteur des efforts à partir des signaux du conditionneur, car son langage permet la multiplication matricielle. Nous pourrons également l'utiliser pour faire le zéro virtuel des ponts de jauge compte tenu des dérives possibles. L'utilisation précise que nous en faisons, est donnée au chapitre V.

La communication avec le mini-calculateur est réalisée par une ligne asynchrone à 9600 bauds.

IV.4. LE MINICALCULATEUR

Le miniordinateur dont nous disposons est un mitra 125, utilisé en mode 15. C'est un ordinateur temps réel, travaillant sur 16 bits.

Sa mémoire principale est de 32 K mots de 16 bits (ou 64 K octets de 8 bits) avec un cycle de base de 800 nas, ce qui représente un débit de 2,5M d'octets par seconde.

Il est possible d'utiliser une mémoire externe ; c'est un disque d'une capacité de 10 Ma.

Dans notre structure, il occupe le plus haut niveau de contrôle. Outre les calculs mathématiques, il se charge de la gestion des deux micro-processeurs et des consoles de communication (contrôle, opérateur, graphique).

IV.4.1. Gestion des entrées-sorties

Les liaisons entre le calculateur et les périphériques que nous utilisons (microprocesseurs, consoles) se font par lignes asynchrones. La vitesse de transmission varie de 300 b à 9 600 b. Nous avons choisi de travailler de la manière suivante.

- Microprocesseurs :
  Le calculateur programme les microprocesseurs dans le langage qui a été développé en leur transmettant des chaînes de caractères. Il peut lancer des exécutions de programme, recevoir de l'information en retour. Cela nécessite des programmes spécifiques pour ne pas perdre de l'information (voir chapitre V). Chaque microprocesseur est relié au calculateur par une ligne asynchrone. La vitesse de transmissior est de 4800 bauds pour le micro qui commande les moteurs. Elle est de 9600 bauds pour le micro qui gère le capteur d'effort.

- Les consoles :
  .Le Télétype opérateur permet le dialogue utilisateur calculateur, le lancement des programmes, la mise au point des manipulations. La communication se fait à la vitesse de 9600 bauds ;

  .La console graphique permet la visualisation des expérimentations en cours. La vitesse de transmission est de 9600 bauds ;

  .Le télétype de contrôle, relié au Mitra par ligne asynchrone, elle travaille à la vitesse de 300 bauds. Chargement du moniteur, du programme principal. Elle nous permet de reprendre le contrôle en cas de déroutement du programme.

IV.4.2. Implantation des programmes

Mis à part quelques programmes de base écrits en assembleur (graphique, conversion...), nous avons implanté notre logiciel en Fortran IV.

La taille de nos programmes étant plus importante (70 K octet) que celle de la mémoire centrale, nous avons dû utiliser la mémoire externe et donc adopter une structure de recouvrement appelée communément "overlay".

Structure "d'overlay"

Cette structure est assimilable à un arbre, dont la racine contient le programme principal, les données communes et des fonctions de base. Les autres primitives sont éclatées en plusieurs branches indépendantes. Un sous programme se trouvant dans une branche ne peut en appeler un autre se situant dans une branche parallèle. Il ne peut appeler que des programmes antécédents de sa branche.

Nous devons donc au moment de la description de cet arbre, faire extrêmement attention aux problèmes de recouvrement de tâches et de transfert de données.

La structure de recouvrement adoptée nous autorise donc à avoir une masse de programme supérieure à la mémoire centrale du calculateur. Les manipulations envisagées peuvent être alors réalisées, mais le temps de calcul est notablement augmenté, car le calculateur doit transférer continuellement les programmes résidant sur le disque.

IV.5. CONCLUSION

Le robot Jaz ne disposant que de quatre degrés de liberté, ceci présente un handicap pour traiter les problèmes d'insertion. Nous avons donc conçu une table à deux degrés de liberté, rotation autour de 2 axes orthogonaux, commandable par moteur à courant continu. L'adjonction de cet autre "robot" fournit à l'ensemble les six degrés de liberté indispensables pour des montages complexes.

Les expérimentations étant orientées vers la maîtrise des phénomènes de contact entre pièces, le capteur d'effort est l'outil essentiel qui a dû être réalisé avec beaucoup de soin de telle sorte qu'il puisse restituer très correctement l'information sur les efforts appliqués à l'ensemble poignet-pièce.

Dans un avenir proche, le transfert du logiciel Fortran implanté sur le Mitra 15, vers un calculateur beaucoup plus puissant (Sell 32) est prévu. Ceci ne présente pas de problème d'adaptation, mais nous permettra d'éviter d'avoir recours à une structure de recouvrement("overlay"). Nous aurons alors des temps de calcul fortement diminués (facteur 100).

C H A P I T R E V

LE LOGICIEL

V.1. INTRODUCTION

Pour pouvoir mettre en oeuvre diverses expérimentations permettant de connaître le comportement du bras manipulateur dans les phases de contact, nous avons dû développer, outre le logiciel spécifique à ce type de problème un logiciel plus général permettant le dialogue opérateur-robot.

Cet outil a été conçu pour donner à de multiples utilisateurs, la possibilité de se servir du matériel existant ainsi que d'un certain nombre de programmes de base : commande du bras manipulateur, asservissements, gestion des microprocesseurs. Seule une partie spécifique à la manipulation envisagée doit être écrite par l'utilisateur potentiel.

Dans la première partie de ce chapitre, nous décrirons l'interpréteur conçu et réalisé par A. GIRAUD, qui est un langage chiffré, permettant à l'opérateur de dialoguer avec le calculateur et les microprocesseurs et de définir et exécuter des manipulations de montage. Il génère également l'appel à des fonctions écrites par l'utilisateur. Nous décrirons celles que nous utilisons.

Nous expliquerons dans la deuxième partie, la manière dont sont programmés les microprocesseurs par le minicalculateur, ainsi que leur langage propre.

V.2. L'INTERPRETEUR

V.2.1. Principe

Le but d'un tel programme est de permettre à un utilisateur peu familiarisé avec les langages de programmation, la mise en place d'une manipulation complexe.

Il est écrit en FORTRAN et implanté sur MITRA 15.

L'utilisateur dipose d'une console, grâce à laquelle il envoie au minicalculateur des ordres constitués par une suite de nombres (six au maximum, séparés par des blancs). Ces nombres correspondent à une tâche à effectuer (calculs, déplacement robot, mesure par capteur...) ou à un élément d'une tâche.

Ces ordres sont en fait des instructions que nous pouvons regrouper en module pour former des tâches plus complexes.

L'exécution des différents modules, préalablement définis, conduit à la réalisation de ou des tâches correspondantes.

Le rôle de l'interpréteur est donc de décoder les instructions chiffrées. Il redemande une définition si l'instruction n'est pas conforme à la syntaxe interne, sinon il la range et attend la demande d'exécution.

En mode "exécution", l'interpréteur exécute les sous-programmes Fortran sélectionnés par les commandes fournies par l'opérateur en mode définition.

Le programme interprété peut contenir au maximum 450 nombres. Une instruction pouvant contenir de 1 à 6 nombres, un pointeur I donnera le numéro de l'instruction et un autre A, l'adresse du premier "mot" (nombre) de l'instruction à prendre en compte.

Une instruction peut comprendre de 1 à 6 chiffres que nous noterons $X_1, X_2, X_3, X_4, X_5, X_6$. X_1 définira le code,
X_2 définira un argument ou un sous-code,
X_3 à X_6 définiront des arguments variables.

Exemple

I = 1 A = 1 MODU = 1

MES 1 = INITIALISATION

	X_1	X_2	X_3
I = 1 A = 1	32.00	21.00	0.00
I = 2 A = 4	32.00	31.00	1.00

Pointeurs I et A COMMANDES

Les pointeurs permettent à l'utilisateur de sélectionner après avoir fait lister un module, l'instruction qu'il veut modifier ou l'adresse où il veut insérer une nouvelle instruction.

Il est à noter que le programme interprété peut être défini en mode conversationnel avec l'aide d'une console ou bien en "BATCH" ; nous perforons des cartes (une instruction par carte) et nous les faisons lire par le lecteur de cartes. Dans ce cas, la première carte définit le nom, six caractères au maximum, que portera le programme interprété.

Pour constituer des modules réalisant certaines fonctions, nous disposons d'un jeu d'instructions d'intérêt général. Associé à cela, nous avons des ordres d'appel à des sous-programmes Fortran résidant sur disque, où l'argument d'appel permet de sélectionner des sous-modules de ces sous-programmes. Nous disposons aussi d'un ordre permettant de modifier des variables situées dans une zone de données commune à plusieurs programmes (COMMON).

V.2.2. *Instructions d'intérêt général*

Pour constituer différents modules, correspondant à la description d'une tâche, nous devons posséder en-dehors de l'appel à des sous-programmes Fortran existant dans la mémoire du calculateur, un certain nombre d'instructions d'intérêt général.

Nous pourrons alors définir des modules contenant des boucles, des branchements (conditionnels ou non) à d'autres modules, des messages littéraux pour la clarté d'utilisation.

De plus, pour la mise au point d'un programme écrit en langage interprété, nous devons posséder des instructions permettant d'éditer les lignes de programme.

Nous pouvons diviser ces instructions en deux groupes :
- instructions exécutables,
- instructions non exécutables.

V.2.2.1. *Instructions non exécutables*

Elles sont peu nombreuses, ceux sont des instructions d'organisation.

1) Déclaration de module

code	arguments	
$X_1 = 7$	X_2	X_3

X_2 = numéro du module

X_3 = numéro d'instruction

Une étiquette est associée à la première instruction d'une suite d'instructions. Par branchement à cette étiquette, nous obtiendrons l'exécution du module, c'est-à-dire de la suite d'instructions.

2) Identification de deux modules

code	arguments	
$X_1 = 20$	X_2	X_3

X_2 = numéro module 1

X_3 = numéro module 2

Cette commande rend le module X_3 identique au module X_2. Les instructions contenues dans la définition du module X_3 sont perdues.

V.2.2.2. *Instructions exécutables*

a) Instructions d'affectation

1) Initialisation Flag

code	arguments	
$X_1 = 12$	X_2	X_3

Les "Flags" (variables de test) sont représentés par les variables entières IFL (1 à 5) situées en zone donnée commune. Nous pouvons positionner cinq variables différentes à la valeur souhaitée.

X_2 = numéro du "Flag" (de 1 à 5)

X_3 = valeur à attribuer à X_2 (0 ou 1)

b) Instructions de contrôle

1) Branchement inconditionnel à un module

code	arguments
$X_1 = 8$	X_2

X_2 = numéro du module où nous devons nous brancher

Cette instruction provoque l'exécution du module où nous nous branchons.

2) Branchement conditionnel si IFL(i) = 0

code	arg.	
X_1 = 11	X_2	X_3

X_2 = numéro du Flag IFL(X_2) compris entre 1 et 5

X_3 = numéro du module où se fait le branchement

Cet ordre entraîne bien sûr l'exécution du module où nous nous branchons.

3) Début de boucle et fin de boucle

Début de boucle

code	arguments	
X_1 = 9	ø	X_3

Fin de boucle

code	arg.	
X_1 = 10	ø	ø

Ces deux instructions permettent d'exécuter X_3 fois les ordres compris entre le début et la fin de la boucle. Equivaut à une boucle DO en Fortran ; nous pouvons aussi imbriquer les boucles.

4) Appel d'un sous-programme

code	arg.
X_1 = 15	X_2

X_2 = numéro du module considéré comme sous-programme

Cette instruction permet d'exécuter le module associé à l'étiquette X_2 et de revenir à l'exécution de l'instruction suivant immédiatement l'instruction d'appel.

5) Fin de sous-programme

code
X_1 = 16

Cette instruction doit terminer un module faisant fonction de sous-programme. Elle permet le retour à l'instruction suivant l'appel du sous-programme (Equivalent RETURN).

6) Remise à zéro de la pile des sous-programmes

code — Initialise à zéro, la pile où se trouve les numéros
$X_1 = 14$ — des modules sous-programmes.

7) Attente

code — arg.
$X_1 = 18$ — X_2 — X_2 = durée en 1/10 de seconde

Cette instruction bloque pendant la durée X_2, l'exécution du programme. Le déroulement reprend normalement quand le temps X_2 s'est écoulé.

8) Définition message et écriture

code — arg.
$X_1 = 19$ — X_2

X_2 = numéro du message à écrire ou à définir

Si X_2 inférieur à zéro ($X_2 < 0$), nous définissons le message à écrire.
Si X_2 supérieur à zéro ($X_2 > 0$), la commande $19_b X_2$ déclenche l'écriture du message.

9) Fin de programme

code
$X_1 = 99$

Quand l'utilisateur a écrit la totalité de son programme interprété, il doit le clore par cette instruction.

10) Fonction trace

code — arg.
$X_1 = 13$ — X_2

Si $X_2 = 1$, les instructions (constituant des modules) sont exécutées une par une. L'opérateur contrôle cette exécution pas à pas.
Si $X_2 = 0$, retour au mode normal.

c) Instructions d'entrée-sortie

1) Retour à la télétype

code
$X_1 = 1$

Cette instruction, incluse dans un module, provoque quand elle est atteinte, l'arrêt de l'exécution du module considéré et redonne le contrôle à l'opérateur (console).

2) Liste sur imprimante et sur disque

code

$X_1 = 2$

Une imprimante étant connectée au calculateur Mitra 15, la liste complète de tous les modules entrant dans la composition du programme interprété est écrite sur l'imprimante et sur la console opérateur. Après la fin de la dernière instruction, le calculateur est en attente d'information de la part de l'opérateur qui doit taper une commande, notée X qui signifie :

- X = 0 le programme interprété n'est pas sauvegardé sur disque,
- X = 1 le ———————— est sauvegardé sur le disque.

3) Liste sur télétype

code arg.

$X_1 = 3$ X_2

X_2 = numéro du module à lister

Cette commande écrit sur l'écran de la télétype, la liste des instructions définissant le module X_2.
L'utilisateur peut alors voir, s'il désire faire une modification, le numéro d'instruction et l'adresse du 1er nombre de cette instruction.

4) Suppression d'une instruction

code arg.

$X_1 = 4$ X_2 X_2 = numéro de l'instruction à supprimer

Deux fonctionnements sont possibles :

- Si le mode trace est actif, l'interpréteur décode $X_1 = 4$ et réécrit la ligne d'instruction à supprimer. Puis il attend une réponse de l'utilisateur : X.
  - .X = 1 la suppression est prise en compte
  - .X ≠ 1 l'interpréteur redonne le contrôle à l'utilisateur sans avoir effectué la suppression.
- Le mode trace n'est pas actif : la suppression est prise en compte sans réécriture de la ligne considérée.

5) Substitution d'une instruction

code arg.

$X_1 = 5$ X_2 X_2 = numéro de l'instruction à substituer

Même fonctionnement qu'au paragraphe précédent :

- mode trace : écriture de l'instruction X_2 et demande à l'opérateur:
  $X = 1$ substitution possible
  $X \neq 1$ pas de substitution
- substitution sans contrôle

6) Insertion d'une instruction

code arg.

$X_1 = 6$ X_2 X_2 = numéro de l'instruction

L'instruction sera insérée au-dessus de celle de numéro X_2.

- Mode trace : l'interpréteur décode $X_1=6$ et réécrit la ligne au-dessus de laquelle doit se faire l'insertion. Il attend la réponse X.
  $X = 1$ l'insertion peut se faire,
  $X \neq 1$ pas d'insertion.

Si le mode trace n'est pas actif, l'opérateur peut insérer sa nouvelle ligne d'instruction, mais il n'a pas la possibilité du contrôle.

Remarque

Après § 4 et 6, il y a renumérotation des instructions par décalage de 1 unité.

7) Lecture d'un nouveau programme

code

$X_1 = 17$

Cette commande permet de sortir du programme interprété sur lequel nous travaillons. Nous pouvons alors, soit appeler un programme interprété existant sur le disque, soit en créer un nouveau par l'intermédiaire du lecteur de cartes (les cartes lues décrivent le contenu de ce nouveau programme interprété).

V.2.3. Appel des S.P. Fortran par programme interprété

En-dehors des instructions précédentes, l'interpréteur permet à l'opérateur d'accéder à des sous-programmes Fortran résidant sur le disque du Mitra 15, de les transférer en mémoire centrale et de les exécuter.

Ces sous-programmes sont formés de plusieurs modules indépendants correspondant à des macro-instructions ; nous pouvons sélectionner les modules voulus à l'aide de leurs arguments d'appel.

De plus, pour échanger entre eux des informations, ces programmes ont des zones de données communes. Ces zones peuvent être atteintes par une commande de l'interpréteur et l'opérateur peut alors modifier une ou plusieurs variables communes. Ceci donne une grande souplesse d'emploi et s'adapte parfaitement à la structure "d'overlays" que nous sommes contraints d'utiliser vu l'insuffisance de capacité mémoire du mini-calculateur.

La figure V.1 schématise l'appel par le programme interprété de ces sous-programmes.

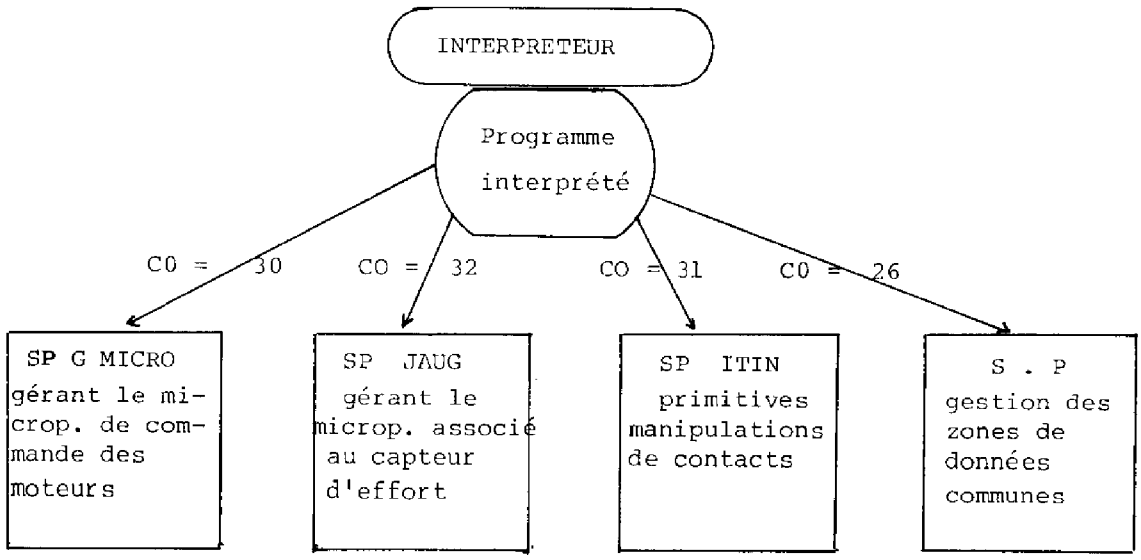


FIGURE V.1 *Appel par le programme interprété des macro-primitives*

V.2.3.1. Commandes d'appel du S.P. gérant le microprocesseur associé au capteur d'effort

La commande d'appel du sous-programme gérant le microprocesseur jauge est défini par un code, suivi d'un sous-code et de ses arguments. En tout six nombres au maximum, séparés par des blancs et que nous noterons $X_1\ X_2\ X_3\ X_4\ X_5\ X_6$.

Le code d'appel est représenté par X_1. Sa valeur est $X_1=31$. Dans la suite de ce paragraphe X_1 gardera toujours la même valeur.

Le sous-code représenté par X_2, correspond aux différents modes de dialogue avec le micro.

L'ordre 31 déclenche l'appel de la primitive JAUG dont les valeurs des arguments sont les valeurs de X_2 à X_6. Nous allons décrire les fonctions que réalise cette primitive sans rentrer dans le détail de la programmation du microprocesseur (voir V.3.1.).

1) Initialisation

Après RESET du microprocesseur, il faut déterminer la vitesse de transmission des informations.

code	sous-code	arguments
$X_1 = 31$	$X_2 = 1$	$X_3 = X_4 = X_5 = X_6 = \emptyset$

Le calculateur envoie six fois le caractère "U" (sans blanc) vers le microprocesseur, qui répond PRET.

2) Envoi de chaînes de caractères préprogrammées

Les chaînes de caractères préprogrammées dans le Mitra sont définies dans le sous-programme JAUCAR à écrire et à compiler pour l'utilisation envisagée (voir § V.3.1.).

code	s/code	argument
$X_1 = 31$	$X_2 = 2$	$X_3\ X_4\ X_5\ X_6$

$X_3 = 1$; $X_4 = k$; $X_5 = X_6 = \emptyset$

Le Mitra envoie la chaîne de caractère numéro k vers le microprocesseur, et n'attend rien en retour.

.$X_3 = 2$; $X_4 = k$; $X_5 = X_6 = \emptyset$

Le Mitra après l'envoi de la chaîne numéro k, attend du microprocesseur la réponse "?" (fin de tâche).

.$X_3 = 3$; $X_4 = k$; $X_5 = N$; $X_6 = \emptyset$

Après l'envoi de la chaîne numéro k, le Mitra se met en attente de N nombres entiers.

.$X_3 = 4$; $X_4 = k$; $X_5 = R$; $X_6 = \emptyset$

Même principe que plus haut, mais attente de R nombres réels.

3) Entrée d'une chaîne de caractères à la télétype

Ce mode autorise une programmation directe du microprocesseur par l'opérateur, mais devient vite assez lourde si les programmes à rentrer sont assez importants.

Son utilisation est intéressante dans une phase de mise au point, car elle évite la compilation de la primitive JAUCAR.

code	sous-code	Arguments
$X_1 = 31$	$X_2 = 3$	X_3 X_4 X_5 X_6

.$X_3 = 1$; $X_4 = X_5 = X_6 = \emptyset$

Entrée d'une chaîne de caractères à la console et envoi vers le microprocesseur ; le mitra n'attend rien.

.$X_B = 2$; $X_4 = X_5 = X_6 = \emptyset$

Même commande que la précédente, mais le mitra attend "?".

.$X_3 = 3$; $X_4 = N$; $X_5 = X_6 = \emptyset$

Après envoi de la chaîne de caractères, le mitra attend N entiers.

.$X_B = 4$; $X_4 = R$; $X_5 = X_6 = \emptyset$

Même chose, mais le mitra attend du microprocesseur R nombres réels.

4) Autres commandes

code	s/code	Arguments
$X_1 = 31$	$X_2 = 4$	$X_3 = R$ $X_4 = X_5 = X_6 = \emptyset$

R est un nombre réel que le mitra transmet au microprocesseur.

code	s/code	Arguments
$X_1 = 31$	$X_2 = 5$	$X_3 = N$ $X_4 = X_5 = X_6 = \emptyset$

Le mitra attend du microprocesseur N nombres entiers.

code	s/code	Arguments
$X_1 = 31$	$X_2 = 6$	$X_3 = R$ $X_4 = X_5 = X_6 = \emptyset$

Le mitra attend du microprocesseur R nombres réels.

V.2.3.2. Commande d'appel du S.P. gérant le microprocesseur du robot

Cette commande est définie de la même manière que la précédente, par un code fixe, suivi d'un sous-code variable et des arguments correspondants. Nous adopterons les mêmes notations que précédemment. Le code d'appel est représenté par X_1. Sa valeur est $X_1=30$; il génére le chargement dans la mémoire centrale du calculateur, de la primitive Fortran GMICRO ; le sous-code détermine à quelle macro-instruction de ce programme il faut se brancher ; les arguments étant les données nécessaires à la réalisation de la tâche.

Cette primitive effectue la gestion du microprocesseur de commande des moteurs pas à pas du bras manipulateur.

Elle permet de faire réaliser par le robot, les mouvements que nous voulons dans le système de coordonnées choisi (orthonomé ou cylindrique) et ceci suivant différents modes.

V.2.3.1. Description des macro-instructions

. code | s/code | Arguments

$X_1 = 30$ $X_2 = 16$ $X_3 = X_4 = X_5 = X_6 = \emptyset$

Initialisation du microprocesseur de commande des moteurs. Tabulation des politiques préprogrammées.

. code | s/code | Arguments

$X_1 = 30$ $X_2 = 6$ $X_3 = X_4 = X_5 = X_6 = \emptyset$

Exécution de la politique de numéro X_3, prétabulée dans le microprocesseur.

. code | s/code | Arguments

$X_1 = 30$ $X_2 = 7$ $X_3 = X_4 = X_5 = X_6 = \emptyset$

Choix par l'opérateur du système de coordonnées dans lequel, il définit les déplacements du bras.

$X_3 = 1$ repère orthonormé

$X_3 = 0$ repère cylindrique

. code s/code Arguments

$X_1 = 30$ $X_2 = 4$ $X_3 = X_4 = X_5 = X_6 = \emptyset$

Arrêt du manipulateur avec décélération.

. code s/code Arguments

$X_1 = 30$ $X_2 = 9$ $X_3 = X_4 = X_5 = X_6 = \emptyset$

Cette commande réalise la lecture de la position du robot qui est rangée dans la variable de numéro X_3. Nous pouvons acquérir dix positions différentes X_3 prenant les valeurs 1 à 10.

. code s/code Arguments

$X_1 = 30$ $X_2 = 28$ $X_3 = X_4 = X_5 = X_6 = \emptyset$

Ecriture des variables définies par la commande précédente. Chaque variable représente un point défini par sa position (X,Y,Z) et son orientation (ϕ autour de Z) par rapport à la référence.

La position courante du robot est toujours présente dans la variable de numéro $X_3 = 10$.

Il est à noter que le point que nous définissons occupe une position fixe sur le bras ; il est repéré par le référentiel RTRO.

. code s/code Arguments

30 14 $X_1\ X_2\ X_3$

Le calculateur transmet au microprocesseur les valeurs des accélérations maximum JMAX ($X_1\ X_2\ X_3$) associées aux degrés de liberté.

. code s/code Arguments

$X_1 = 30$ $X_2 = 18$ $X_3\ X_4\ X_5\ X_6$

Cette commande permet de faire des petits déplacements de type incrémentaux de valeur (DX,DY,DZ,Dϕ) = ($X_3\ X_4\ X_5\ X_6$).

. code s/code Arguments

$X_1 = 30$ $X_2 = 19$ $X_3\ X_4\ X_5\ X_6$

Génération de petits déplacements jusqu'à la position définie par XM(X,Y,Z,ϕ) = ($X_3\ X_4\ X_5\ X_6$)

. code s/code Arguments

$X_1 = 30$ $X_2 = 1$ $X_3\ X_4\ X_5\ X_6$

Mouvement non coordonné jusqu'à la position définie par XM(X,Y,Z,ϕ) = ($X_3\ X_4\ X_5\ X_6$). En cartésien ou en cylindrique.

. code s/code Arguments

$X_1 = 30$ $X_2 = 24$ $X_3 = X_4 = X_5 = X_6 = \emptyset$

Mouvement non coordonné jusqu'à la position contenue dans la variable de numéro X_3 (réf. commande X_1=30 X_2=9).

. code s/code Arguments

$X_1 = 30$ $X_2 = 8$ $X_3\ X_4\ X_5\ X_6$

DNP = ($X_3\ X_4\ X_5\ X_6$)

Commande dans l'espace des moteurs. L'argument DNP représente le nombre de pas moteur à faire pour chaque degré de liberté. DNP (XPAS,YPAS,ZPAS,ϕ PAS). C'est une commande de type incrémental, les valeurs de DNP doivent être petites ($\simeq$50).

V.2.3.3. Commande d'appel du S.P. utilisateur

Le principe est le même que pour les commandes précédentes ; nous avons un code de sélection du sous-programme Fortran, suivi d'un sous-code donnant l'adresse de branchement à la macro-instruction que nous voulons activer et bien sûr les arguments appropriés.

Cette primitive a pour nom ITIN, son code de sélection est X_1=32. Ces arguments sont au nombre de deux X_2,X_3.

C'est la partie la plus spécifique à la manipulation envisagée. Elle contient des fonctions d'intérêt général (gestion de repères, tableaux,...) mais aussi des modules propres à l'expérimentation souhaitée.

Cette primitive doit donc être écrite ou modifiée par l'utilisateur suivant les tâches qu'il s'est définies ou les capteurs dont il dispose (possibilité traitement image...).

Le chapitre II a abordé une partie des fonctions regroupées dans cette primitive. En fait, la primitive ITIN, pour tenir compte des problèmes de recouvrement ("overlay"), appelle deux sous-primitives importantes ITINB et ITINC. Dans chacune de ces sous-primitives nous avons regroupé des modules utilisant des sous-programmes différents. Ceci nous a permis de mettre dans des branches parallèles de notre "arbre d'overlay" ces deux primitives (cf. Annexe).

Si nous n'avions pas adopté cette structure, la taille mémoire du calculateur aurait été insuffisante pour contenir notre programme.

Nous donnons ci-dessous les fonctions réalisées par le sous-programme ITIN suivant la valeur de ses arguments X_2 et X_3.

.(X_2,X_3) = (1, .)
Cette commande permet d'initialiser le microprocesseur de commande des moteurs du robot, pour le positionner dans le mode de travail de l'asservissement en force et position.

.(X_2,X_3) = (2,N)
Asservissement en force et position (un coup) dans repère de consigne défini par N. (En général, le repère N correspond au repère RCCO).Si N est négatif, il y a impression de la valeur des efforts mesurés pour l'exécution de l'asservissement ainsi que des incréments de déplacement générés sur chaque axe.

.(X_2,X_3) = (6,.)
Sortie du mode de travail du microprocesseur spécifique à l'asservissement.

.(X_2,X_3) = (3,.)
Redéfinition après un mouvement de la position du robot par rapport à sa référence (actualisation RTRO par rapport à REF).

.(X_2,X_3) = (4,.)
Tracé sur la console graphique du déplacement du robot.

.(X_2,X_3) = (5,.)
Fonction spécifique permettant de faire évoluer le repère (RCO) dans lequel se fait l'asservissement vers le point où se fait le contact.

(cf. chapitre II.3.1.1.).

Les commandes suivantes appellent des fonctions qui sont réalisées par la sous-primitive ITIN B.

- Commandes d'initialisation

.(X_2,X_3) = (21,N)

N = 1 initialisation du microprocesseur gérant le capteur d'effort

N = 2 initialisation du microprocesseur commandant les mouvements du robot

N = 3 initialisation de la primitive GRARO gérant la représentation de l'univers du robot par des repères tridimensionnels

N = 0 regroupe en une seule fonction les trois initialisations précédentes

N = 4,5,6 initialisation des paramètres de l'asservissement à la télé-type-opérateur

- Commandes se rapportant à la gestion du microprocesseur du capteur d'effort

.(X_2,X_3) = (23,.)

Réalise l'acquisition du signal en provenance des conditionneurs jauges pour faire le zéro virtuel des ponts de jauges.

.(X_2,X_3) = (24,.)

Lancement d'une politique sur le microprocesseur du capteur d'effort permettant de tester si une force ou un moment dépasse un certain seuil .

.(X_2,X_3) = (22,.)

Modification du seuil S précédent.

.(X_2,X_3) = (25,.)

Arrêt de la politique comparant les efforts à un seuil.

.(X_2,X_3) = (25,10)

Fait suite à la commande X_2=24 ; permet de transférer vers le minicalculateur le vecteur des efforts.

.(X_2,X_3) = (26,4)

Acquisition du signal en sortie des conditionneurs, envoi vers le minicalculateur, obtention du torseur de force dans le repère RJAU.

Les commandes suivantes appellent des fonctions qui sont réalisées par la sous-primitive ITINC.

- Commandes d'ordre général

.(X_2,X_3) = (31,.)

Remise à zéro d'un tableau utilisé pour mémoriser les torseurs de force.

.(X_2,X_3) = (33,.)

Création d'une classe (type A) de nom NPE en sortie

.(X_2,X_3) = (34,.)

Destruction de toutes les classes de type A, mais pas de leur sous-classes (type B).

.(X_2,X_3) = (35,.)

Destruction de la classe de type A de nom NPE et des sous-classes associées.

- Commandes plus spécifiques à la gestion des repères

.(X_2,X_3) = (36,N)

Cette commande est relative à l'appel de la primitive GRARO (cf. II.2.2) de gestion des repères tridimensionnels.

N = 16 donne la liste des repères sur l'imprimante

Pour les autres valeurs de N, l'opérateur devra donner à la console un supplément d'information concernant les repères sur lesquels il veut travailler.

N = 3 efface une liaison entre deux repères

N = 4 création d'une liaison par recherche d'un chemin entre deux repères

N = 13 création d'un repère identique à un autre

N = 15 création d'un repère par donnée de sa position et orientation

.(X_2,X_3) = (37,.)

Contrairement à la commande précédente, celle-ci permet l'appel du sous-programme GRARO quand ses arguments ont été au préalable définis en COMMON. L'opérateur n'a pas à entrer d'information supplémentaire. Cette commande est fondamentale dans la création d'un programme interprété.

Exemple

```
          MCDL= 18
I=110 A=335       26.00        4.00
I=111 A=337       23.00     1902.00        6.00        7.00       15.00
I=112 A=342       23.00    -1905.00        0.00       40.00        0.00
I=113 A=347       23.00    -1911.00        0.00        0.00        1.00
I=114 A=352       23.00    -1917.00        1.00        0.00        0.00
I=115 A=357       15.00       21.00

I=138 A=429  MCDL= 21

I=138 A=429       32.00       37.00        0.00
I=139 A=432       16.00
```

Les instructions 110 à 114 correspondent à la définition des arguments de la primitive GRARO.

L'instruction 138 réalise l'appel de la primitive.

Les commandes relatives à X_2, variant de 41 à 48, ont été décrites au chapitre II.4.2.

- Commandes propres aux manipulations de contact

.(X_2,X_3) = (51,N)

Rangement dans un tableau (50 places) du torseur de force mesuré dans le repère RJAU. Représentation par une droite, exprimée dans le repère N, du torseur précédent. Rangement de cette droite dans un tableau mémoire TDR (50 places). La déformation du capteur d'effort est prise en compte (RJAU ⟷ RDEF).

N < 0 la déformation du capteur n'est pas prise en compte

.(X_2,X_3) = (52,.)

Classification des droites pour extraire le point de contact.

.(X_2,X_3) = (52,2)

Comme précédemment, mais l'opérateur peut modifier le seuil de classification.

.(X_2,X_3) = (53,.)

Liste des différents points de contact obtenus et des essais correspondants.

.(X_2,X_3) = (54,.)

Création du repère RCCO par rapport à RCO par des variables définies dans un COMMON.

.(X_2,X_3) = (55,.)

Etalonnage repère du capteur (RJAU) par rapport au repère du robot (RTRO).

.(X_2,X_3) = (56,.)
Détermination d'une arête.

.(X_2,X_3) = (57,N)
Création du repère RCCO par rapport à RCO par une translation de N millimètres suivant l'axe des X.

.(X_2,X_3) = (58,N)
Création du repère RCCO par rapport à RCO par une translation de N millimètres suivant l'axe des Y.

.(X_2,X_3) = (59,N)
Création du repère RCCO par rapport à RCO par une translation de N millimètres suivant l'axe des Z.

.(X_2,X_3) = (60,N)
Création du repère RCCO par rapport à RCO par une rotation de N degrés autour de l'axe des Z.

V.2.3.4. Commande d'entrée-sortie dans zonesde données communes

La taille de la mémoire centrale du minicalculateur Mitra 15 étant insuffisante pour contenir tous les programmes Fortran dont nous avions besoin, nous avons dû faire intervenir une structure d'overlay permettant d'utiliser le disque comme zone mémoire (cf. chapitre II.3).

Muni d'une telle structure, le passage par arguments des données et résultats entre ces différents programmes, devient dangereux. Nous avons donc choisi d'utiliser des zones de données communes à plusieurs programmes, dans lesquelles seraient rangées les variables que nous voulons sauvegarder.

Ces zones communes étant placées dans la racine de l'arbre, il n'y a pas de destruction possible des volumes de ces variables. Ces zones communes sont appelées dans le langage Fortran des COMMON. Nous utiliserons cette appellation par la suite.

L'intérêt d'une telle configuration en-dehors du fait de sauvegarder les variables, serait assez minime si nous n'avions pas accès à ces blocs de données communes.

L'interpréteur peut décoder plusieurs commandes qui permettront de sélectionner une zone précise et d'aller y lire ou écrire à l'adresse d'implantation des variables que l'opérateur veut atteindre.

La liste des COMMONS et l'adresse d'implantation des variables dans chaque COMMON seront données en Annexe.

V.2.3.4.1. Commande de sélection d'un COMMON

Nous disposons actuellement de huit blocs de données communes différents. Nous ne détaillerons pas leur contenu (cf. Annexe) mais expliciterons la commande permettant de les sélectionner.

Code	Arguments	
$X_1 = 26$	X_2	X_2 = numéro du COMMON choisi

Nous pouvons alors faire des opérations d'entrée-sortie sur le COMMON choisi. X_2 varie de 0 à 8.

A l'exécution, le décodage de cette commande entraîne l'écriture du COMMON défini par X_2.

Si X_2=0, l'interpréteur écrit le nom du bloc sur lequel nous travaillons.

V.2.3.4.2. Entrée-sortie de variables sises dans un COMMON

Au préalable, le bloc commun a été choisi par la commande 26.

a) Lecture d'un tableau d'entiers ou de réels

Après le choix d'un COMMON, nous pouvons faire écrire sur la console un tableau de variables entières ou réelles, qui sont repérées par leur position dans le COMMON, en nombre de mots.

Cette commande est décrite par :

code	Arguments
$X_1 = 25$ X_2	X_3, X_4, X_5, X_6

X_2 = adresse en nombre de mots depuis le début du COMMON, de la première variable à lire

$X_2 > 0$ pour des nombres entiers (un mot par entier)

$X_2 < 0$ pour des nombres réels (deux mots par réel)

Les arguments X_3 X_4 X_5 X_6 définissent les dimensions du tableau à écrire. Notons que le nombre maximum de colonnes est égal à 6.

X_3 = pas colonne X_4 = nombre de colonnes

X_5 = pas ligne X_6 = nombre de lignes

Exemple

```
   26.4        appel du COMMON n°4
 CTRAV         réponse du calculateur qui écrit le nom du COMMON
 25 - 1  1 3 3 3   demande d'écriture d'un tableau de 3 lignes et 3 colonnes,
                   à partir de l'adresse 1 du COMMON. Les variables écrites
                   sont des réelles.
1 . 2 . 3 .
4 . 5 . 6 .
3 . 8 .12 .  écriture par le calculateur
```

b) Ecriture dans un COMMON de 1 à 4 variables

Cette commande est essentielle pour la description d'une manipulation à l'aide d'un programme interprété.

Elle permet de modifier des variables dans une zone de données communes à plusieurs programmes pour engendrer des applications différentes (asservissement, gestion repère...).

Le code de commande est variable suivant que nous voulons écrire une ou plusieurs variables entières ou réelles.

. code	Argument
$X_1 = 21$	X_2, X_3

X_2 a la même signification qu'au paragraphe précédent ; c'est l'adresse en mot depuis le début du COMMON.

$X_2 > 0$ pour écrire des entiers ; $X_2 < 0$ pour écrire des réels.

Cette commande réalise l'écriture d'un nombre réel ou entier X_3 selon le signe de X_2 à partir de l'adresse X_2.

. code	Argument
$X_1 = 22$	X_2, X_3, X_4

Même signification que plus haut, mais écriture de deux nombres X_3 X_4.

. code	Argument
$X_1 = 23$	X_2, X_3, X_4, X_5

Ecriture de trois nombres X_3 X_4 X_5

. code	Argument
$X_1 = 24$	X_2, X_3, X_4, X_5, X_6

Ecriture de 4 nombres X_3 X_4 X_5 X_6

Exemple

26 4 sélection par l'opérateur du COMMON de travail, il porte le numéro 4.

CTRAV

24 -1 15 20-5 -7 4

entrée à partir de l'adresse -1 du COMMON CTRAV de quatre nombres réels consécutifs 15, 20.5, -7 et 4

V.2.4. *Définition d'une manipulation à l'aide de l'interpréteur*

L'intérêt essentiel de l'interpréteur réside dans la possibilité de construire à partir des commandes précédemment définies, un enchaînement de tâches représentant une manipulation complexe.

C'est un programme "Maître" qui fixe le déroulement des opérations. L'exécution d'une tâche ou de plusieurs est réalisée par la commande de branchement inconditionnel à un module.

Le programme interprété sera donc conçu comme un programme classique écrit en Fortran.

Le module dont nous demandons l'exécution fait fonction de programme principal ; il appelle les sous-modules qui sont en réalité des sous-programmes.

Au cours de l'exécution, le programme interprété peut dialoguer avec le bloc entrée-sortie demandant des données supplémentaires ou fournissant des résultats numériques sur la console opérateur, la console graphique ou tout autre périphérique prévu à cet effet.

Quand la ou les tâches sont effectuées, il redonne le contrôle à l'opérateur, qui peut alors relancer un programme ou redéfinir des modules.

D'autre part, dans la phase de développement d'une manipulation, l'interpréteur détecte les erreurs syntaxiques commises dans la définition des instructions. A ce moment là, l'instruction mal définie n'est pas prise en compte et l'interpréteur génère un message d'erreur dans lequel il indique le nombre d'arguments attendu d'après le code frappé (c'est le premier nombre, noté X_1, d'une commande). L'utilisateur a alors la possibilité de redéfinir son instruction. Ceci permet d'éviter certaines erreurs difficilement détectables à l'exécution.

Pour avoir des exemples d'utilisation de l'interpréteur, nous pouvons nous reporter au chapitre VI, qui décrit trois expérimentations définies à l'aide de trois programmes interprétés.

V.3. LA GESTION DES MICROPROCESSEURS

Les deux microprocesseurs que nous utilisons sont identiques physiquement, mais leurs fonctions sont bien séparées.

Nous avons donc deux langages de commande spécifiques à ces deux micros. A partir de ces langages que nous détaillons plus bas, nous avons dû gérer la liaison Mitra 15 -Microprocesseur- c'est-à-dire concevoir les programmes qui prétabulent des "politiques" dans les deux microprocesseurs, les exécutent et peuvent éventuellement les modifier.

V.3.1. Le Microprocesseur "jauge"

V.3.1.1. Le langage du microprocesseur jauge

Pour faciliter l'utilisation du microprocesseur par différents opérateurs, un mini langage composé de macro-instructions a été développé. Nous donnons ci-après la liste et les fonctions de ces commandes.

MICROPROCESSEUR "JAUGE de CONTRAINTES"

LISTE DES COMMANDES

Rem: Format Z1 sauf indication contraire.

CO='O',NV,FV,LIG,COL	: Définition variable
CO='I',NV2,NV1	: NV2 est équivalente à NV1 prédéfinie
CO='2',NV,NV2,NV1	: NV est déplacée de sa longueur.Si son implantation dépasse ou est égale à celle de NV2,NV est réimplantée en NV1.(Implantation circulaire)
CO='3',Nbre de voies,Liste des numéros(0 à 7) de voies	: Définition du dispositif d'acquisition des tensions analogiques.
CO='A'	: Initialisation totale,désarmement '?', mode Mitra.
CO='D'	: Fin fichier politique et/ou réarmement de '?'
CO='4'	: Désarmement de '?'
CO='B'	: RàZ de l'espace mémoire'variables'
CO='6',NUM	: Exécution de la politique NUM
CO='7',NUM	: Tabulation de la politique NUM.La liste des caractéres se termine par 'D','G'
CO='C',BATT(Z4)	: Attente en millisecondes approximatives
CO='E',IMA	: =O mode Mitra,=1 mode télétype
CO='5',NV	: Acquisition analogique dans NV
CO='8',OP,NV1,NV2,(NV3)	: Opérations à 2 ou 1 opérandes.
CO='9',OP,NV '9',OP,Addresse(Z4)	: Opérations d'entrée/sortie
CO='F',SCO, NUM	: Opérations de branchement dans table politiques,SP,boucle DO,Flag....

Pour la définition de variable, FV décrit le type de la variable créée :

FV = 2 pour un entier,

FV = 4 pour un réel.

Détail des commandes CO="8"... CO="9"... CO="F"...

CO='8',OP,NV1,NV2,(NV3) OPERATIONS

OP= 0 : Multiplication normale de matrice
≠ 0 : Opérations terme à terme (lus cycliquement,les colonnes évoluant le plus rapidement)

= I ADDRR
= 2 SUBRR
= 3 MULRR
= 4 ADDEE
= 5 SUBEE
= 6 ADERR Entier plus réel dans réel
= 7 SUERR
= 8 MUERR
= 9 DIVRR
= A TERR Si (NVI) inférieur à(NV2)-au moins un terme- alorsFLA2=I
= B CONER Conversion entier en réel
= C CONRE
= D DANS Les valeurs de NVI sont transférer dans NV2

CO='9',OP,NV ENTREE/SORTIE
CO='9',OP,MEF(Z2),HEF(Z2)

OP= 0 ENTR Entrée d'un réel
= I ENTE
= 2 SORR
= 3 SORE
= 4 ABSR Valeur absolue d'un réel
= 5 Transposé d'un vecteur
= 6 Indicateur de charge mémoire RAM(valeurs numériques,table politiques)
= 7 DUMP mémoire à partir adresse (Z4) donnée.
= 8 SUBstiturion mémoire à partir adresse donnée (Z4)
= 9 Sortie sur port des bits définis par MEF ,prenant la valeur HEF, les autres bits étant non affectés.
= A Sortie des valeurs MEF,HEF (cf OP=9) si FLA2=1

CO='F' ,SCO,.... BRANCHEMENTS

SCO= 0,FLA(1 à 3),NUM : Branchement à politique NUM si FLAi=1
= I,FLA ,Valeur : FLAi prend la valeur indiquée

= 2,CDO(Z2) : Initialisation d'une boucle "DO"
= 3 : Fin de boucle "DO" précédente

= 4 , NUM : Appel de sous-programme défini par lapolitique NUM.
= 5 : Fin de sous-programme

= 6 Envoi de FLAI,FLA2,FLA3 sur ligne asynchrone.

Nous disposons donc d'un jeu d'instructions assez complet regroupant : définition de l'espace mémoire des tableaux de valeurs entières ou réelles, calcul, définition de la communication par ligne asynchrone et de la synchronisation, tabulation, lecture, branchement dans la structure "politique" (macro-commandes).

En mode Mitra, les entiers sont envoyés en format Z_2 ou Z_4, les réels en format Z_8. Il faut alors transcoder le réel du microprocesseur dans le code nombre réel du calculateur récepteur. Le microprocesseur reçoit en format libre F (fin par "retour chariot").

V.3.1.2. Programmation du Microprocesseur par le Mitra

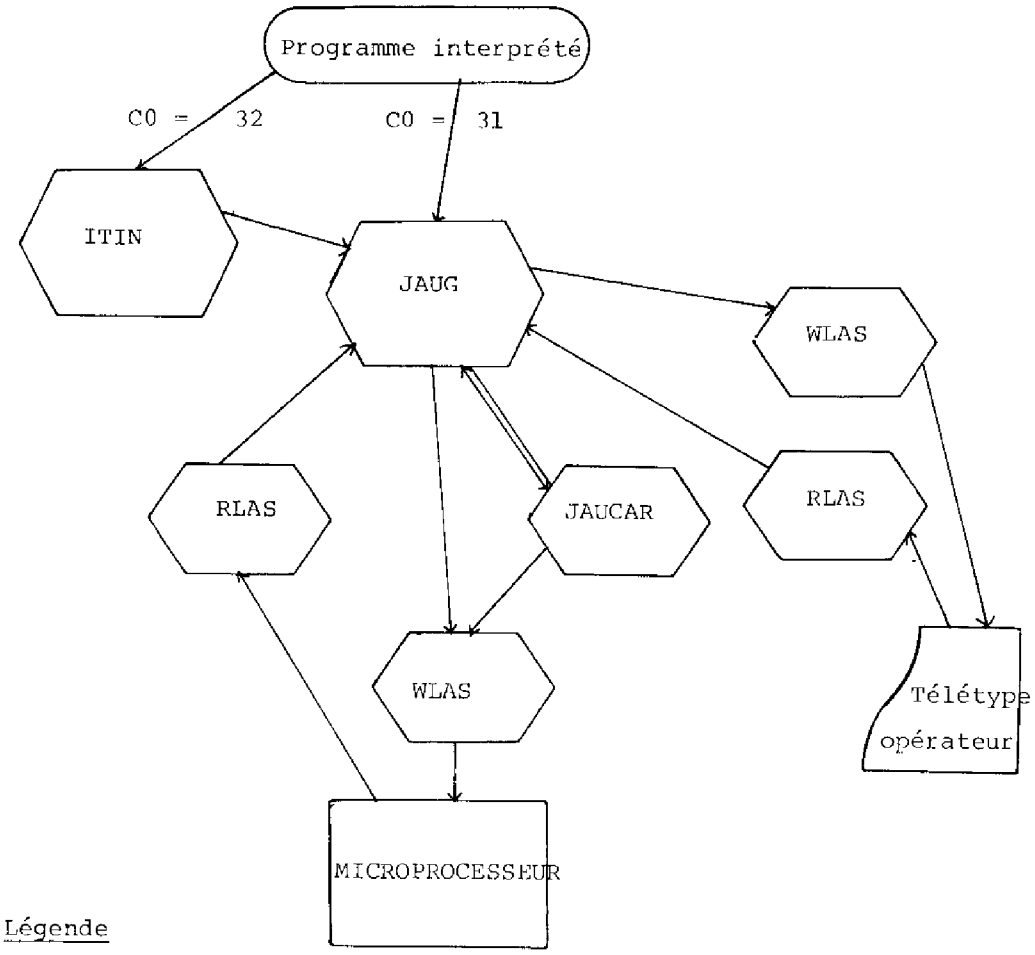
Nous avons vu précédemment l'utilisation possible du microprocesseur en mode "Mitra" ou "télétype". Dans le cas d'une manipulation intégrée, le Mitra a le plus haut niveau de contrôle et il assume donc la gestion du Microprocesseur.

Nous avons donc créé des primitives qui permettent la tabulation de politiques et leur exécution dans le microprocesseur.

La primitive principale est le sous-programme JAUG (cf. V.2.3.1), qui détermine la vitesse et le mode d'envoi de chaînes de caractères vers le micro, ainsi que l'état du calculateur suivant la réponse que lui fournira le microprocesseur.

Les différents modes de fonctionnement de cette primitive ont été explicités précédemment. Elle peut être appelée par le programme interprété ou par l'intermédiaire du programme utilisateur ITIN.

Nous pouvons illustrer son fonctionnement par le schéma suivant.



Légende

NOM = Sous-programmes Fortran

Nous laisserons de côté le mode de programmation du microprocesseur par entrée de chaînes de caractères à la télétype, car son utilisation est trop spécifique.

Nous détaillerons l'envoi des chaînes de caractères par la primitive JAUCAR, ainsi que leur signification.

1) Tabulation du microprocesseur par la primitive JAUCAR

Cette primitive (Fortran) ne possède qu'un seul argument ; c'est une variable entière K, sa valeur permet de sélectionner le bloc de caractères à envoyer vers le microprocesseur. Toutes les politiques sont tabulées par le même envoi. Si nous modifions des affectations de variables précédemment

définies, nous retabulons systématiquement toutes les politiques. Cette précaution doit être prise pour empêcher l'empilement des programmes et le déroutement du microprocesseur.

Détail des commandes de JAUCAR

K = 1	tabulation de toutes les politiques et exécution de la politique numéro 1 : initialisation
K = 2,3,4,5	exécution des politiques 2,3,4,5 se trouvant dans le microprocesseur
K = 6	modification d'une variable fondamentale sise dans la politique numéro 2
K = 7	arrêt d'une politique en cours d'exécution

2) Description fonctionnelle des politiques tabulées

Nous allons décrire la fonction des différents programmes que le Mitra implante sur le microprocesseur.

Politique 1 : Initialisation totale des variables programmées dans le microprocesseur, puis définition des variables nécessaires aux calculs à conduire. Sélection des voies du dispositif d'acquisition des tensions analogiques.

Politique 2 : Etat du signal délivré par les ponts de jauges, quand le système est au repos. C'est la variable NV4 qui sert à définir le zéro virtuel. Cette variable est transférée vers le calculateur pour que l'opérateur puisse en visualiser les valeurs. La variable NV4 est entière.

Politique 3 : Le microprocesseur Jauge teste en permanence si les signaux en provenance du capteur de force sont inférieurs à un certain seuil. Si un des ponts de jauges (resp. forces ou moments) délivre après amplification un signal supérieur au seuil fixé par programme ou par l'utilisateur, il se branche à l'exécution de la politique 6 (voir plus loin).

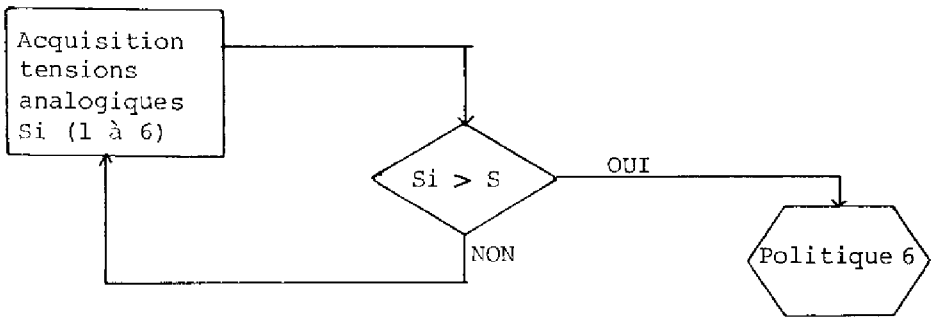


Schéma de fonctionnement de la politique 3

Politique 6 : La politique 6, test de sortie de la précédente génère l'envoi d'un nombre entier égal à 1 vers le calculateur.

Une extension possible est l'envoi d'un bit d'arrêt d'urgence vers le micro de commande des moteurs pas à pas du bras manipulateur.

Politique 4 : Déclenche l'acquisition du vecteur tensions analogiques (6 valeurs) en provenance des ponts de jauges, et les range dans la variable NV3, qui est entière. Puis la valeur de NV4 (zéro des jauges) est soustraite de NV3. Le résultat est rangé dans NV3. Cette même variable est alors transmise au minicalculateur Mitra 15 qui fera la multiplication nécessaire par la matrice SIGTO (cf. IV.3.3) pour obtenir le vecteur des efforts. Il est à noter que la variable NV3 est une variable dite tournante, c'est-à-dire qu'après transfert vers le Mitra 15, elle est déplacée de sa longueur et ceci jusqu'à concurrence de 50 fois sa longueur. Nous pouvons alors avoir en permanence dans le microprocesseur les cinquante dernières acquisitions faites sur les conditionneurs des ponts de jauge.

Politique 5 : Cette politique permet de lire la variable NV3. C'est-à-dire que le contenu de NV3 est transféré au minicalculateur puis la variable NV3 est déplacée de sa longueur comme dans la politique 4. Ceci permettra au minicalculateur d'aller lire quand il le désire les cinquante dernières acquisitions analogiques mémorisées dans la variable tournante.

Remarque

Bien que le jeu d'instructions créé sur le microprocesseur permette la multiplication matricielle, nous ne l'avons pas utilisé pour transformer le vecteur des tensions analogiques en vecteur des efforts. Ceci pour la simple raison que le calcul demandait davantage de temps sur le microprocesseur (conversion entier réel puis multiplication matricielle) que sur le minicalculateur. C'est pourquoi nous nous sommes contentés de transformer l'information représentative des efforts vers le Mitra 15 qui se charge de la transformer.

V.3.2. *Le microprocesseur de commande du bras*

Ce microprocesseur gère la position du robot et génère la commande des moteurs du bras manipulateur en fonction des données transmises par le minicalculateur.

Il travaille dans l'espace des coordonnées généralisées. Le minicalculateur contrôlera donc son mode de fonctionnement et lui enverra les consignes en nombre de pas-moteur qu'il devra réaliser sur chacun des quatre axes.

.Gestion de la position

La position du robot sera représentée par le nombre de pas-moteur (ceci pour chaque moteur), relatif à une position de référence du manipulateur située en butées arrières. Ces quatre valeurs sont rangées dans une variable dont le minicalculateur peut demander la lecture.

.Mode de travail

Plusieurs types de fonctionnement sont possibles sur le microprocesseur. Nous ne détaillerons que le mode utilisé par l'asservissement décrit au chapitre III. En effet, cet asservissement générant des déplacements incrémentaux, doit pouvoir travailler le plus rapidement possible avec le microprocesseur.

Le minicalculateur envoie la commande constituée par les caractères "2E0" pour positionner le microprocesseur dans le mode de travail voulu. Puis il transmet les incréments de déplacement (en nombre de pas moteur) à générer sur chaque axe, ainsi que la vitesse du mouvement pour chaque axe.

En retour le microprocesseur renvoie le nombre de pas qu'il a exécuté sur chaque axe entre l'exécution de l'avant-dernière commande et la commande qu'il vient de recevoir. Nous travaillons alors dans un mode échantillonné non synchrone qui nous permet de ne pas attendre la fin de l'exécution d'une commande pour en transmettre une nouvelle.

La position absolue est gérée par le minicalculateur qui la réactualise en fonction des données renvoyées par le microprocesseur.

La vitesse de transmission étant de 4800 bauds, nous arrivons à travailler avec deux échantillonnages par seconde. Cette vitesse est assez lente, mais il faut tenir compte du fait que le minicalculateur, outre les calculs des coefficients de l'asservissement doit aller lire sur un autre microprocesseur les signaux issus du capteur de force et ceci à chaque passage dans la fonction asservissement.

V.4. CONCLUSION

L'interpréteur tel qu'il est conçu actuellement permet à l'opérateur averti de définir des tâches complexes relatives au montage, à l'assemblage.

Nous avons pu mesurer toute sa puissance dans les phases de développement et d'expérimentations : génération de contact, reconnaissance d'objet, asservissements.

D'autre part, la possibilité offerte par les microprocesseurs de décharger le calculateur d'un certain nombre de tâches permet de gagner en temps sur une manipulation et autorise l'accomplissement de tâches en parallèles (mesure d'effort, calcul asservissement, mouvement du manipulateur).

Une prochaine étape en cours de développement permettra d'utiliser un langage littéral. Un compilateur générera à partir de ces primitives littérales, le code objet que nous connaissons actuellement. Cette nouvelle approche facilitera l'écriture d'un programme interprété, de plus de nouvelles fonctions viendront s'ajouter à celles déjà existantes pour simplifier les phases de mise au point.

CHAPITRE VI

EXPÉRIMENTATION

VI.1. INTRODUCTION

Le but de ce chapitre est d'illustrer de manière expérimentale, en tenant compte du matériel dont nous disposons, la versatilité des primitives que nous avons développées.

Tout d'abord le manipulateur Jaz doté de son organe sensoriel (capteur d'effort) se doit d'être bien calibré. En effet, l'asservissement est fondé sur la dualité force position, il est donc primordial de connaître avec précision l'orientation du repère dans lequel se fait la mesure des efforts et celui où sont générés les déplacements.
Nous avons déterminé cette orientation par une manipulation d'étalonnage automatique, décrite dans le première partie.

La deuxième partie est centrée sur l'exploitation des résultats obtenus sur l'analyse du contact ponctuel. Nous voyons de quelle manière nous pouvons reconnaître les points anguleux d'une pièce polyédrique.

Enfin, nous avons voulu dans la troisième partie mette en oeuvre une expérience plus générale tenant compte de l'analyse des forces (contact) et des déplacements. Le robot muni d'une pièce rectangulaire à son extrémité, doit après avoir pris contact sur un plan faisant obstacle à sa direction de déplacement, plaquer une arête sur ce plan et entourer le suivi de cette paroi en maintenant en permanence le contact avec elle, quelle que soit sa géométrie.

La réalisation pratique de ces manipulations s'est faite en utilisant le langage interprété décrit au chapitre V.

VI.2. CALIBRAGE

Nous avons vu au chapitre IV que nous disposions d'un manipulateur assez fruste, que nous avons muni d'un organe terminal jouant à la fois le rôle de poignet déformable et de capteur d'efforts.

Dans notre problématique la mesure précise des efforts prenant naissance sur le poignet quand il est en contact avec son environnement a relevé toute notre attention. En effet, tel que nous avons conçu l'asservissement, le déplacement du robot est lié à son action physique sur l'extérieur et donc aux forces et moments enregistrés par le capteur. Ces efforts sont exprimés dans un repère (RJAU) lié au capteur, puis dans un repère propre à l'asservissement (RCO) où nous pouvons faire leur analyse ; par contre, les déplacements calculés dans le repère RCO doivent être transcrits dans le repère de la tête du robot RTRO pour pouvoir être générés par la suite. Notre première tâche est donc de positionner avec précision le repère RJAU où sont mesurés les efforts par rapport au repère RTRO où sont exprimés les déplacements.

La modélisation du capteur nous a permis de situer RJAU avec précision Mais à la mise en place du poigent sur le robot, il est difficile de maîtriser parfaitement l'orientation à donner au capteur pour que les deux repères soient identiques en direction.

Nous avons donc pensé à une procédure d'étalonnage automatique de l'orientation du repère RJAU par rapport au repère RTRO, quand leur erreur d'orientation est faible.

VI.2.1. Principe du calibrage

Nous allons chercher à déterminer l'orientation de RJAU par rapport à RTRO, en utilisant la mesure des efforts associés aux déplacements du bras manipulateur.

L'hypothèse de départ est que le repère RJAU est identique en orientation au repère RTRO. Si cela est vrai, un déplacement dans la direction X^+ de RTRO entraînera au moment du contact dans un plan vertical orthogonal à cette direction, un effort de réaction ayant une composante unique en X. Par contre, s'il existe une mauvaise orientation, la mesure enregistrée dans le repère RJAU aura une composante suivant l'axe Y non négligeable.

Nous faisons plusieurs contacts successifs dans les conditions décrites plus haut en augmentant la force d'appui et nous obtenons un premier groupe de mesures qui contient l'information sur la direction de l'axe X du repère RJAU réel par rapport à la direction de l'axe X du repère RJAU supposé bien orienté.

La même procédure est utilisée pour la direction Y. Nous calculons à partir de ces mesures l'orientation réelle du repère RJAU par rapport à RTRO (voir Annexe).

VI.2.2. Réalisation pratique

Le dispositif expérimental adapté comprend outre le bras manipulateur et le poignet capteur d'effort, une pièce vissée à l'extrémité du poignet. Cette pièce est formée d'un cube de dimension 50x50x50 mm et d'une plaque carré fixée à sa partie inférieure de dimension légèrement supérieure 60x60 mm (voir Figure VI.1). La plaque métallique dépassant sous le cube, le contact sur le plan vertical extérieur se fera sur un des ces sommets.

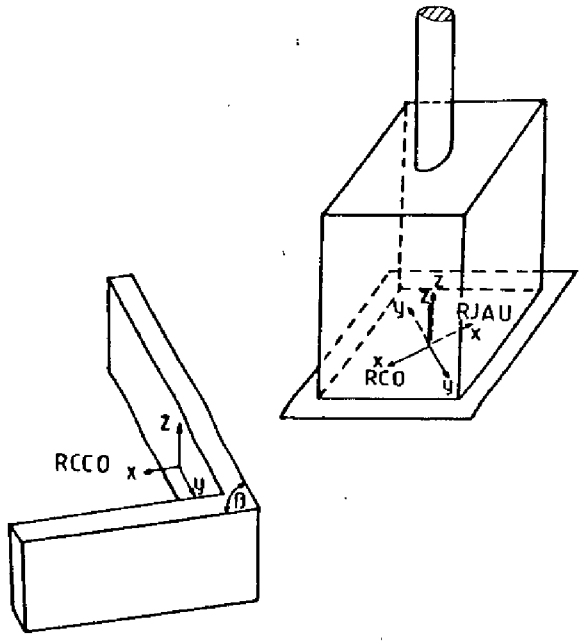


FIGURE VI.1 Schéma du dispositif expérimental

Nous devons définir les différents repères qui vont nous permettre de réaliser cette manipulation, ainsi que le mode d'asservissement choisi.

VI.2.2.1. *Description du graphe des repères*

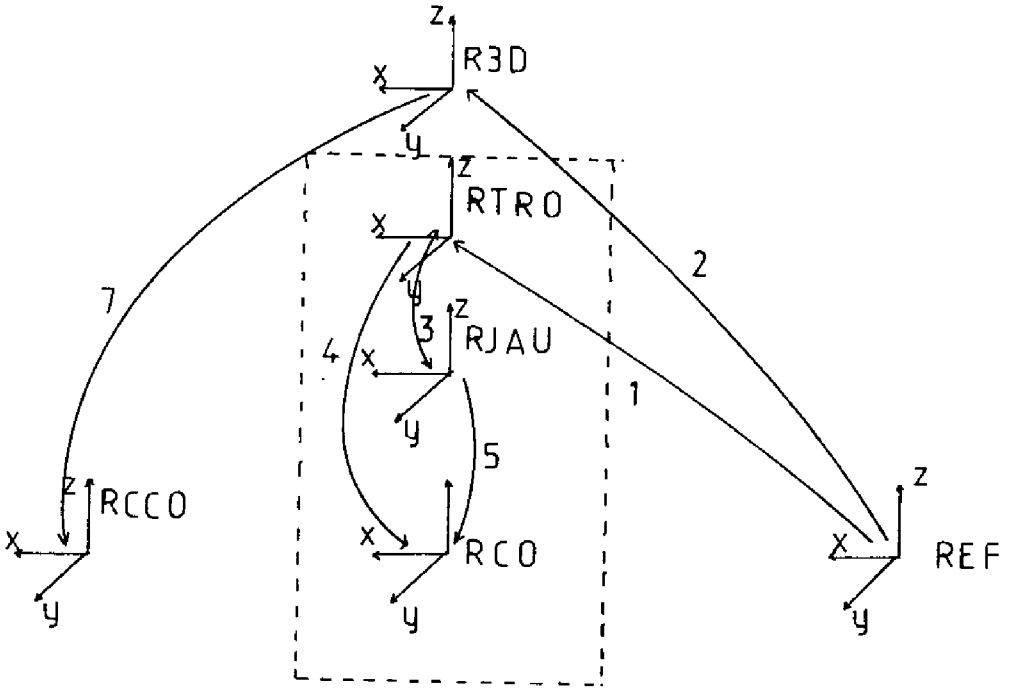


FIGURE VI.2 *Graphe des repères*

R3D = repère dans lequel se fait le tracé graphique

REF = repère de référence, butées arrières du robot

RTRO = repère de la tête du robot, donnant la position et l'orientation de l'organe terminal du bras par rapport à REF

RJAU = repère où sont mesurées les forces et les moments

RCO = repère dans lequel se font les calculs de l'asservissement forces ⟶ déplacements

RCCO = consigne en position et orientation, paramètre de l'asservissement

RTRO, RJAU et RCO sont situés sur le robot et donc mobiles. REF, R3D et RCO ne bougent pas. A chaque déplacement l'arc 1 exprimant RTRO/REF est réactualisé en tenant compte de la nouvelle position.

VI.2.2.2. Description de l'asservissement

Nous pouvons distinguer deux phases, la recherche du contact dans la direction positive de l'axe X, noté X^+, puis la recherche du contact suivant la direction positive de l'axe Y, notée Y^+.

a) Déplacement dans la direction X

Nous définissons la consigne en position (RCCO) dans la direction X^+ de RTRO. RCO étant identique à RTRO à une distance x^* située au-delà de la paroi du plan où doit se faire le contact.

Nous donnons aussi pour ce même axe une consigne de force F_X^*.

Les autres axes sont en surveillance (cf III.4.3), c'est-à-dire que si un obstacle imprévu est sur la trajectoire du robot ou bien s'il se produit une intervention extérieure, le robot se déplacera pour minimiser les efforts ayant pris naissance sur ces axes.

D'autre part, si le module de la force ou du moment mesurée dépassait une valeur limite (cf III.4.2), nous aurions arrêt du mouvement sur tous les axes avec envoi d'un message à l'opérateur.

Nous aurons donc un déplacement du bras manipulateur jusqu'au plan où se produira le contact avec un des coins de la pièce métallique vissée à l'extrémité du poignet. Quand la valeur de la force de consigne sera atteinte, il s'arrêtera et mémorisera la mesure. Après s'être dégager du plan par une translation d'axe X^-, la même démarche recommencera avec une force de consigne un peu plus élevée. Ces deux mesures seront représentatives de l'axe X.

b) Déplacement dans la direction Y

Partant de la position précédente, le bras manipulateur effectue une translation d'axe X^- pour se dégager du plan, puis une rotation de +90° autour de l'axe Z vertical.

La direction de l'axe Y^+ est donc en vis à vis du plan de contact utilisé précédemment.

Nous définissons les consignes de l'asservissement comme précédemment l'axe Y jouant alors le rôle de l'axe X.

L'asservissement ainsi défini va entraîner un déplacement en direction du plan, perpendiculairement à celui-ci. L'arrêt se produira quand l'angle de la pièce étant arrivé sur le plan, la force demandée sera satisfaite. Comme pour la direction X, cette manipulation est répétée avec une consigne en force plus élevée. Après enregistrement des mesures, le robot se dégage du plan et le programme spécifique à l'étalonnage, calcule l'orientation de RJAU par rapport à RTRO (Annexe).

Programme interprété permettant le calibrage en orientation au repère du capteur par rapport au repère du robot.

```
PROG SUR DISK= PETES2
I=  1 A=  1  MODU=  1
MES 1= INITIALISATION

I=  1 A=  1       32.00      21.00       0.00
I=  2 A=  4       32.00      31.00       1.00
I=  3 A=  7        8.00       5.00

I= 33 A= 96  MODU=  9
MES 9= PRISE CONTACT
I= 33 A= 96       15.00       7.00
I= 34 A= 98       15.00       8.00
I= 35 A=100       15.00      15.00
I= 36 A=102       15.00       3.00
I= 37 A=104       15.00       4.00
I= 38 A=106       15.00       5.00
I= 39 A=108       15.00       6.00
I= 40 A=110       32.00      36.00     -16.00
I= 41 A=113       15.00       2.00
I= 42 A=115  MODU= 10
MES10= CALCUL ASSIE PC.
I= 42 A=115        5.00       0.00      50.00
I= 43 A=118       32.00       2.00      -7.00
I= 44 A=121       11.00      -1.00      11.00
I= 45 A=124       11.00      -2.00      16.00
I= 46 A=127       10.00       0.00       0.00
I= 47 A=130        1.00
```

```
I= 56 A=168  MCDU= 11
MES11= CONTACT X
I= 56 A=168        32.00       6.00        1.00
I= 57 A=171        11.00      -5.00       14.00
I= 58 A=174        11.00      -4.00       13.00
I= 59 A=177        11.00      -3.00       12.00
MES11= CONTACT X
I= 60 A=180        19.00      11.00
I= 61 A=182        32.00      26.00        4.00
I= 62 A=185        32.00      51.00       -6.00
I= 63 A=188        19.00      23.00
I= 64 A=190        30.00      27.00      -15.00       0.00        0.00
I= 65 A=196        19.00      19.00
I= 66 A=198        19.00      17.00
I= 67 A=200        26.00       3.00
I= 68 A=202        24.00     -17.00        0.60       2.00        2.00       60
I= 69 A=208        12.00       3.00        0.00
I= 70 A=211         8.00      10.00


I= 71 A=213  MCDU= 12
MES12= CONTACT 1 EN Y
I= 71 A=213        32.00      26.00        4.00
I= 72 A=216        32.00      51.00       -6.00
I= 73 A=219        19.00      23.00
I= 74 A=221        30.00      27.00      -30.00       0.00        0.00        0
I= 75 A=227        30.00      27.00        0.00       0.00        0.00      -90
I= 76 A=233        19.00      19.00
I= 77 A=235        32.00      36.00      -16.00
MES12= CONTACT 1 EN Y
I= 78 A=238        19.00      12.00
I= 79 A=240        32.00      33.00        1.00
I= 80 A=243        26.00       3.00
I= 81 A=245        24.00     -17.00        2.00       0.40        2.00       60
I= 82 A=251        19.00      18.00
I= 83 A=253        12.00       4.00        0.00
I= 84 A=256         8.00      10.00


I= 85 A=258  MCDU= 13
MES13= CONTACT 2 EN Y
I= 85 A=258        32.00      26.00        4.00
I= 86 A=261        32.00      51.00       -6.00
I= 87 A=264        19.00      23.00
I= 88 A=266        30.00      27.00      -15.00       0.00        0.00        0.
MES13= CONTACT 2 EN Y
I= 89 A=272        19.00      13.00
I= 90 A=274        19.00      19.00
I= 91 A=276        19.00      18.00
I= 92 A=278        26.00       3.00
I= 93 A=280        24.00     -17.00        2.00       0.60        2.00       60.
I= 94 A=286        12.00       5.00        0.00
I= 95 A=289         8.00      10.00
```

```
I= 96 A=291  MODU= 14
MES14= ETALONNAGE
I= 96 A=291        19.00       14.00
I= 97 A=293        32.00       26.00        4.00
I= 98 A=296        32.00       51.00       -6.00
I= 99 A=299        19.00       23.00
I=100 A=301        30.00       27.00      -15.00        0.00        0.00        0.
I=101 A=307        26.00        4.00
I=102 A=309        23.00     1902.00        4.00        8.00        3.00
I=103 A=314        19.00       21.00
I=104 A=316        32.00       36.00      -10.00
I=105 A=319        32.00       55.00        1.00
I=106 A=322        32.00       30.00       16.00
I=107 A=325         1.00
     Sous-programmes appelés par P.P.

I=  4 A=  9  MODU=  2
MES 2= PAZ PLAC 3 4 5
I=  4 A=  9        12.00        3.00        1.00
I=  5 A= 12        12.00        4.00        1.00
I=  6 A= 15        12.00        5.00        1.00
I=  7 A= 18        32.00       33.00        1.00
I=  8 A= 21        16.00
I=  9 A= 22  MODU=  3
MES 3= NB3D/NBEF
I=  9 A= 22        26.00        4.00
I= 10 A= 24        23.00     1902.00        4.00       20.00       13.00
I= 11 A= 29        19.00       21.00
I= 12 A= 31        23.00     1902.00        3.00       20.00        4.00
I= 13 A= 36        19.00       21.00
I= 14 A= 38        23.00     1902.00        4.00       20.00        3.00
I= 15 A= 43        19.00       21.00
I= 16 A= 45        32.00       46.00      -20.00
I= 17 A= 48        16.00
I= 18 A= 49  MODU=  4
MES 4= NBJAU/NBEF ET NB
I= 18 A= 49        26.00        4.00
I= 19 A= 51        23.00     1902.00        4.00        6.00       15.00
I= 20 A= 56        23.00    -1905.00        0.00        0.00        0.00
I= 21 A= 61        23.00    -1911.00        0.00        0.00        1.00
I= 22 A= 66        23.00    -1917.00        1.00        0.00        0.00
I= 23 A= 71        19.00       21.00
I= 24 A= 73        16.00
I= 25 A= 74  MODU=  5
MES 5= NBCC ET NBCCC
I= 25 A= 74        26.00        4.00
I= 26 A= 76        23.00     1902.00        4.00        8.00       13.00
I= 27 A= 81        19.00       21.00
I= 28 A= 83        23.00     1902.00        8.00        6.00        4.00
I= 29 A= 88        19.00       21.00
I= 30 A= 90  MODU= 17
MES17= CLCI?
I= 30 A= 90        32.00       57.00       40.00
I= 31 A= 93        19.00       19.00
I= 32 A= 95        16.00
```

```
I= 48 A=131  MODU=  6
MES 6= INIT FC,POS
I= 48 A=131       26.00        3.00
I= 49 A=133       22.00       -1.00        3.00      100.00
I= 50 A=137       24.00       -5.00        2.00        2.00        2.00        2.0
I= 51 A=143       24.00      -17.00        0.40        2.00        2.00       60.0
I= 52 A=149       24.00      -29.00        0.05        0.05        0.05        0.0
I= 53 A=155       24.00      -41.00        1.50        1.50        1.50       50.0
I= 54 A=161       24.00      -53.00        5.00        5.00        5.00        5.0
I= 55 A=167       16.00

I=107 A=326  MODU= 15
MES15= ACTUP,ZEROS CALC
I=108 A=326       32.00        1.00        7.00
I=109 A=329       32.00       23.00        1.00

I=110 A=332       16.00

I=111 A=333  MODU= 16
MES16= PLOCAGE ASSEZ FC
I=111 A=333       19.00       16.00
I=112 A=335       15.00       15.00
I=113 A=337        1.00
I=114 A=338  MODU= 18
MES18= ARCCO EN Y
I=114 A=338       32.00       58.00       40.00
I=115 A=341       15.00       19.00
I=116 A=343       16.00
I=117 A=344  MODU= 19
MES19= ARCCO ET AREF
I=117 A=344       26.00        4.00
I=118 A=346       23.00     1902.00       20.00        7.00        4.00
I=119 A=351       15.00       21.00
I=120 A=353       23.00     1902.00        8.00        7.00        3.00
I=121 A=358       15.00       21.00
I=122 A=360       16.00
I=123 A=361  MODU=  7
MES 7= ECHELON DEPLACEM
I=123 A=361       26.00        7.00
I=124 A=363       22.00       -1.00       10.00       10.00
I=125 A=367       21.00        5.00       20.00
I=126 A=370       23.00       -6.00     -200.00      200.00     -160.00
I=127 A=375       24.00      -20.00       -1.00        0.00       -0.50       -0.5
I=128 A=381       24.00      -26.00        0.00        1.00        0.00        0.0
I=129 A=387       23.00      -36.00        0.00        0.00        0.00
I=130 A=392       32.00       41.00        7.00
I=131 A=395       16.00
I=132 A=396  MODU=  8
MES 8= INIT RCFCT
I=132 A=396       30.00        6.00        0.00
I=133 A=399       30.00        7.00        1.00
I=134 A=402       30.00        1.00      100.00      530.00     1120.00        0.0
MES32= RCFCTEN PLACE
I=135 A=408       19.00       32.00
I=136 A=410       16.00
```

```
I=137 A=411   MODU= 21
MES21= PIEGE PROG INTER
I=137 A=411        32.00       37.00        0.00
I=138 A=414        16.00
I=139 A=41F   MODU= 22
I=139 A=41F        26.00        4.00
I=140 A=417        25.00     -827.00        1.00        6.00        6.00
I=141 A=423        16.00
I=142 A=424   MODU= 23
I=142 A=424        26.00        3.00
I=143 A=426        25.00     -265.00        1.00        6.00        6.00
I=144 A=432        16.00
I=145 A=432        99.00
   Fin de programme interprété
```

VI.3. RECONNAISSANCE DES ANGLES D'UNE PIECE A L'AIDE DES PROPRIETES DU CONTACT PONCTUEL

Nous ne reviendrons pas sur les propriétés du contact ponctuel décrites au chapitre III. Pour illustrer cette analyse, nous avons pensé qu'une expérience consistant à déterminer les coordonnées des points anguleux d'une pièce, à l'aide de l'information recueillie sur les efforts appliqués en ces points, confirmerait les résultats théoriques.

Le dispositif expérimental, adopté est identique à celui utilisé pour le calibrage (Figure VI.1).

VI.3.1. Procédure d'exploration d'une partie de la géométrie de la pièce

Cette procédure doit nous permettre de déterminer les points anguleux de la pièce, situés dans un même plan. Nous distinguerons deux étapes principales, la détermination des coordonnées d'un point et l'enchaînement pour la recherche et la détermination des autres points.

VI.3.1.1. Détermination des coordonnées d'un point de la pièce

VI.3.1.1.1. Recherche du contact

A l'instant initial, le robot se trouve dans une position connue par rapport au repère de référence REF. La pièce (plaque métallique) saisie par le robot est inconnue, mais nous avons une idée du domaine spatial, borné supérieurement, qu'elle est susceptible d'occuper.

Il faut donc à partir de cette position initiale, amener la pièce au contact d'un plan, de manière à pouvoir déterminer les coordonnées du point de contact.

Nous devons donc choisir une direction de déplacement et un test d'arrêt quand nous mesurons un effort. Ceci revient à définir un asservissement en force et position.

a) Choix de l'asservissement

A priori, il n'y a pas de raisons pour privilégier une direction de déplacement plutôt qu'une autre. Néanmoins, pour des commodités expérimentales, connaissant la position du plan de contact, nous choisirons cette direction dans un plan horizontal orthogonal à celui-ci.

1) position

Le repère RCO dans lequel se fait l'asservissement a son centre confondu avec le centre du repère RJAU.

La direction de son axe X est orthogonal au plan de contact. L'axe Z reste vertical.

La consigne à atteindre est matérialisée par le repère RCCO qui est défini identique à RCO à une translation près : X_C dans la direction X

La figure VI.3 illustre la position des repères pour cette première phase.

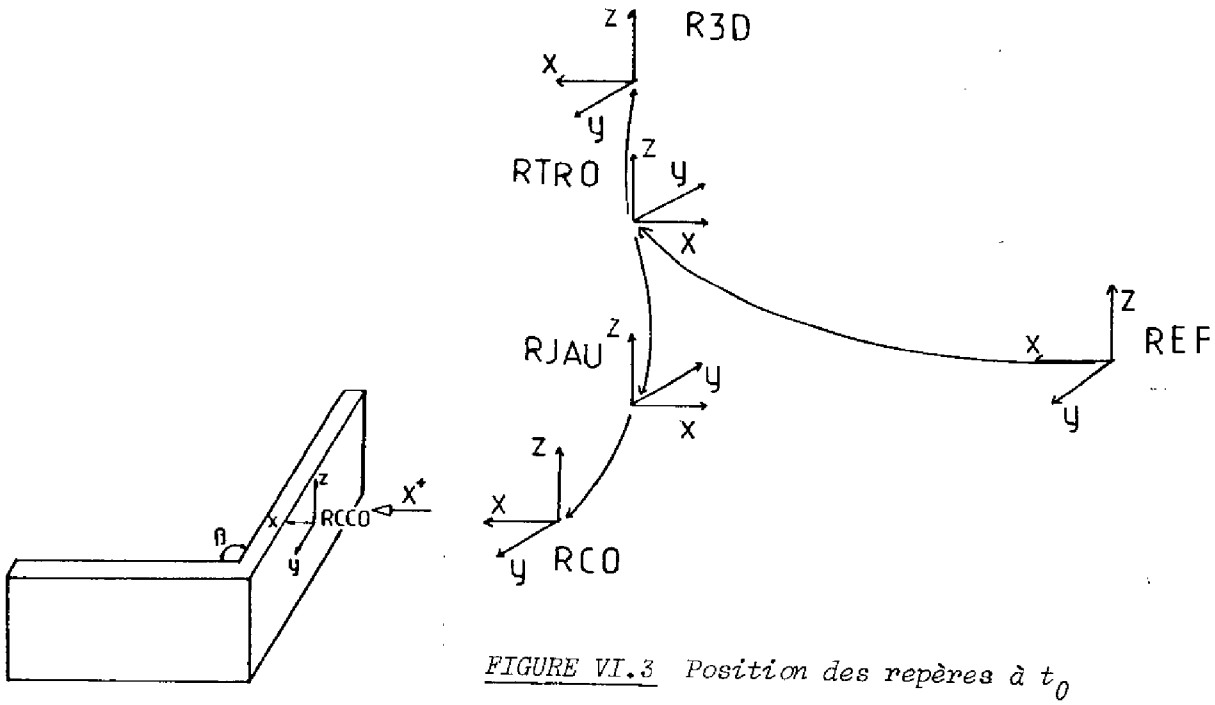


FIGURE VI.3 Position des repères à t_0

2) consigne de force

La pièce portée par le robot, se déplaçant dans la direction X de RCO, nous devons donner une consigne en force relative à cet axe là, pour s'arrêter au moment de la prise de contact sur le plan. Nous aurons donc une consigne F_X^+ suivant X égal à -0.6 DN , tandis que les axes Y et Z ainsi que la rotation autour de Z, seront mis en surveillance.

La priorité étant donnée à la première consigne atteinte, le robot s'arrêtera quand il aura détecté une force de -0.6 DN dans la direction X.

VI.3.1.1.2. Détermination automatique des coordonnées du point de contact

En utilisant les résultats obtenus au chapitre III, nous devons pour parvenir à la détermination des coordonnées du point de contact, obtenir une information conséquente au niveau du point d'appui.

Le robot doit donc produire des petits déplacements tout en gardant le même point de la pièce en contact avec le plan. Il enregistrera les efforts produits quand ils satisferont à des consignes prédéterminées.

a) Choix des asservissements

Nous avons jugé que quatre mesures, si celles-ci sont assez discriminantes, peuvent nous permettre d'obtenir les coordonnées du point d'application des efforts.

C'est-à-dire que les angles faits entre eux par les vecteurs de force issus des tests relatifs à un même point devront être suffisamment grands pour ne pas entraîner d'erreur sur le calcul des coordonnées du point (cf III.2.3).

Le robot aura donc quatre consignes successives à satisfaire. Chaque fois que l'une d'elles sera atteinte, il mémorisera les efforts mesurés, puis enchaînera sur la consigne suivante.

Pour avoir des forces non seulement dans la direction X mais aussi en Y et Z, le bras manipulateur devra déplacer la pièce dans les directions Y et Z tout en maintenant une certaine force de contact en X. Nous aurons donc pour chaque mode d'asservissement des consignes de position et des consignes de force.

Nous détaillons ci-dessous les quatre modules d'asservissement que le robot devra réaliser.

Module 1

Il correspond à la prise de contact décrite précédemment, la mesure est acquise quand la consigne en force FCS = -0.5 DN est satisfaite.

Module 2

Nous retrouvons la dualité force et position.

1) position

La consigne de position entraînera un déplacement depuis le point de contact, dans les directions X^+, Y^+, Z^+.

2) force

Trois consignes de forces doivent être satisfaites pour que la mesure soit acceptée par le robot :

$F_X^* = 0.65 \pm 0.05$ DN

$F_Y^* = 0.15 \pm 0.07$ DN

$F_Z^* = 0.2 \pm 0.07$ DN

Le moment MZ est en surveillance.

Module 3

1) position

Cette fois nous donnons une consigne qui entraînera un déplacement dans les directions X^+,Y^+ et Z^-.

2) force

$F_X^* = 0.75 \pm 0.05$ DN

$F_Y^* = 0.10 \pm 0.07$ DN

$F_Z^* = 0.25 \pm 0.07$ DN

MZ est en surveillance.

Module 4

1) position

Le déplacement se fera suivant X^+, Y^- et Z^+.

2) force

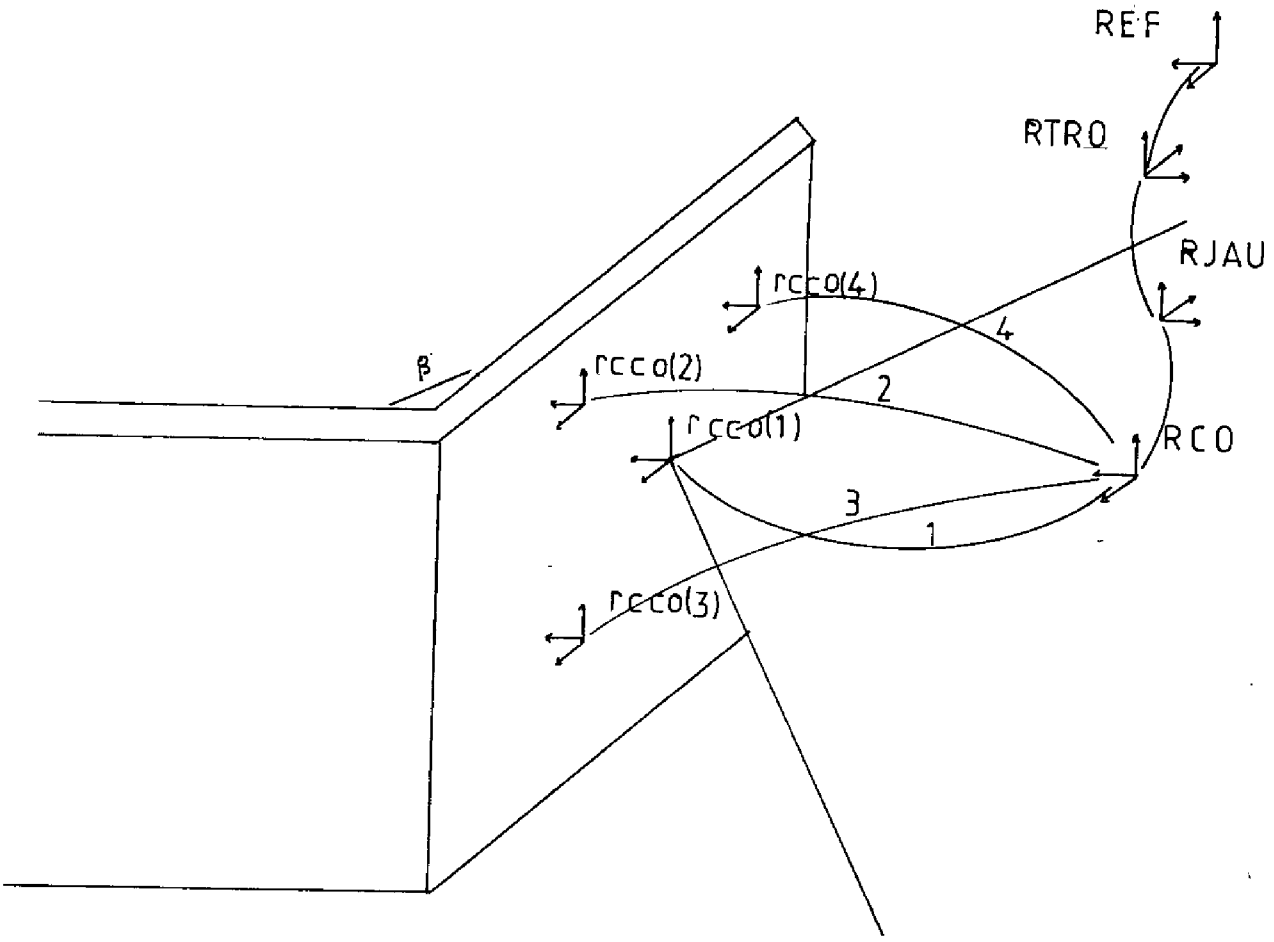
$F_X^* = 0.8 \pm 0.05$ DN

$F_Y^* = -0.15 \pm 0.07$ DN

$F_Z^* = 0.15 \pm 0.07$ DN

Il est à noter que pour les modules 2,3,4, le repère RCCO définissant la consigne de position, est lié au repère RCO. Ce qui fait que la consigne en position n'est jamais atteinte, et le robot s'arrête quand la consigne en force sur chaque axe est conforme aux valeurs données.

FIGURE VI.4 Détermination des coordonnées du point de contact



VI.3.1.2. Détermination des points suivants

Dans notre approche expérimentale, nous utilisons toujours le même plan pour engendrer les efforts, nous permettant de calculer les coordonnées des points de contact.

Après la détermination d'un premier point, le robot se trouve dans une position où il lui est impossible de découvrir un nouveau point. Nous allons donc définir une procédure qui l'autorisera à chercher un autre angle de la pièce en utilisant l'algorithme décrit au paragraphe précédent.

a) Recherche d'un autre point

Cette recherche est simple et se décompose en trois phases.

phase 1 : retrait du robot du plan de contact par une translation d'axe X, négative

phase 2 : rotation du robot d'un angle ϕ autour de l'axe vertical. Dans notre manipulation nous avons pris $\phi = 90°$

phase 3 : redéfinition de RCO, repère de l'asservissement par rapport à RJAU et RTRO, en tenant compte des déplacements précédents. Ceci afin d'utiliser le même plan pour le contact.

Les phases 1,2 et 3 étant réalisées, le robot reexécute la procédure relative à la détermination des coordonnées d'un point (cf VI.3.1.1).

VI.3.2. Résultats expérimentaux

La manipulation décrite au paragraphe précédent a été programmée à l'aide du langage interprété.

Le robot a trouvé les coordonnées des quatre points d'angle de la pièce avec une précision de l'ordre de ± 3 mm. Nous pouvons donc envisager d'apporter une contribution à la reconnaissance de forme à l'aide des informations tactiles fournies par un capteur de force. Pour pouvoir exploiter ces résultats, nous devrions bien sûr faire appel à des stratégies plus intelligentes où le robot irait consulter en mémoire les modèles des pièces à reconnaître, puis les comparerait aux pièces qu'il étudie.

Notre but n'est pas ici de faire de la reconnaissance de forme proprement dite, mais simplement de montrer comment un robot "aveugle" peut avoir une perception de son environnement. En effet, une autre application pourrait être de reconnaître non pas une pièce tenue par le robot, mais des obstacles situés sur la trajectoire du robot. Le contournement par le suivi de ces obstacles serait alors possible sans risque de détérioration pour le bras manipulateur.

Nous donnons ci-dessous le programme interprété que nous avons défini.

L'opérateur demande l'exécution du module 1 (8. 1.) et le robot enchaîne sur l'exécution de tous les autres modules. L'expérimentation se termine quand le robot a effectué une rotation de 270° à partir de sa position initiale de premier contact.

```
PROG SUR DICK= PERFET
I=  1 A=  1  MODU=  1
MES 1= INITIALISATION
I=  1 A=  1      32.00      21.00       0.00
I=  2 A=  4      32.00      31.00       1.00
I=  3 A=  7       8.00      13.00

I= 57 A=185  MODU= 13
MES13= RECHERCHE CONTAC
I= 57 A=185      15.00       2.00
I= 58 A=187      15.00       3.00
I= 59 A=189      15.00       4.00
I= 60 A=191      15.00       5.00
I= 61 A=193      15.00       6.00
I= 62 A=195      15.00       7.00
I= 63 A=197      15.00       8.00
I= 64 A=199      15.00       9.00
I= 65 A=201      15.00      12.00
I= 66 A=203       8.00      15.00

I= 72 A=218  MODU= 15
MES15= CONTACT 1
I= 72 A=218      15.00      14.00
I= 73 A=220      15.00      24.00
MES15= CONTACT 1
I= 74 A=222      15.00      15.00
I= 75 A=224      15.00      21.00
I= 76 A=226      26.00       3.00
I= 77 A=228      23.00      -5.00       0.80       1.50       1.00
I= 78 A=233      23.00     -17.00       0.65       0.15       0.20
I= 79 A=238       8.00      16.00
```

```
I= 90 A=240  MODU= 16
MES16= CONTACT 2
I= 90 A=240       15.00       14.00
MES16= CONTACT 2
I= 91 A=242       19.00       16.00
I= 92 A=244       15.00       24.00
I= 93 A=246       26.00        4.00
I= 94 A=248       24.00    -1905.00        5.00        5.00       -5.00        0.
I= 95 A=254       15.00       28.00
I= 96 A=256       26.00        3.00
I= 97 A=258       23.00      -17.00        0.75        0.10       -0.25
I= 98 A=263        8.00       17.00

I= 99 A=265  MODU= 17
MES17= CONTACT 3
I= 99 A=265       15.00       14.00
MES17= CONTACT 3
I= 90 A=267       15.00       17.00
I= 91 A=269       15.00       24.00
I= 92 A=271       26.00        4.00
I= 93 A=273       24.00    -1905.00        5.00       -5.00        5.00        0.
I= 94 A=279       15.00       28.00
I= 95 A=281       26.00        3.00
I= 96 A=283       23.00      -17.00        0.80       -0.15        0.15
I= 97 A=288       15.00       14.00
MES29= CONTACT 4

I= 98 A=290       15.00       29.00
I= 99 A=292       15.00       24.00
I=100 A=294        8.00       18.00

I=101 A=296  MODU= 18
MES19= COORDONNEE POINT
I=101 A=296       32.00       52.00        2.00
I=107 A=299       32.00       53.00        1.00
I=103 A=302       11.00       -3.00       25.00
I=104 A=305       15.00       22.00
I=105 A=307       15.00        5.00
I=106 A=309       11.00       -5.00       20.00
I=107 A=312       11.00       -4.00       19.00
I=108 A=315       15.00       23.00
I=109 A=317       21.00    -1917.00       -1.57
I=110 A=320       15.00       11.00
I=111 A=322       15.00       26.00
I=112 A=324       15.00        9.00
I=113 A=326       15.00       12.00
I=114 A=328       12.00        4.00        0.00
I=115 A=331        8.00       15.00
```

```
I=116 A=330  MODU= 19
MES19= POINT 3
I=116 A=332        15.00       23.00
I=117 A=335        21.00    -1917.00        0.00
I=118 A=338        15.00       11.00
I=119 A=340        15.00       26.00
I=120 A=342        15.00        9.00
I=121 A=344        15.00       12.00
I=122 A=346        12.00        5.00        0.00
I=123 A=349         8.00       15.00
I=124 A=351  MODU= 20
MES20= POINT 4
I=124 A=351        15.00       23.00
I=125 A=353        21.00    -1917.00        1.57
I=126 A=356        15.00       11.00
I=127 A=358        15.00       26.00
I=128 A=360        15.00        9.00
I=129 A=362        15.00       12.00
I=130 A=364        12.00        3.00        0.00
I=131 A=367         8.00       15.00

I=140 A=422  MODU= 25
MES25= TRACE POINT
I=140 A=427        32.00       43.00        6.00
I=140 A=425         1.00
```

Les modules suivants se terminant par 16 sont des sous-programmes.

```
I=  4 A=  9  MODU=  2
MES 2= ET BENDEZ-VOUS
I=  4 A=  9        30.00        6.00        0.00
I=  5 A= 12        30.00        7.00        1.00
I=  6 A= 15        30.00        1.00      110.00      530.00     1100.00     180.
MES 2= ET BENDEZ-VOUS
I=  7 A= 21        19.00        2.00
I=  8 A= 23        16.00
I=  9 A= 24  MODU=  3
MES 3= FAZ FLAG
I=  9 A= 24        12.00        3.00        1.00
I= 10 A= 27        12.00        4.00        1.00
I= 11 A= 30        12.00        5.00        1.00
I= 12 A= 33        16.00
I= 13 A= 34  MODU=  4
MES 4= INIT TRACE
I= 13 A= 34        26.00        7.00
I= 14 A= 36        22.00       -1.00       10.00       10.00
I= 15 A= 40        21.00        5.00       20.00
I= 16 A= 43        23.00       -6.00     -150.00      150.00     -120.00
I= 17 A= 48        24.00      -20.00       -1.00        0.00       -0.50       -0.
I= 18 A= 54        24.00      -28.00        0.00        1.00        0.00        0.
I= 19 A= 60        23.00      -36.00        0.00        0.00        0.00
I= 20 A= 65        32.00       41.00        7.00
I= 21 A= 68        16.00
I= 22 A= 69  MODU=  5
MES 5= ZEROS CALCES,ACT
I= 22 A= 69        32.00       23.00        1.00
I= 23 A= 72        32.00        1.00        7.00
I= 24 A= 75        16.00
```

```
I= 25 A= 76  MODU=  6
MES 6= MESAU/METRO
I= 25 A= 76        26.00        4.00
I= 26 A= 78        23.00     1902.00        4.00        6.00       15.00
I= 27 A= 83        23.00    -1905.00        0.00        0.00        0.00
I= 28 A= 88        23.00    -1911.00        0.00        0.00        1.00
I= 29 A= 93        23.00    -1917.00        1.00       -0.05        0.00
I= 30 A= 98        15.00       11.00
I= 31 A=100        16.00
I= 32 A=101  MODU=  7
MES 7= MESD/METRO
I= 32 A=101        26.00        4.00
I= 33 A=103        23.00     1902.00        4.00       20.00       13.00
I= 34 A=108        15.00       11.00
I= 35 A=110        16.00
I= 36 A=111  MODU=  8
MES 8= MRCC/MESAU
I= 36 A=111        26.00        4.00
I= 37 A=113        23.00     1902.00        4.00        8.00        2.00
I= 38 A=118        23.00    -1905.00        0.00        0.00        1.00
I= 39 A=123        24.00    -1911.00        0.00        0.00        0.00      -3.
I= 40 A=129        15.00       11.00
I= 41 A=131        23.00     1902.00        8.00        6.00        4.00
I= 42 A=136        15.00       11.00
I= 43 A=138        16.00
I= 44 A=139  MODU=  9
MES 9= MRCCC/MRCC
I= 44 A=139        32.00       57.00       60.00
I= 45 A=142        16.00

I= 47 A=144  MODU= 11
MES11=
I= 47 A=144        32.00       37.00        0.00
I= 48 A=147        16.00
I= 49 A=148  MODU= 12
MES12= DEF. VAR. ASSER.
I= 49 A=148        26.00        3.00
I= 50 A=150        22.00       -1.00        3.00      150.00
I= 51 A=154        24.00       -5.00        1.00        2.00        2.00        2.
I= 52 A=160        24.00      -17.00        0.50        2.00        2.00      100.
I= 53 A=166        24.00      -29.00        0.05        0.07        0.07        5.
I= 54 A=172        24.00      -41.00        1.50        1.50        1.50       90.
I= 55 A=178        24.00      -53.00        5.00        5.00        5.00        5.
I= 56 A=184        16.00

I= 67 A=205  MODU= 14
MES14= BOUCLE ASSERVISS
I= 67 A=205         9.00        0.00      150.00
I= 68 A=208        32.00        2.00       -7.00
I= 69 A=211        11.00       -1.00       26.00
I= 70 A=214        10.00        0.00        0.00
I= 71 A=217        16.00
```

```
T=132 A=369   MODL= 21
MES21= MACCC/MACC
T=132 A=369        26.00        4.00
T=133 A=371        24.00    -1905.00        5.00        5.00        5.00        0.
T=134 A=377        32.00       54.00        1.00
T=135 A=380        16.00
T=136 A=381   MODL= 22
MES22= DEB. ECRCT
T=136 A=381        30.00       27.00      -20.00        0.00        0.00        0.
T=137 A=387        30.00       27.00        0.00        0.00        0.00      -90.
T=138 A=393        16.00
T=139 A=394   MODL= 23
MES23= MACO/MEJAL
T=139 A=394        26.00        4.00
T=140 A=396        23.00     1902.00        6.00        8.00        2.00
T=141 A=401        23.00    -1905.00        0.00        0.00        1.00
T=142 A=406        23.00    -1911.00        0.00        0.00        0.00
T=143 A=411        16.00
T=144 A=412   MODL= 24
MES24= MESURE EFFORTS
T=144 A=412        32.00        6.00        1.00
T=145 A=415        32.00       26.00        4.00
T=146 A=418        32.00       51.00       -6.00
T=147 A=421        16.00

T=150 A=426   MODL= 26

MES26= MACO/MATEO CHEMI
T=150 A=426        26.00        4.00
T=151 A=428        23.00     1902.00        4.00        8.00        4.00
T=152 A=432        15.00       11.00
T=153 A=435        16.00
T=154 A=436   MODL= 28
MES28=
T=154 A=436        32.00       54.00        1.00
T=155 A=439        16.00
T=156 A=440        99.00
```

VI.4. SUIVI D'UNE PAROI PAR L'EXTREMITE TERMINALE DU ROBOT

Nous avons adopté le même dispositif expérimental pour l'ensemble bras manipulateur - poignet - pièce que celui utilisé pour le calibrage. L'obstacle sur lequel la pièce doit être mise en contact est un assemblage de deux plans verticaux orthogonaux entre eux. Le robot a face à sa direction X la partie convexe de l'obstacle. Nous pouvons illustrer ce dispositif par la figure VI.5.

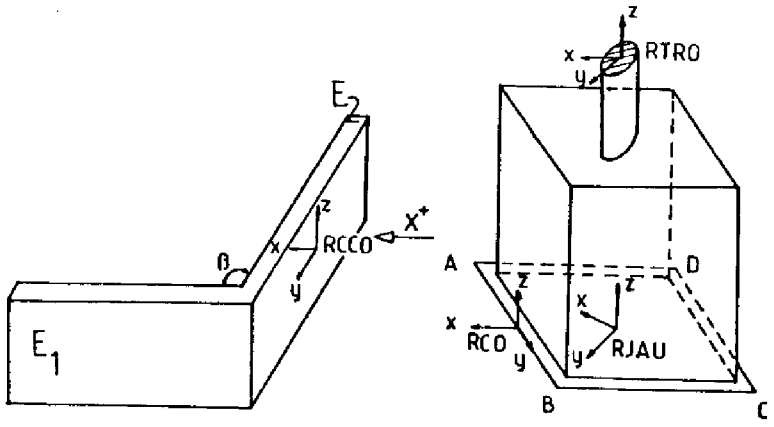


FIGURE VI.5 *Schéma du dispositif expérimental*

Nous pouvons distinguer trois phases dans cette manipulation. Premièrement la prise de contact sur l'obstacle ; le déplacement se fait dans la direction X^+, qui fait vis à vis à l'obstacle.

Deuxièmement la rotation du poignet, conditionnée aux forces et moments produits et qui doit plaquer l'arête AB de la pièce contre le plan.

La troisième étape est le suivi de la paroi de l'obstacle par l'arête de la pièce. Quand celle-ci arrive sur un angle, elle pivote tout en maintenant le contact et se replaque contre la paroi pour poursuivre le suivi.

Nous évoluons donc suivant les degrés de liberté de la liaison créée.

La manipulation s'achève lorsque la pièce a deux de ses faces (AB et BC) en contact avec l'obstacle (E_1 et E_2).

VI.4.1. Analyse des forces au contact

Nous allons examiner successivement pour les différentes étapes de la manipulation, les efforts de réaction produits par l'action du bras manipulateur.

1) Contact point sur plan

Au moment de la mise en contact, au point A la réaction du plan à l'avancement du robot engendrera une force F ; cette force aura une direction comprise dans le cône de frottement délimité par l'angle formé par les arêtes AB et AD et un plan vertical orthogonal au plan où se fait le contact.

Dans notre approche, le plan de contact est vertical et le déplacement est perpendiculaire à ce plan. La force de réaction va donc se trouver dans le plan X Y (cf Figure VI.6.a).

Le moment exprimé dans RJAU est voisin de zéro ; il n'est pas nul car le poignet se déforme et le support de la force ne passe pas rigoureusement par le centre du repère.

Par contre, en un point situé sur le milieu de l'arête AB, le moment aura une valeur non nulle non négligeable (cf Figure VI.6.b).

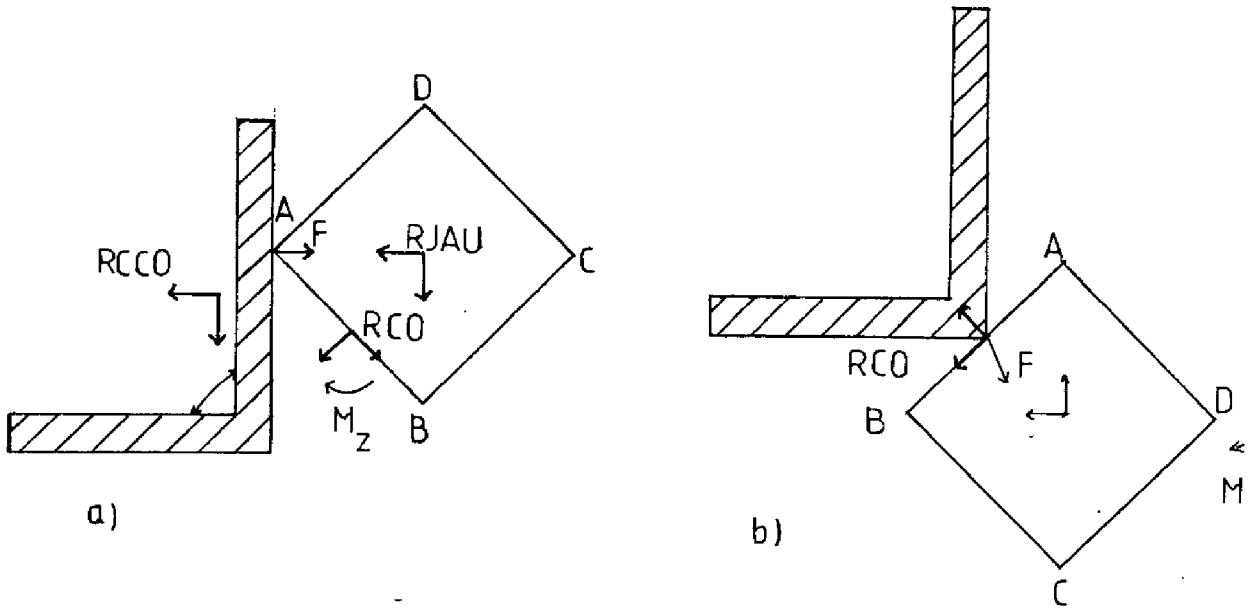


FIGURE VI.6 *Contact ponctuel a) point sur plan*
b) arête sur arête

2) Contact plan sur plan

Après rotation,la pièce se trouve être en contact sur une arête contre le plan précédent.

L'effort produit par le manipulateur pour maintenir la pièce en place induit une force de réaction. La résultante de cette force passe théoriquement par le milieu de l'arête AB, et donc par le centre du repère RCO. Dans ce référentiel, le moment autour de l'axe vertical Z est nul.

Pratiquement, la plaque et le poignet se déformant, nous avons apparition d'un moment en Z_j mais de faible valeur.

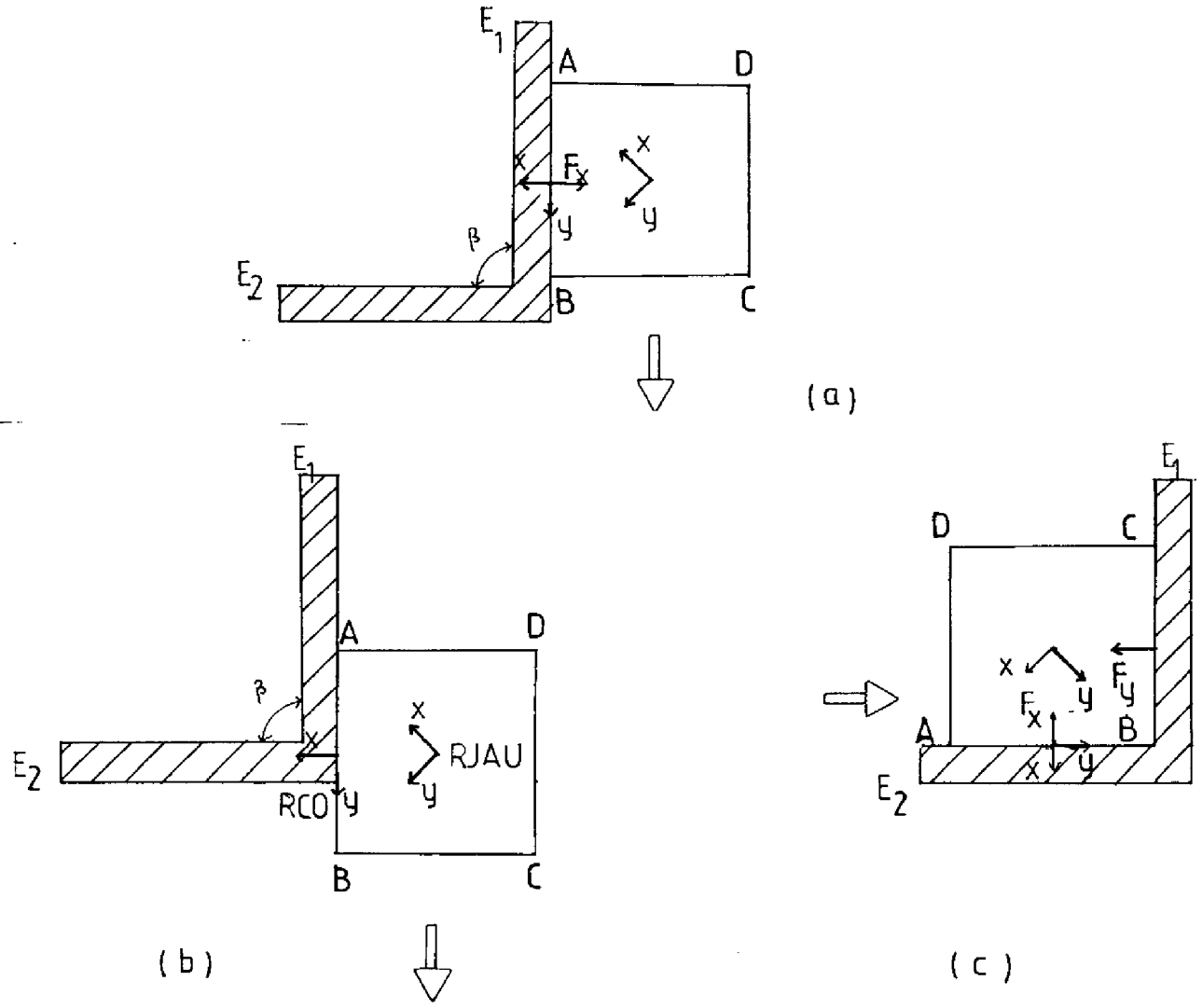


FIGURE VI.7 Contact plan sur plan

Sur la figure 7.b, il y a augmentation du moment quand le milieu de l'arête AB commence à échapper du plan. La force dans la direction X^+ étant toujours maintenue constante.
C'est la valeur de ce moment qui entraînera la rotation comme nous le verrons plus tard.

Sur la figure C, la pièce est en contact sur deux arêtes avec les plans E_1 et E_2. Dans le repère RCO, la force issue du plan E_2 sera dans la direction X^- et n'entraînera pas de composante en M_Z. Par contre, la force de réaction du plan E_1 sur BC fera apparaître une composante en MZ tel que :

$$M_Z = Fy \cdot X = (-BC/2) \times (-|Fy|) = \frac{Fy}{2} \times (BC)$$

Nous allons voir dans le paragraphe suivant comment nous exploitons ces propriétés.

VI.4.2. *Choix des modes d'asservissement*

Pour que le manipulateur puisse en fonction des efforts qu'il mesure, quand il est au contact de l'obstacle, plaquer l'arête contre cet obstacle et le suivre en maintenant une force d'appui constante ; il est nécessaire qu'il travaille suivant plusieurs modes d'asservissement, tout en étant capable de s'adapter à une situation imprévue.

Nous définirons donc deux phases de travail :

- l'une qui consiste à venir sur le plan E_1 et à plaquer l'arête AB,
- l'autre qui, à partir de cette position, va permettre au robot de suivre les contours de cet obstacle jusqu'à détection d'une consigne d'arrêt.

VI.4.2.1. Phase 1 = approche

a) Définition des repères

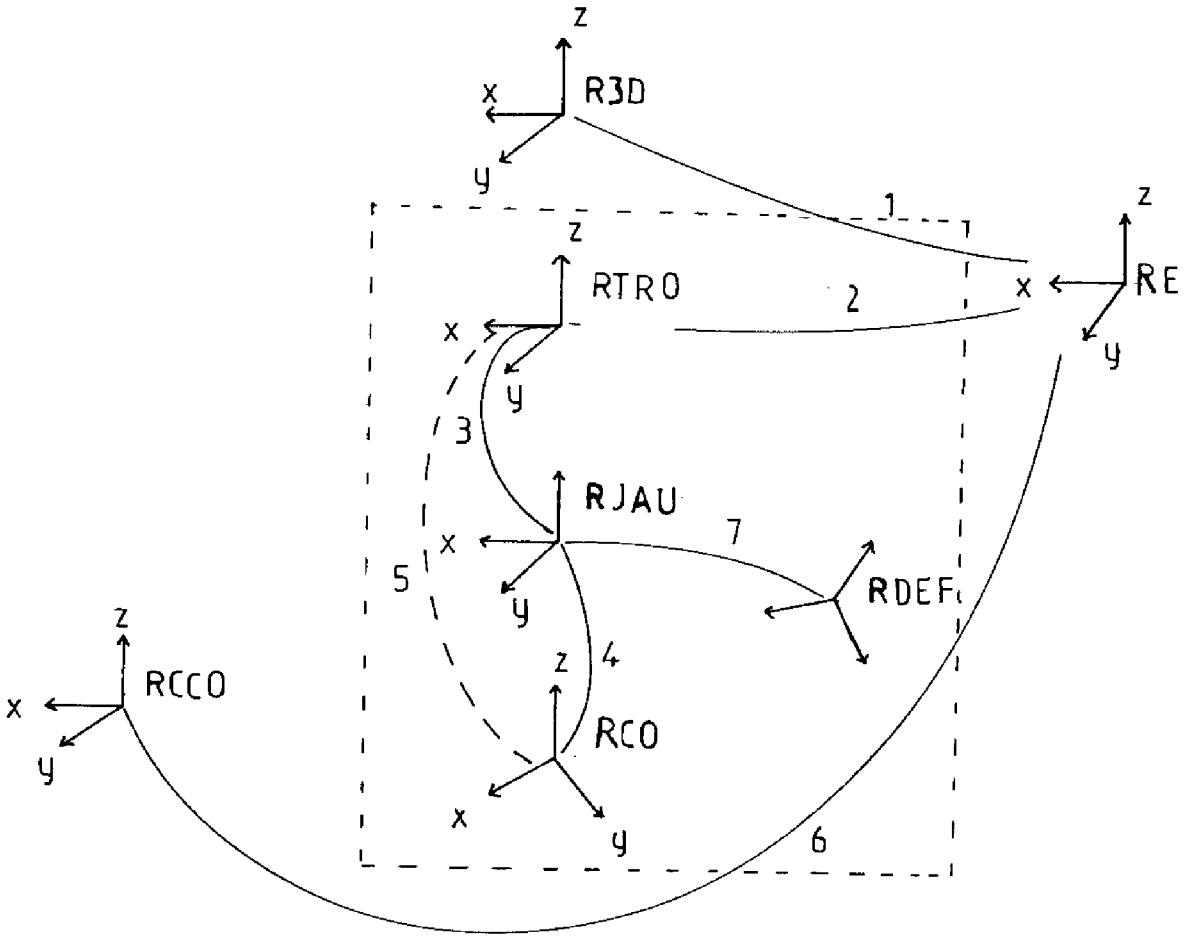


FIGURE VI.8 Graphe des repères

Le repère R3D représente le repère dans lequel sera fait le tracé, il es lié à la référence. A l'instant initial, il est confondu avec RTRO, qui représente la position et l'orientation du robot par rapport à sa référence REF.

Le repère RJAU est défini par rapport à RTRO conformément au résultat du calibrage (voir VI.2.).

Le repère RCO, dans lequel sont faits les calculs de l'asservissement est positionné au milieu d'une des arêtes (AB) de la pièce.
Sa direction X est perpendiculaire à AB et dirigé vers l'extérieur, la direction Y est tangente à la direction de l'arête.

Le repère RCCO représente la consigne en position, nous le définissons au-delà de la paroi à atteindre, ses axes X et Y étant orientés comme ceux du repère RJAU.

Le repère RDEF est le repère RJAU déformé sous l'action des efforts dus au contact. Le calcul de cette déformation est prise en compte dans les programmes Fortran, il est donc transparent pour l'utilisateur.

b) Définition des variables de l'asservissement

L'asservissement se faisant axe par axe, nous détaillons pour chacun son type de fonctionnement.

1) axe X :

La paroi se trouve dans la direction X de RJAU, nous donnons donc pour cet axe, une consigne en position (RCCO) située à une distance supérieure à celle séparant le robot du plan E_1. De même, nous donnons une consigne de force suivant cette même direction X.

La consigne en position étant suffisamment grande, la consigne de force sera satisfaite en premier.

La surveillance joue toujours son rôle si la force mesurée en X venait à dépasser une valeur limite.

δd_{mx} = 1.5 mm incrément de déplacement
F_x^* = 0.6 DN $\pm$ 0.05
x^* = 50 mm
FSUVX = 1.5 DN force de surveillance

2) axe Y

Pas de force de consigne dans cette direction.
Une consigne en position et la surveillance de la force.

δd_{my} = 1.5 mm F_y^* = 0 y^* = -50 mm FSUVY = 1.5 DN

3) axe Z

Uniquement la surveillance de la force en Z.

FSUVZ = 1.5 DN $\qquad \delta d_{mZ} = 1.5$ mm

4) rotation autour de Z

surveillance du moment M_Z MSUVZ = 5

incrément de déplacement $\delta \phi_Z = 10$

Pour les trois axes et la rotation, il y a en plus une surveillance du module de la force et du moment résultant qui arrête le robot si leur valeur dépasse un seuil fixé par :

pour la force Module Maxi = 3 DN

pour le moment ———————— = 50 mm . DaN

Quand toutes les consignes sont satisfaites sur les différents axes, l'arête se trouve plaquée contre le plan E_1 où nous avions pris le contact initial.

Le programme enchaîne sur la phase suivante après avoir redéfini certaines de ses variables.

VI.4.2.2. *Phase 2 = suivi de la paroi*

a) Définition des repères

Seul, le repère de consigne de l'asservissement RCCO est redéfini. Il est identique au repère RCO à deux translations près, une en X et une en Y. De plus, il est lié au repère RCO, ce qui entraîne qu'à chaque déplacement du manipulateur, le repère RCCO se déplace d'autant. La consigne en position ne sera jamais atteinte, mais ceci permet d'avoir quand la consigne en force n'est pas satisfaite, un recollement de la pièce située à l'extrémité du poignet sur la paroi à suivre.

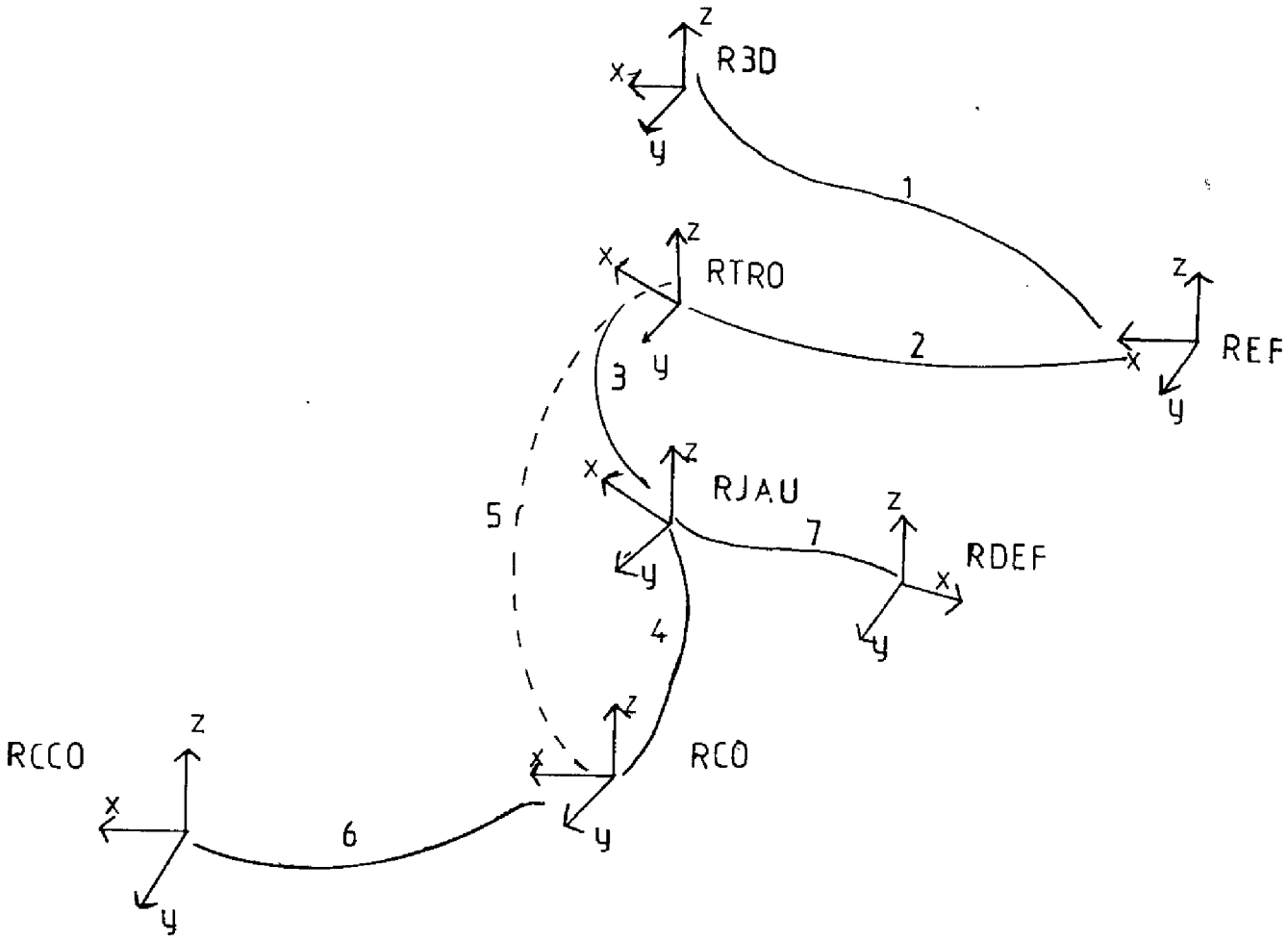


FIGURE VI.9 Graphe des repères

b) Mode d'asservissement

1) axe X

.consigne en force $F_X^* = 0.5$ et en position $X^* = 10$ mm

.surveillance $FSUV_X = 1.5$ DN

2) axe Y

.consigne en force $F_Y^* = 0.2$ et en position $y^* = 10$ mm

.surveillance $FSUV_y = 1.5$ DN

3) axe Z

.surveillance seule FSUVZ = 1.5 DN

4) rotation autour de Z

.surveillance seule MSUVZ = 6 mm daN

Nous avons rajouté un mode, qui permet de bloquer le déplacement suivant un axe si la consigne de force ou de moment n'est pas satisfaite sur un autre axe.

Ce mode de travail est conditionné à l'activation d'une variable accessible par le programme interprété.

Nous nous sommes servis de cette fonction pour privilégier la force de consigne suivant la direction X^+ et la rotation autour de l'axe Z au détriment du déplacement le long de la direction Y^+.

Quand la pièce tenue par le robot se trouve en contact par ses arêtes sur deux plans à la fois (Figure VI.10), le robot s'arrête si toutes les consignes sont satisfaites. En pratique, la valeur du moment est supérieure au seuil de surveillance et le robot ne trouve pas sa consigne d'arrêt. Pour poursuivre le suivi, il faut redéfinir les paramètres de l'asservissement. Un centre de décision est alors indispensable pour faire les choix qui s'imposent en fonction des mesures des efforts.

VI.4.3. Résultats expérimentaux

Le programme permettant au bras manipulateur JAZ de réaliser cette expérimentation, est écrit en langage interprété. Une fois son exécution lancée, le robot enchaîne les différentes tâches qu'il doit accomplir et peut même s'adapter à une perturbation, externe dans la mesure où elle ne menace pas sa constitution physique.

Nous donnons ci-dessous l'évolution des forces et du moment en Z, dans deux phases principales de la manipulation.

Au cours de la manipulation complète de suivi d'un obstacle, nous avons enregistré les efforts servant dans le calcul de l'asservissement pour la génération des incréments de déplacements.

Nous avons tracé des courbes rendant compte de l'évolution de la force en X et du moment autour de Z, en fonction du temps. Nous n'avons pas tracé les évolutions de F_Y et F_Z qui pour la manipulation réalisée restent pratiquement nuls et constants.

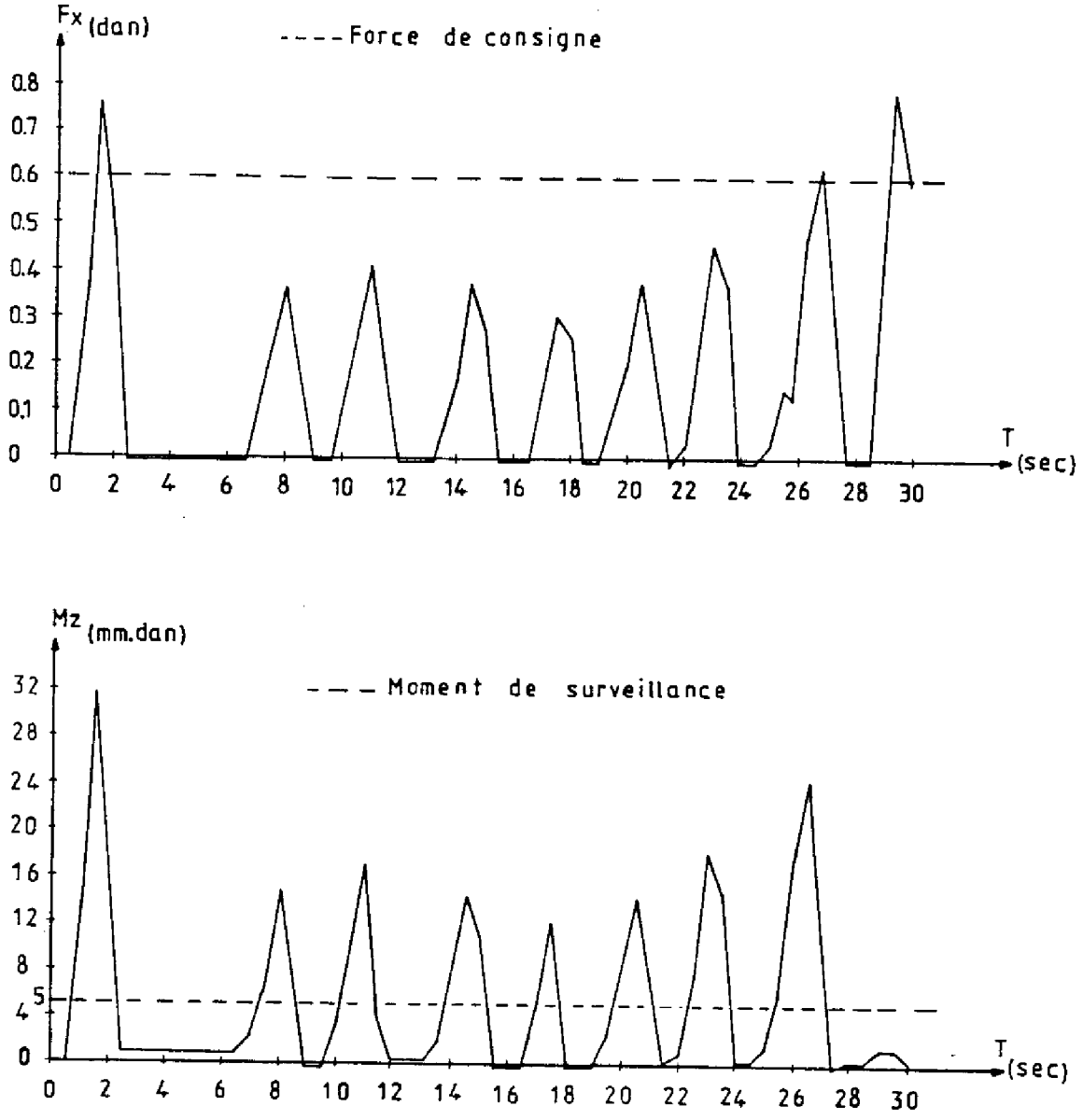


FIGURE VI.10 *Evolution de F_X et M_Z dans la phase de mise en contact d'une arête sur le plan, à partir d'un contact ponctuel*

Les courbes de la figure VI.10 permettent de voir l'évolution de la force et du moment dans la phase de mise en contact de l'arête de la pièce sur le plan de travail. Le contact est ponctuel à l'origine (cf Figure VI.

Nous pouvons voir que le robot se déplace dans la direction X pour générer une force F_X égale à la consigne. Le moment augmente alors et dépasse la valeur de surveillance ; ceci entraîne une rotation de la tête du robot, donc un dégagement du plan de contact ; les efforts sont alors nuls. La consigne en force n'étant pas satisfaite, le robot se déplace pour retrouver le contact. Quand simultanément la force de consigne est atteinte et lemoment est inférieure à la valeur de surveillance, la première phase est terminée, l'arête est plaquée contre le plan.

Les courbes de la figure VI.11 visualise aussi l'évolution de F_X et M_Z, mais cette fois, la pièce tenue par le robot, est dans la phase de suivi du plan et notamment en contact arête sur arête. Le robot doit donc "virer" en fonction des efforts qu'il mesure tout en se maintenant au contac (Figure VI.). Le robot appuie dans la direction X pour satisfaire sa force de consigne, ce qui entraîne une augmentation du moment ; dès que celui-ci dépasse la valeur de surveillance, la tête du robot tourne d'un incrément de rotation, ce qui contribue à diminuer la force en X. De plus, le robot a une consigne de déplacement dans la direction Y pour permettre l'avancement le long de l'obstacle à suivre (cf VI.4.2.2.).

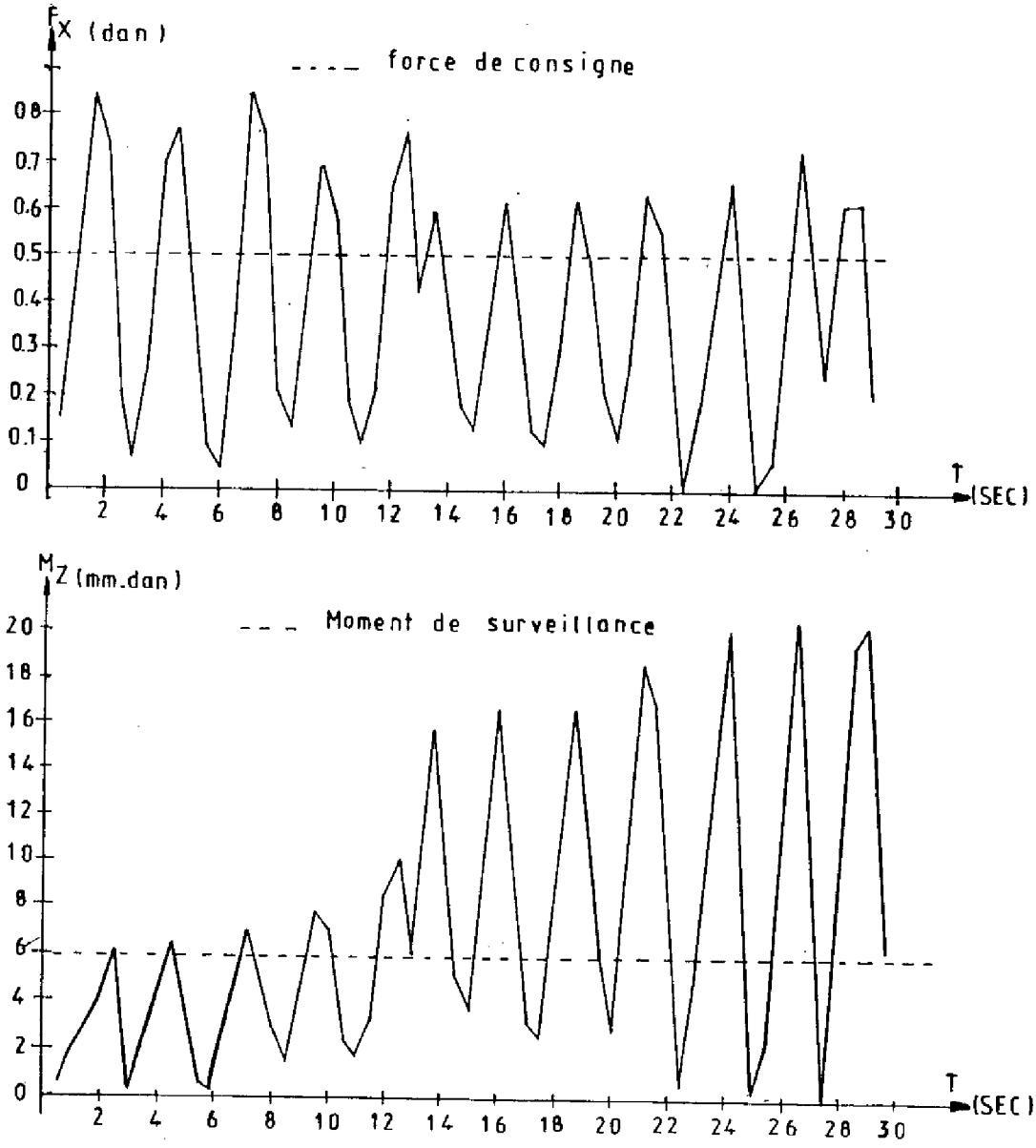


FIGURE VI.11 *Evolution de F_X et M_Z dans la phase de suivi de la paroi par le robot, plus particulièrement contact arête sur arête*

Programme interprété réalisant la manipulation de suivi

```
PROC SUR DISK= PRTES
I=  1 A=  1  MODU=  1
MES 1= INITIALISATION
I=  1 A=  1       32.00       21.00        0.00
I=  2 A=  4       32.00       31.00        1.00
I=  3 A=  7        8.00       15.00

I= 81 A=272  MODU= 15
MES15= ARETE
I= 81 A=272       15.00        5.00
I= 82 A=274       15.00        8.00
I= 83 A=276       15.00        6.00
I= 84 A=278       15.00        2.00
I= 85 A=280       15.00        3.00
I= 86 A=282       15.00        4.00
I= 87 A=284       15.00        7.00
I= 88 A=286       15.00       10.00
MES15= ARETE
I= 89 A=288       19.00       15.00
I= 90 A=290       32.00        6.00        1.00
MES12= SUIVI PAROI
I= 91 A=293       19.00       12.00
I= 92 A=295       32.00        1.00        7.00
I= 93 A=298       15.00       12.00
I= 94 A=300       15.00       10.00
I= 95 A=302       32.00        6.00        1.00
I= 96 A=305        1.00

I=  4 A=  9  MODU=  2
MES 2= NB3D/NBEF
I=  4 A=  9       26.00        4.00
I=  5 A= 11       23.00     1902.00        3.00       20.00        2.00
I=  6 A= 16       23.00    -1905.00        0.00        0.00        1.00
I=  7 A= 21       24.00    -1911.00      100.00      530.00     1120.00     27
I=  8 A= 27       15.00       11.00
I=  9 A= 29       16.00
I= 10 A= 30  MODU=  3
MES 3= NBJAU/NBTRO
I= 10 A= 30       26.00        4.00
I= 11 A= 32       23.00     1902.00        4.00        6.00       15.00
I= 12 A= 37       23.00    -1905.00        0.00        0.00        0.00
I= 13 A= 42       23.00    -1911.00        0.00        0.00        1.00
I= 14 A= 47       23.00    -1917.00        1.00       -0.05        0.00
I= 15 A= 52       15.00       11.00
I= 16 A= 54       16.00
```

```
I= 17 A= 55  MODU=  4
MES 4= NRCO ET NRCCC
I= 17 A= 55        26.00       4.00
I= 18 A= 57        23.00    1902.00        6.00        8.00        2.00
I= 19 A= 62        23.00   -1905.00        0.00        0.00        1.00
I= 20 A= 67        24.00   -1911.00      -29.00       29.00        0.00
I= 21 A= 73        15.00      11.00
I= 22 A= 75        15.00       9.00
I= 23 A= 77        26.00       4.00
I= 24 A= 79        24.00   -1905.00       55.00      -55.00        0.00    -4
I= 25 A= 85        32.00      54.00        1.00
I= 26 A= 88        26.00       4.00
I= 27 A= 90        23.00    1902.00        3.00        7.00        4.00
I= 28 A= 95        15.00      11.00
I= 29 A= 97        23.00    1902.00        7.00        8.00        3.00
I= 30 A=102        15.00      11.00
I= 31 A=104        16.00
I= 97 A=306  MODU= 14
MES14=
I= 97 A=306        16.00

I= 32 A=105  MODU=  5
MES 5= ECHELLE TRACE
I= 32 A=105        26.00       7.00
I= 33 A=107        22.00      -1.00       10.00       20.00
I= 34 A=111        21.00       5.00       20.00
I= 35 A=114        23.00      -6.00     -200.00      200.00     -160.00
I= 36 A=119        24.00     -20.00       -1.00        0.00       -0.50    -
I= 37 A=125        24.00     -28.00        0.00        1.00        0.00
I= 38 A=131        23.00     -36.00        0.00        0.00        0.00
I= 39 A=136        32.00      41.00        7.00
I= 40 A=139        16.00
I= 41 A=140  MODU=  6
MES 6= ACTUR ZERO JALGE
I= 41 A=140        32.00       1.00        7.00
I= 42 A=143        32.00      23.00        1.00
I= 43 A=146        16.00
I= 44 A=147  MODU=  7
MES 7= DEF ASSER
I= 44 A=147        26.00       3.00
I= 45 A=149        22.00      -1.00        3.00       50.00
I= 46 A=153        24.00      -5.00        1.50        1.50        1.50     1
I= 47 A=159        24.00     -17.00        0.60        0.00        2.00     4
I= 48 A=165        24.00     -29.00        0.05        0.05        0.05
I= 49 A=171        24.00     -41.00        1.50        1.50        1.50
I= 50 A=177        24.00     -53.00        5.00        5.00        5.00
I= 51 A=183        16.00
I= 52 A=184  MODU=  8
MES 8= PT RENDEZ VOUS
I= 52 A=184        30.00       6.00        0.00
I= 53 A=187        30.00       7.00        1.00
I= 54 A=190        30.00       1.00      100.00      530.00     1120.00    27(
MES 8= PT RENDEZ VOUS
I= 55 A=196        15.00       8.00
I= 56 A=198        16.00
```

```
I= 57 A=199  MODU=  9
MES 9= MRCO/MRTRC CHEMI
I= 57 A=199        26.00        4.00
I= 58 A=201        23.00     1902.00        4.00        8.00        4.00
I= 59 A=206        15.00       11.00
I= 60 A=208        16.00
I= 61 A=209  MODU= 10
MES10= BOUCLE  ASSER
I= 61 A=209         9.00        0.00     1000.00
I= 62 A=212        32.00        2.00        7.00
I= 63 A=215        11.00       -1.00       14.00
I= 64 A=218        11.00       -2.00       13.00
I= 65 A=221        10.00        0.00        0.00
I= 66 A=224        16.00
I= 67 A=225  MODU= 11
MES11=
I= 67 A=225        32.00       37.00        0.00
I= 68 A=228        16.00

I= 69 A=229  MODU= 12
MES12= SUIVI PAROI
I= 69 A=229        26.00        1.00
I= 70 A=231        24.00        1.00        0.00        2.00        0.00        2.00
I= 71 A=237        26.00        3.00
I= 72 A=239        24.00       -5.00        2.00        2.00        2.00        7.00
I= 73 A=245        24.00      -17.00        0.50        0.50        2.00       40.00
I= 74 A=251        24.00      -41.00        1.50        1.50        1.50        6.00
I= 75 A=257        26.00        4.00
I= 76 A=259        24.00    -1905.00       10.00       60.00        0.00        0.00
I= 77 A=265        32.00       54.00        1.00
I= 78 A=268        16.00
I= 79 A=269  MODU= 13
MES13= BLOCAGE ASSER
I= 79 A=269        19.00       13.00
I= 80 A=271        16.00
```

C O N C L U S I O N

Les travaux que nous avons présentés dans cette thèse constituent une première étape dans le cadre de l'étude de l'assemblage automatique de pièces mécaniques au moyen d'un manipulateur non spécialisé.

Nous nous sommes donc intéressés aux problèmes posés par la mise en contact d'objets à l'aide d'un bras manipulateur, muni d'un capteur d'effort tridimensionnel. A cette fin, nous avons créé des opérateurs permettant :

- la représentation tridimensionnelle de l'univers du manipulateur,
- la maîtrise des forces et des déplacements du robot, notamment par la mise en oeuvre d'un asservissement général en force et position, prenant en compte les contraintes de sécurité,
- l'extraction de caractéristiques géométriques d'un objet inconnu à partir de tests de contact,
- la communication entre l'homme et le robot.

La synthèse de ce travail a été l'intégration de ces différentes fonctions dans une structure hiérarchisée et modulaire, permettant la réalisation de manipulations canoniques de contact (chapitre VI).

A partir de ces bases indispensables, des travaux sont actuellement en cours pour réaliser des manipulations d'insertion simple ou multiple.

L'utilisation à partir d'un poignet flexible passif, d'une compliance active comme celle que nous avons développée, semble être une bonne solution de travail. De plus, dans des montages complexes où les contraintes peuvent être élevées, l'analyse par le contact des degrés de liberté des liaisons mécaniques sera essentielle. Un module décisionnel, déduisant de l'analyse précédente les actions à enchaîner, conduira à la réalisation de la tâche.

Dans notre approche, le support expérimental, constitué par un bras manipulateur à quatre degrés de liberté (trois translations et une rotation) est apparu limitatif. La réalisation d'une table à deux degrés de liberté (2 rotations) permettra d'obtenir les six degrés de liberté souhaités pour faire du montage.

L'adjonction de cet autre robot présente un intérêt supplémentaire qui est celui de coordonner la commande de deux robots participant à la réalisation d'une tâche commune.

Du point de vue de la commande en force et position, nous nous sommes placés sous un aspect quasi-statique, dans la mesure où nous nous situons dans un univers pratiquement inconnu. Le robot ne connaissant pas les obstacles éventuels, il doit générer des petits déplacements jusqu'à la détection d'une force qui lui indique qu'il se trouve au contact d'un objet. Dans ces conditions la stabilité de l'asservissement (cf III.4.1.1.3.) ne nous est pas apparue critique.

La phase suivante sera d'augmenter le nombre d'échantillons de déplacement (deux par seconde actuellement) pour avoir un mouvement plus rapide dans les phases de contact. Ceci peut être réalisé par une reprogrammation du microprocesseur de commande du bras manipulateur pour travailler à une vitesse de 9600 bauds (actuellement 4800 bauds) et par une réduction du temps d'exécution de nos programmes. Les facteurs de cette réduction peuvent être :

- L'écriture de certaines primitives en langage assembleur au lieu du langage Fortran. Par exemple, la primitive (GRARO) gérant les représentations tridimensionnelles par des repères orthonormés,
- Le transport sur un calculateur beaucoup plus puissant (SEL 32) de l'ensemble de notre logiciel (gain d'un facteur 100 en temps de calcul). En particulier, la structure de recouvrement ("overlay") ne serait plus nécessaire.

Nous envisageons aussi dans le cas de tâches répétitives, et quand l'univers de travail est mieux connu, de permettre au robot de se mouvoir le long d'une trajectoire. Un contrôle des efforts fournirait les corrections nécessaires à cette trajectoire, dans le cas d'un mouvement au contact.

Une dernière étape sera d'étudier le comportement dynamique de notre système, notamment au niveau de la réponse du capteur au cours de déplacement rapide au contact. Ce problème est encore largement ouvert, mais fait l'objet de travaux importants. La prise en compte des inerties du système, pour une commande dynamique rapide, demande néanmoins, au niveau expérimental de travailler dans un univers où les objets sont décrits. Ceci pour s'affranchir des problèmes d'accostage. Dans le cas du travail du robot avec des contraintes de forces sur son organe terminal (insertion), la vitesse n'est cependant pas un facteur critique, la priorité étant donnée à la maîtrise des forces.

D'autre part, ne disposant pas d'une pince sur notre bras manipulateur, nous n'avons pas pu aborder le problème de la saisie. Néanmoins, sa prise en compte ne devrait pas ajouter de grande complexité, en regard du logiciel développé, lorsque la pièce à saisir a été localisée par d'autres moyens de perception, notamment la vision.

Nous pouvons donc envisager dans un proche avenir la réalisation complète d'une manipulation d'assemblage automatique regroupant la localisation d'objets, la saisie, le transport et le montage.

§§§

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

[ABA 63] L. ABADIE
"Utilisation des quaternions pour la représentation des coordonnées angulaires. Techniques de calcul analogique et numérique en aéronautique". Congrès de Liège, 1963, pp. 342-346.

[BAU 81] G. BAUZIL, M. BRIOT, P. RIBES
"A navigation sub-system using ultrasonic sensors for the mobile robot Hilare", 1st International conference on a robot vision and sensory controls, Stratford-upon-Avon, U.K., April 1-3 1981.

[BEJ 80] A.K. BEJCZY
"Kinesthetic and graphic feedback for integrated operator control", 6th annual advanced control conference man machine interfaces for industrial control, Perdue University, Avril 1980.

[BRI 77] M. BRIOT
"La stéréognosie en robotique : application au tri de solides", Thèse d'Etat présentée à l'U.P.S. de Toulouse, Novembre 1977.

[BRU 77] H. VAND BRUSSEL, J. SIMONS
"Automatic assembly by active force feedback accomodation", Mechanical engineering department.

[CAL 75] H. Mac CALLION, P.C. WONG
"Some thoughts on the automatic assembly of a peg and a hole", Fourth world congress on the theory of machines and mechanisms, University of Newcastle upon Time, 8-12 September 1975, pp.347-352.

[DRA 77] S.H. DRAKE, P.C. WATSON and S.N. SIMUNOVIC
"High speed robot assembly of precision parts using compliance instead of sensory feedback", M.I.T./Charles Stark draper laboratory, Proceedings 7th International symposium on industrial robots, Tokyo, 1977.

[FER 81] M. FERRER, G. BAUZIL, M. BRIOT
"Etude d'un système de traitement d'images destiné au robot mobile HILARE", 8e Colloque sur le traitement du signal et ses applications, Juin 1981.

[FIN 74] R. FINKEL et al
"AL, a programming system for automation", Memo AIM-243, Stanford University - Intelligence Laboratory, Stanford Californie, November 1974.

[FIN 75] FINKEL, R. TAYLOR, R. BOLLES, R. PAUL, J. FELDMAN
"An overview of AL, a programming system for automation", 4th I.J.C.A.I. Tbilissi (USSR), September 1975.

[GER 78] E.G.R. GERELLE
"Force feedback control", Institut microtechnique, 8th Symposium on industrial robots, Juin 1978, STUTTGART.

[GIR$_a$ 78] G. GIRALT
"Le développement de la robotique", Conférence présentée aux 41e journées de la S.E.E., PARIS, Novembre 1978.

[GIR$_b$ 79] A. GIRAUD
"Robot de chargement et déchargement d'un poste de travail sur une machine à plateau tournant", Rapport de fin de contrat en collaboration avec les sociétés JAZ et CERCI, Décembre 1979.

[GIR$_b$ 81] A. GIRAUD, R. PRAJOUX
"La problématique de l'assemblage automatique en robotique", 3ème journées scientifiques et techniques de la production automatisée, Toulouse, 3,4 et 5 Juin 1981.

[GOT 72] T. GOTO, K. TAKEYASU, T. INOYAMA, R. SHIMOMURA
"Compact packaging by robot with tactile sensors", Central research laboratory of Hitachi Ltd, 2nd International Symposium on Industrial robots, CHICAGO, May 1972.

[GOT 74] T. GOTO, T. INOYAMA, K. TAKEYASU
"Precise insert operation by tactile controlled robot", Proceedings of the end conference on industrial robot technology, September 1974.

[GRO 75] D. GROSSMAN
"Procedural representation of three dimensional objects", Research Report RC-5314 IBM-T, J. Watson Research Center, Yorktown Heights, New York, March 1975.

[GUR 74] V.C. GURFINKEL, A.V. SCHNEIDER, E.M. KANAIEV
"Le sens tactile des robots", Nouvelles de l'académie des sciences de l'URSS, La cybernétique technique (section "recherches"), MOSCOU, 1974.

[KEN 80] J. KENNETH SALISBURY
"Active stiffness control of a manipulator in cartesian coordinates" Department of computer science, Stanford University, 19th IEEE Conference on decision and control, December 1980, ALBUQUERQUE, New Mexico.

[LAN 69] L. LANDAU et E. LIFCHITZ
"Physique théorique - tome 1 - Mécanique", Editions MIR, MOSCOU, 1969

[LAT 80] J.C. LATOMBE, E. MAZER
"Définition d'un langage de programmation pour la robotique (LM), Analyse fonctionnelle de l'interpréteur", Rapport de recherche IMAG n°197, Mars 1980.

[LIE 75] L. LIEBERMAN
"Autopass, a very high level programming language for mechanical assembler systems", RC 5599, n°24205 IBM-T, J. Watson Research Center, Yorktown Heights, New York, Août 1975.

[MAZ 81] E. MAZER
"Réalisation d'un support expérimental de recherche pour le projet robotique Pandore. Définition et implantation du langage LM". Thèse de 3e cycle en génie informatique, 12 Janvier 1981.

[NEV 73] J.L. NEVINS, D.E. WHITNEY
"The force vector assembler concept", C.S.D./M.I.T., Proceedings 1st CISM symposium, UDINE, ITALY, 1973.

[NEV 74] J.L. NEVINS
"Exploratory research in industrial modular assembly", C.S.D.L. Rep. n°R.850, Décembre 1974, pp. 11-26.

[NEV 78] J.L. NEVINS, D.E. WHITNEY and al
"What is the remote center compliance (RCC) and what can it do ?" C.S.D.L. Report n°P-278, Novembre 1978.

[NIT 79] D. NITZAN
"Robotic automation at SRI", SRI International, Proceedings of Midcon 179, CHICAGO, 6-8 November 1979.

[OHW 78] M.S. OHWOVORIOLE, S.N. HILL, B. ROTH
"On the theory of single and multiple insertions in industrial assemblies", Stanford University

[PAR 77] W.T. PARK
"Minicomputer software organization for control of industrial robots", Stanford Research Institute, Proceedings of the 1977 joint automatic control conference.

[PAU 77] R. PAUL
"WAVE : a model-based language for manipulator control", The industrial robots, Mars 1977.

[PRA 71] B. PRADIN
"Semi-automatic manipulating robots control on the basis of specialized calculators", Proceedings 3rd CISM, Symposium on theory and practice of robots and manipulators UDINE, 12-15 September 1978.

[REN 80] M. RENAUD
"Contribution à la modélisation et à la commande dynamique des robots manipulateurs", Thèse d'Etat en automatique à U.P.S. Toulouse, 15 Septembre 1980.

[REU 63] F. REULEAUX
"Kinematics of Machinery", DOVER, NEW YORK, 1963, pp. 96-114.

[SIM 75] S. SIMUNOVIC
"Force information in assembly processes", C.S.D.L. Proceedings of the 5th International symposium on industrial robot, CHICAGO, Illinois, September 1975.

[STU 80] F. STUCK
"Réalisation d'un système adaptatif de traitement d'images pour l'identification et la localisation de pièces en robotique", Thèse de 3ème cycle U.P.S. Toulouse, 3 Décembre 1980.

[TAK 76] K. TAKEYASU, T. GOTO, T. INOYAMA
"Précision insertion control robot and its application", Journal of engineering for industry, November 1976.

[WAT 75] P.C. WATSON and S.H. DRAKE
"Pedestal and wrist force sensors for automatic assembly", C.S.D.L. V^{e} international symposium on industrial robot, September 1975, CHICAGO.

[WHIT 77] D.E. WHITNEY, P.C. WATSON, S.H. DRAKE, S.N. SIMUNOVIC
"Robot and manipulator control by exteroceptive sensors", Proceedings of the 1977 joint automatic control conference.

TABLE DES MATIÈRES