Génération de taches bicolores : application aux caractères d'imprimerie; problèmes de nature ordinale

Marc Bloch

ECOLE NATIONALE
SUPERIEURE DES MINES
DE SAINT-ETIENNE

INSTITUT NATIONAL
POLYTECHNIQUE
DE GRENOBLE

N° d'ordre : 23 II

THESE

présentée par

Marc BLOCH

pour obtenir le grade de

DOCTEUR-INGENIEUR
"SYSTEMES ET RESEAUX INFORMATIQUES"

GENERATION DE TACHES BICOLORES

-APPLICATION AUX CARACTERES D'IMPRIMERIE-

PROBLEMES DE NATURE ORDINALE

Soutenue à Saint-Etienne le 1er Juillet 1981 devant la commission d'examen :

Président	M. ANCEAU
Examinateurs	MM. COUEIGNOUX
	LUCAS
	MAHL
	NAUDIN

ECOLE NATIONALE
SUPERIEURE DES MINES
DE SAINT-ETIENNE

INSTITUT NATIONAL
POLYTECHNIQUE
DE GRENOBLE

N° d'ordre : 23 II

THESE

présentée par

Marc BLOCH

pour obtenir le grade de

DOCTEUR-INGENIEUR
"SYSTEMES ET RESEAUX INFORMATIQUES"

GENERATION DE TACHES BICOLORES

-APPLICATION AUX CARACTERES D'IMPRIMERIE -

PROBLEMES DE NATURE ORDINALE

Soutenue à Saint-Etienne le 1er Juillet 1981 devant la commission d'examen :

Président	M. ANCEAU
Examinateurs	MM. COUEIGNOUX
	LUCAS
	MAHL
	NAUDIN

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

Directeur	: M. M.	MERMET
Directeur des Etudes et de la Formation	: M. J.	LEVASSEUR
Directeur des Recherches	: M. J.	LEVY
Directeur Administratif et Financier	: M. A.	COINDE

PROFESSEURS DE 1ère CATEGORIE

MM.	COINDE	Alexandre	Gestion
	GOUX	Claude	Métallurgie
	LEVY	Jacques	Métallurgie
	RIEU	Jean	Mécanique - Résistance des Matériaux
	SOUSTELLE	Michel	Chimie
	FORMERY	Philippe	Mathématiques Appliquées

PROFESSEURS DE 2ème CATEGORIE

MM.	GUIBOUD-RIBAUD	Serge	Informatique
	LOWYS	Jean-Pierre	Physique
	TOUCHARD	Bernard	Physique Industrielle

DIRECTEUR DE RECHERCHE

M.	LESBATS	Pierre	Métallurgie

MAITRES DE RECHERCHE

MM.	BISCONDI	Michel	Métallurgie
	COUEIGNOUX	Philippe	Informatique
	DAVOINE	Philippe	Géologie
Mle	FOURDEUX	Angeline	Métallurgie
MM.	KOBYLANSKI	André	Métallurgie
	LALAUZE	René	Chimie
	LANCELOT	Francis	Chimie
	LE COZE	Jean	Métallurgie
	MATHON	Albert	Gestion
	PERRIN	Michel	Géologie
	THEVENOT	François	Chimie
	TRAN MINH	Canh	Chimie

PERSONNALITES HABILITEES A DIRIGER DES TRAVAUX DE RECHERCHE

MM.	DRIVER	Julian	Métallurgie
	GUILHOT	Bernard	Chimie
	THOMAS	Gérard	Chimie

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année Universitaire
1980-1981

Président : M. Daniel BLOCH

PROFESSEURS DES UNIVERSITES

MM ANCEAU François	Informatique fondamentale et appliquée
BESSON Jean	Chimie Minérale
BLIMAN Samuël	Electronique
BLOCH Daniel	Physique du Solide - Cristallographie
BOIS Philippe	Mécanique
BONNETAIN Lucien	Génie Chimique
BONNIER Etienne	Métallurgie
BOUVARD Maurice	Génie Mécanique
BRISSONNEAU Pierre	Physique des Matériaux
BUYLE-BODIN Maurice	Electronique
CHARTIER Germain	Electronique
CHENEVIER Pierre	Electronique
CHERADAME Hervé	Chimie Physique Macromoléculaire
MmeCHERUY Arlette	Automatique
MM CHIAVERINA Jean	Biologie, biochimie, agronomie
COHEN Joseph	Electronique
COUMES André	Electronique
DURAND Francis	Métallurgie
DURAND Jean-Louis	Physique Nucléaire et Corpusculaire
FELICI Noël	Electrotechnique
FOULARD Claude	Automatique
GUYOT Pierre	Métallurgie Physique
IVANES Marcel	Electrotechnique
JOUBERT Jean-Claude	Physique du Solide - Cristallographie
MmeJOURDAIN Geneviève	Traitement du Signal
MM LACOUME Jean-Louis	Géophysique - Traitement du Signal
LANCIA Roland	Electronique - Automatique
LESIEUR Marcel	Mécanique
LESPINARD Georges	Mécanique
LONGEQUEUE Jean-Pierre	Physique Nucléaire Corpusculaire
MOREAU René	Mécanique
MORET Roger	Physique Nucléaire Corpusculaire
PARIAUD Jean-Charles	Chimie-Physique
PAUTHENET René	Physique du Solide - Cristallographie
PERRET René	Automatique
PERRET Robert	Electrotechnique
PIAU Jean-Michel	Mécanique
POLOUJADOFF Michel	Electrotechnique
POUPOT Christian	Electronique - Automatique
RAMEAU Jean-Jacques	Electrochimie - Corrosion
ROBERT André	Chimie appliquée et des matériaux
ROBERT François	Analyse numérique
SABONNADIERE Jean-Claude	Electrotechnique
MmeSAUCIER Gabrielle	Informatique fondamentale et appliquée

.../...

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

PROFESSEURS DES UNIVERSITES

MmeSCHLENKER Claire	Physique du Solide - Cristallographie
MM SCHLENKER Michel	Physique du Solide
SOHM Jean-Claude	Chimie Physique
TRAYNARD Philippe	Chimie - Physique
VEILLON Gérard	Informatique fondamentale et appliquée
ZADWORNY François	Electronique

CHERCHEURS DU C.N.R.S. (Directeur et Maîtres de Recherche)

M FRUCHART Robert	Directeur de Recherche
MM ANSARA Ibrahïm	Maître de Recherche
CARRE René	Maître de Recherche
DAVID René	Maître de Recherche
DRIOLE Jean	Maître de Recherche
KAMARINOS Georges	Maître de Recherche
KLEITZ Michel	Maître de Recherche
LANDAU Ioan-Doré	Maître de Recherche
MERMET Jean	Maître de Recherche
MUNIER Jacques	Maître de Recherche

Personnalités habilitées à diriger des travaux de recherche (Décision du Conseil Scientifique)

E.N.S.E.E.G.

MM ALLIBERT Michel	
BERNARD Claude	
CAILLET Marcel	
MmeCHATILLON Catherine	
MM COULON Michel	
HAMMOU Abdelkader	
JOUD Jean-Charles	
RAVAINE Denis	
SAINFORT	C.E.N.G.
SARRAZIN Pierre	
SOUQUET Jean-Louis	
TOUZAIN Philippe	
URBAIN Georges	Laboratoire des Ultra-Réfractaires ODEILLO

E.N.S.M.S.E.

MM BISCONDI Michel	
BOOS Jean-Yves	
GUILHOT Bernard	
KOBILANSKI André	
LALAUZE René	
LANCELOT Francis	
LE COZE Jean	
LESBATS Pierre	
SOUSTELLE Michel	
THEVENOT François	

.../...

THOMAS Gérard
TRAN MINH Canh
DRIVER Julian
RIEU Jean

E.N.S.E.R.G.

MM BOREL Joseph
CHEHIKIAN Alain
VIKTOROVITCH Pierre

E.N.S.I.E.G.

MM BORNARD Guy
DESCHIZEAUX Pierre
GLANGEAUD François
JAUSSAUD Pierre
Mme JOURDAIN Geneviève
MM LEJEUNE Gérard
PERARD Jacques

E.N.S.H.G.

M DELHAYE Jean-Marc

E.N.S.I.M.A.G.

MM COURTIN Jacques
LATOMBE Jean-Claude
LUCAS Michel
VERDILLON André

*
* *
*

Je tiens à remercier :

Monsieur François ANCEAU, Professeur à l'Ecole Nationale d'Ingénieurs en Mathématiques Appliquées de Grenoble, qui m'a fait l'honneur de présider le jury de cette thèse,

Monsieur Michel LUCAS, Professeur à l'Université de Nantes, dont j'ai apprécié les conseils encourageants, et qui a accepté de juger ce travail,

Monsieur Robert MAHL, Responsable du secteur Energies et Matières Premières à la DGRST, qui, tout au long du projet, s'est toujours intéressé à nos problèmes,

Monsieur Paul NAUDIN, directeur de LogAbax-Arts Graphiques, sans lequel l'ensemble des recherches, depuis longtemps menées sur le sujet à l'ENSMSE ou ailleurs, n'aurait pu aboutir de manière concrète,

Monsieur Philippe COUEIGNOUX, Maître de Recherche à l'Ecole Nationale Supérieure des Mines de Saint-Etienne, initiateur de cette étude, qui ne m'a jamais ménagé ni ses critiques, ni ses encouragements.

J'exprime aussi mon amitié à mon complice Christian SICO, sur lequel je me suis souvent appuyé pour la résolution des problèmes rencontrés.

Enfin, je remercie tous ceux qui ont contribué à la réalisation matérielle de cet ouvrage : Mesdames BONNEFOY et AVONDO, Messieurs BROSSARD, VELAY, LOUBET et DARLES.

Avant-propos

Cet ouvrage repose sur un travail de recherche mené conjointement avec Christian SICO : étude et réalisation d'un générateur digital de caractères d'imprimerie, destiné à des applications de haute qualité dans les domaines de la photocomposition et de la bureautique.

La première partie (Chapitre 1) présente les principes généraux qui ont sous-tendu le projet, et décrit brièvement l'architecture logicielle et matérielle du prototype que nous avons construit. D'écriture plus descriptive que scientifique, elle a été rédigée en commun avec C. SICO ; elle est néanmoins nécessaire à une bonne compréhension des problèmes abordés, aussi bien dans le mémoire de C. SICO, dont elle formera également l'introduction, que dans celui que je présente ici.

Dans la deuxième partie, dont le plan est donné au chapitre 2, est exposé le travail personnel sur lequel repose ce mémoire :

Les chapitres 3 et 4 traitent de deux points importants faisant l'objet de développements théoriques originaux ; il s'agit d'étudier certaines relations sur l'ensemble des parois d'une tache bicolore régulière, ce qui légitime le sous-titre : "problèmes de nature ordinale".

Les chapitres 5 et 6 évoquent deux problèmes intéressants relatifs à la partie du projet dont la conception m'incombait (gestion de programmes et de données, analyse des performances dans un environnement multiprocesseur à fort degré de parallélisme).

Plan

Chapitre 1

CHAPITRE 1

UN GENERATEUR DIGITAL DE CARACTERES

1.- Représentation des caractères d'imprimerie.

A.- L'esprit et la lettre.
B.- Le caractère d'imprimerie.
C.- Codage digital des caractères d'imprimerie.

2.- Définition d'un nouveau générateur digital de caractères.

A.- Constat initial.
B.- Objectifs.
C.- Marchés.

3.- Principes de réalisation.

A.- Interpolation du contour.
    1) Codage du contour.
    2) Compression numérique.
    3) Compression linguistique.
B.- Structures de parallélisme.
    1) Mise en parallèle des parois.
    2) Pipe-line de décodage et modularité.
    3) Variantes.

4.- Résultats obtenus.

A.- Structure effective.

1) Expansion linguistique.

2) Expansion numérique.

3) Suivi de contour.

4) Mise à l'échelle.

5) Synchronisation.

6) Réalisation.

B.- Performances.

1 - PRESENTATION DES CARACTERES D'IMPRIMERIE

A - L'ESPRIT ET LA LETTRE

Depuis les grandes découvertes de la Renaissance, qui ont assuré la généralisation de sa diffusion par l'impression, l'écriture est, dans notre civilisation, l'instrument privilégié de la transmission de l'information et de la connaissance. Aujourd'hui encore, ses avantages : faible coût, disponibilité immédiate, faible dégradabilité, richesse graphique, confidentialité, etc... expliquent qu'elle résiste bien aux autres moyens de communication de notre époque (audiovisuel, télématique) ; ces avantages n'auraient jamais été valorisés sans les progrès continus des techniques d'impression et de manipulation du texte. A l'heure actuelle, la mémorisation des textes, leur mise en page sont effectuées sur des ordinateurs, et leur restitution sur des machines complexes (presses offset, héliogravure, imprimante).

L'un des problèmes essentiels pour la reproduction des textes est le passage d'une représentation interne, (c'est-à-dire en machine), qui condense provisoirement le message en sa plus simple expression (codes ASCII, binaire...), à la représentation externe et définitive, dont la forme sur le support restitue le sens du message : c'est la phase de composition, d'abord manuelle, puis automatisée (HOU 78) qui engendre le texte en tant que forme.

Il est ici important de bien distinguer le sens d'un texte, le signifié, de la forme qu'il prend, le signifiant : un ensemble de symboles archétypiques, les caractères qui ne condensent aucune signification dans notre écriture se regroupent en mots, puis en phrases pour former un texte qui peut ou non avoir un sens. Mais, comme c'est le cas pour d'autres canaux de communication (la parole par exemple), la forme de l'écriture est elle même chargée d'une signification secondaire (caractères anciens ou modernes, gras ou légers,

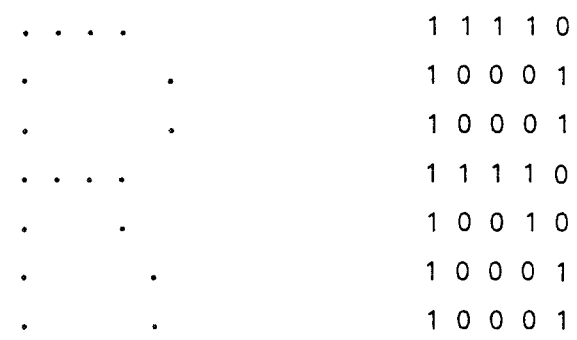
ronds ou droits). Au message principal, le sens des mots et du texte se superpose un message secondaire , l'ambiance du texte. Même automatisé, le processus d'écriture doit conserver ces variations, ce surplus d'information, cette redondance qui permettent une meilleure lisibilité et restituent une certaine sensibilité.

Pour des raisons technologiques et économiques, l'informatique a beaucoup sous-estimé l'importance de la forme de l'écrit ; de toutes façons, l'informaticien, qui connaît le prix de l'information, n'aime guère la redondance. Tout cela rend les listages d'ordinateur à la fois si uniformes et si peu lisibles... Par contre, dans ses applications au traitement de texte et à l'impression classique, l'informatique doit permettre d'égaler la richesse graphique des outils qu'elle tend à supplanter (machines à écrire, caractères au plomb...).

B - LE CARACTERE D'IMPRIMERIE

Les méthodes rapides d'impression pour le livre, la presse ou l'informatique exigent des méthodes rapides de restitution du texte ; il en existe deux grandes classes. Dans la première, la forme du caractère préexiste et est simplement transférée sur le support, par des moyens mécaniques (imprimantes à marguerites ou à chaîne) ou optiques (photocomposeuses à flash). De tels procédés ne nous concernent pas directement ; signalons cependant que les technologies utilisées ont limité l'évolution de deux caractéristiques importantes : la vitesse (à qualité donnée), et l'aptitude à manipuler la forme même du caractère (mise à l'échelle, inclinaison, épaississement). La seconde classe comprend toutes les méthodes où la forme du caractère doit être engendrée à chaque utilisation, (imprimantes à aiguille, afficheurs électroluminescents à segments ou à points, écrans de visualisation à balayage vidéo, photocomposeuses digitales) ; il faut alors mémoriser indépendamment de la technique de visualisation, une représentation du caractère. La plus commode, la mieux adaptée aux outils modernes de traitement de l'information, est la représentation digitale : la surface du caractère est discrétisée par échantillonnage sur une grille régulière, et on obtient une matrice de points

blancs ou noirs, à laquelle on fait correspondre une matrice binaire, la matrice du caractère. Une réalisation très répandue de ces principes se retrouve dans les écrans à points ou les imprimantes à aiguilles. La taille de la matrice est alors très petite : 5x7, 7x9 (figure 1.1).



Discrétisation d'un caractère — Matrice associée (taille 7 x 5)

Figure 1-1

L'utilisation de petites matrices convient parfaitement aux imprimantes d'ordinateur, dont la qualité n'est pas la caractéristique essentielle. Il en va autrement pour le traitement de texte ou l'impression classique ; il faut alors représenter les caractères par de grandes matrices, et cela pour trois raisons principales :

* le nombre de formes que l'on peut engendrer à partir d'une petite matrice est très réduit ; une grille 5x7, par exemple, permet de représenter de manière schématique les majuscules de l'alphabet latin, les chiffres et quelques symboles spéciaux, mais non les caractères d'autres alphabets, souvent plus complexes (arabe, chinois).

* on ne peut représenter l'esthétique de la forme ; le message est pratiquement brut, et la signification secondaire, dont nous avons dégagé la nécessité, en est absente.

* l'appréhension de la forme doit être immédiate et agréable ; le bruit de quantification inhérent à la numérisation, qui se manifeste par des sauts quantifiés sur le support, doit être insignifiant après transfert sur le support, c'est-à-dire au maximum de l'ordre du pouvoir séparateur de l'oeil ; on peut alors restituer l'apparence d'une surface homogène dont le bord est parfaitement régulier.

  C'est ici qu'intervient la technique de visualisation par l'intermédiaire de la résolution du support, dont la gamme s'étend de quelques points au centimètre (écrans à balayage vidéo) à quelques centaines de points au centimètre (écrans graphiques de haute qualité, imprimantes xérographiques ou électrostatiques).

La résolution du support, et la taille de la matrice définissant le caractère déterminent ensemble la taille du caractère sur le support : un caractère défini dans une grille de 2000 points par 2000 points aura une hauteur de 4 cm sur un support de résolution 500 points/cm. Réciproquement, la taille maximale désirée des caractères sur le support, et la définition de ce support déterminent la taille des matrices : pour une hauteur maximale de 4 cm (pensons aux gros titres des journaux), et une résolution de 400 points/cm (légèrement supérieure au pouvoir séparateur de l'oeil), il faut représenter le caractère dans une grille de 1600 points sur 1600 points.

Il est malheureusement impossible de mémoriser les caractères par leur matrice, dès que la taille de celle-ci devient grande : le stockage d'un seul caractère de taille 2000x2000 exige 4 millions de bits... De toutes façons, la quantité d'information utilisée est alors grossièrement surévaluée ; par exemple, lorsque les caractères sont assez réguliers, les points du contour sont bien moins nombreux que les points de la surface, et en permettent pourtant la reconstitution. Un codage de la matrice est nécessaire ; la restitution se fait alors par l'intermédiaire d'un générateur de caractères, plus complexe que celui qui assure une simple recopie de la matrice.

C - CODAGE DIGITAL DES CARACTERES D'IMPRIMERIE

Avant d'expliciter les différentes catégories de code, nous allons décrire les variables qui vont nous permettre de les caractériser (COU -). Les premières ne dépendent que du code utilisé ; ces facteurs endogènes sont :

. la quantité moyenne de mémoire nécessaire à la description d'un caractère ; nous dirons la taille du code ;

. la dimension de la plus grande matrice de points engendrable sans déformation, ou qualité du code ;

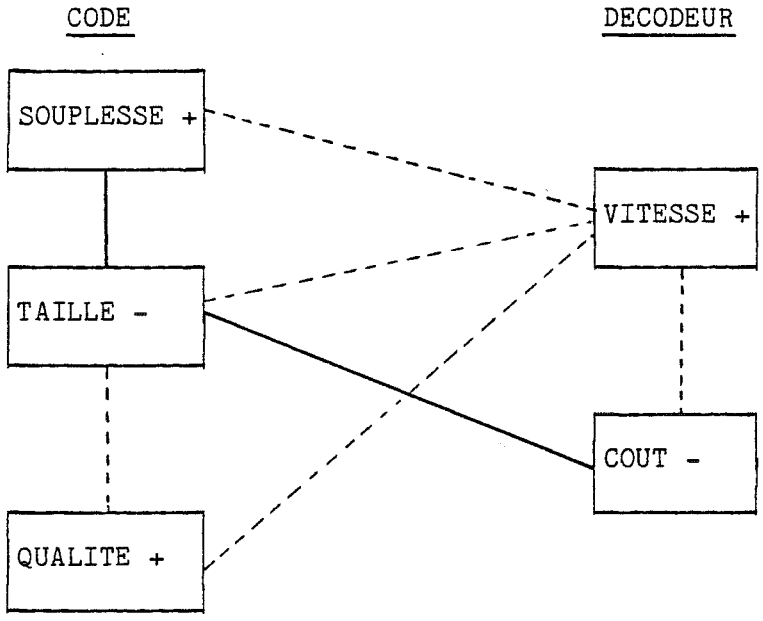
. l'ensemble des diverses modifications (mise à l'échelle, rotations, inclinaisons, épaississement du trait) que la nature du code permet d'automatiser simplement, ou souplesse du code.

D'autres facteurs dépendent aussi du décodeur :

. la vitesse de génération ;

. le coût du générateur.

Bien sûr, ces variables ne sont pas indépendantes, et la figure 1-2 montre certaines de leurs interrelations :



1) Pour chaque facteur, le sens de variation souhaité est indiqué par le signe.

2) Une liaison pleine (resp. pointillée) indique que les sens de variation souhaités sont compatibles (resp. incompatibles).

Figure 1-2

Des critères externes peuvent aussi intervenir (COG 80) :

. la résolution du support, que nous avons déjà mentionnée.

. la méthode de remplissage : les codes qui utilisent ou calculent une représentation du contour ont besoin d'un tampon de mémoire pour le remplissage de la surface. Si le contour est déterminé de manière purement séquentielle, le remplissage se fait après calcul dans un tampon dont la taille doit être celle de la matrice de représentation. Nous verrons que certains types de codes permettent le remplissage au vol du contour, en une passe, horizontale ou verticale, la taille du tampon est alors celle d'une ligne ou d'une colonne de la matrice.

. la méthode de balayage : les balayages cavaliers sont peu adaptés à l'encrage des surfaces ; pour minimiser le déplacement, on en arrive à simuler un balayage vidéo ligne par ligne. Nous sommes donc concernés essentiellement par les balayages vidéo ; il faut en distinguer deux classes :

  - balayage vidéo unitaire : les caractères sont engendrables individuellement sur le support.

  - balayage vidéo composite : il faut calculer toute une ligne de texte avant de pouvoir la restituer sur le support, c'est-à-dire engendrer, individuellement ou non, l'ensemble des caractères qui la constituent dans un tampon de mémoire spécifique.

Les compromis dépendent ici de la technique de visualisation : les supports qui sont gérés par une mémoire d'image permettent aussi bien un remplissage a postériori qu'au vol et sont mieux adaptés à un balayage composite, puisque les tampons nécessaires existent déjà ; par contre, les supports qui n'ont pas besoin de mémoire intermédiaire propre - par exemple tubes de photocomposeuses où la

mémorisation se fait immédiatement sur un film photosensible -, demandent plutôt un remplissage au vol et un balayage unitaire, pour minimiser la quantité de tampon externe.

Il existe trois grandes classes de code : représentations de la surface, du contour ou de la structure des caractères ; leurs diverses variantes permettent d'obtenir un codage optimal pour un usage déterminé. Le lecteur pourra trouver dans (COU -) une bibliographie et des développements sur le sujet.

1°) Codages de la surface

N est la dimension de la grille de discrétisation.

* matrice binaire

Le caractère est codé par sa matrice ; la vitesse de génération est optimale, mais la taille du code est proportionnelle à N^2 (fig 1-3-1).

* code par plages

Pour chaque ordonnée Y_i (resp. abscisse X_i), on mémorise les abscisses (resp. les ordonnées) des points d'intersection du contour avec la ligne $Y=Y_i$ (resp. la colonne $X=X_i$) ; les segments ainsi déterminés sont des plages. C'est un code en $kN \log_2 N$, où k est une constante représentative de la complexité moyenne du caractère : 4 pour les caractères de polices romaines de corps de texte, plus de 10 pour les chinoises.
Ce code est particulièrement adapté au balayage vidéo (figure1-3-2).

2°) Codages du contour

* code par plages différentiel

On se contente de noter les variations des extrémités des plages. Les deux extrémités d'une plage particulière peuvent alors être traitées séparément sous forme de deux listes de

listes de variations. Cette dichotomie a l'avantage de faire apparaître des morceaux de contour, qu'on nommera parois, indépendants les uns des autres, donc adaptés à un traitement parallèle. Il faut malheureusement rajouter au code une superstructure qui permet de repérer les naissances et les morts des parois ; on perd aussi une notion d'ordre qui était implicite dans le balayage par plage (fig. 1-3-3).

La taille d'un tel code, bien adapté au balayage vidéo, peut être approchée par $6\ k\ N + b \log_2 N + c$.

k : nombre moyen d'intersections, donc de parois $\simeq 4$

b : nombre moyen de points de naissance (Romaine $\simeq 4$)

c : caractéristique de la gestion des naissances (16 bits).

* code interpolé du contour

Plutôt que de coder toutes les variations des points du contour, on considère ce dernier comme la concaténation d'interpolants canoniques, tels que segments de droites, arcs de coniques ou plus généralement splines (COU 73) (COU 75), mémorisés sous forme analytique. Une telle approche est compatible avec la structure dégagée au paragraphe précédent, à condition qu'une paroi soit codée par un nombre entier de courbes. Les points d'une paroi sont calculés à partir de l'équation aux différences finies de l'interpolant.

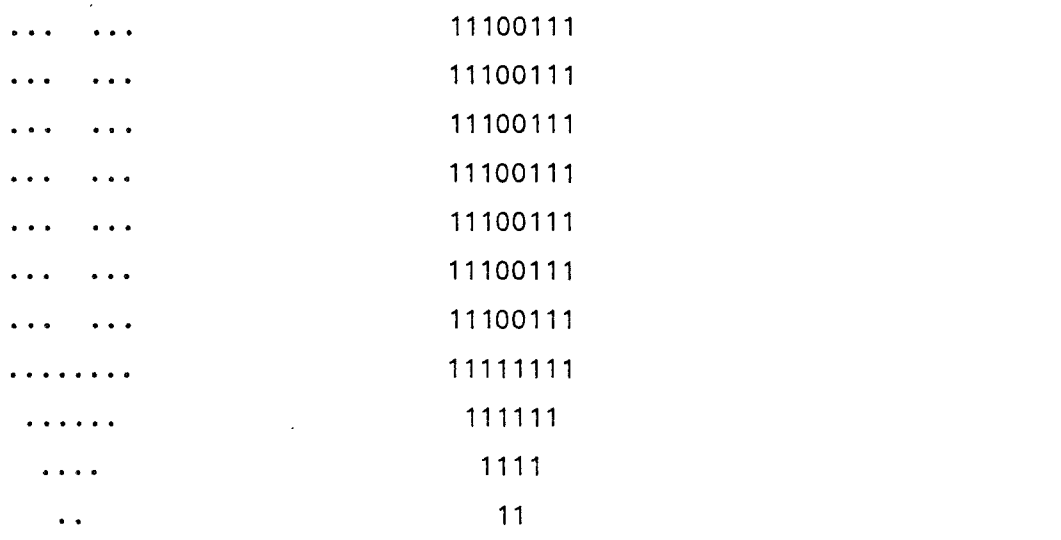
3°) codage de la structure

* Primitives (COU 75)

Le caractère est décrit comme la juxtaposition de primitives globales : barres, courbes... sur lesquelles on peut opérer certaines opérations (inclinaisons, symétries). Les paramètres géométriques mémorisés sont hauteur, longeur, carrure, pente, etc... ; chaque primitive est décrite par une interpolation de son contour.

* Squelettes (KNU 78)

Le caractère est codé par son squelette, et par une fonction qui en définit l'épaisseur en tout point ; le modèle imite le déplacement sur le papier d'un pinceau d'épaisseur et d'orientation variables.



1-3-0 : Discrétisation

1 3 6 8
1 3 6 8
1 3 6 8
1 3 6 8
1 3 6 8
1 3 6 8
1 3 6 8
1 8
2 7
3 6
4 5

1-3-1 : Matrice binaire

1(*)	3(*)	6(*)	8(*)
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0(†)	0(†)	0
1			-1
1			-1
1(†)			-1(†)

* : naissance
† : mort

1-3-2 : Code par plages (plages horizontales)

1-3-3 : Codage par plages différentiel

Figure 1-3

Le tableau 1-4 synthétise les caractéristiques générales de ces différents codes.

		Taille du code	Qualité	Souplesse	Vitesse	Tampon	Balayage vidéo
SURFACE	Matrice binaire	N^2	$N \leq 50$	négligeable	très grande	0	+ + +
	Code par plages	$kN\log_2 N$	$N \leq 100$	"	grande	N	+ +
CONTOUR	Code par plages différentiel	$6kN + b[\log_2 N + c]$	$N \leq 100$	"	moyenne	N	+ +
	Code interpolé	$K \log_2 N$	$N \leq 2000$	dépend de l'interpolant	faible	N ou N^2	+
STRUCTURE	Primitives	$K'\log_2 N$	$N \leq 250$	très grande	très faible	N^2	-
	Squelettes	$K'\log_2 N$	?	grande	faible	N^2	-

Figure 1-4 : Tableau récapitulatif codage

2 - DEFINITION D'UN NOUVEAU GENERATEUR DIGITAL DE CARACTERES

A - CONSTAT INITIAL

. Local et global

Les codes qui autorisent des manipulations sur la forme des caractères utilisent une représentation très éloignée de la matrice d'origine. En effet, pour les inclinaisons, rotations... une information globale sur la structure est nécessaire ; elle ne peut être extraite des codes à dominante locale (matrice, contour), sans faire appel à des techniques complexes et coûteuses voisines de celles de la reconnaissance des formes.

L'utilisation de codes à dominante globale présente cependant deux inconvénients majeurs : d'une part les nombreux calculs limitent la vitesse de génération, et d'autre part une restitution en une passe de balayage est pratiquement impossible.

Si l'on ne veut pas tenir compte de la structure globale du caractère, les codages du contour, locaux (code par plages différentiel) ou interpolés (splines...) sont optimaux au sens de la théorie de l'information.

Il est bien sûr toujours envisageable de rajouter des informations globales à un code local pour l'adapter.

. Interpolation

On peut toujours améliorer un codage du contour par interpolation :

- en choisissant comme noeuds des points significatifs du contour qui permettent de relier le local au global (points à tangente horizontale ou verticale, points d'inflexion ou de rebroussement).

- en déterminant un interpolant canonique qui soit à la fois suffisamment simple pour faciliter le codage, minimiser les calculs requis par les opérations spéciales, augmenter la souplesse, tout en permettant une grande rapidité de décodage des contours (HOC 79).

. Vitesse et parallélisme

La vitesse effective du décodage peut être élevée, pour peu qu'on puisse découper le processus de génération en tâches indépendantes ; il suffit de prévoir leur réalisation en parallèle.

. L'état de l'art

Les machines actuelles les plus modernes utilisent en fait un codage par contour, mais elles sont limitées par le choix des interpolants.

En particulier les interpolants polygonaux produisent des angles peu esthétiques (Linotron 202) ; les interpolants circulaires sont difficiles à placer sur le contour et les interpolants d'ordre supérieur engendrent trop de calculs.

Signalons pour terminer que, faute de savoir décoder un contour suffisamment vite, les photocomposeuses de haut de gamme (Lasercomp, Digiset...) utilisent un codage par plages, extrêmement coûteux en mémoire de masse.

B - OBJECTIFS

A partir de ce constat initial, et compte tenu de son expérience dans le codage digital et le décodage des caractères d'imprimerie (COU 73) (COU 75) (HOU 78) (BLO 79) (SIC 80), l'équipe Communications Visuelles a développé un nouveau générateur de caractères.

Ce générateur utilise un code original d'interpolation du contour et une structure hautement parallèle, pour atteindre les objectifs suivants :

. Très bonne qualité :
Chaque caractère est codé sur une grille de 2000 points sur 2000 points, et peut être décodé à cette définition sans perte d'information.

. Forte compacité :
Pour une police romaine de corps de texte, 800 bits en moyenne par caractère.

. Souplesse
Tout facteur d'échelle peut être appliqué pour amener le caractère dans une grille de taille désirée, inférieure à 2000x2000 ; cette transformation se fait par réduction à partir de la représentation de base, donc sans augmentation du bruit de quantification. Les rapports peuvent être différents en X et en Y.

. Vitesse
1000 caractères par seconde, pour des caractères réduits dans une grille 100 x 100.

. Balayage
Génération du caractère en une passe de balayage vidéo.

Ces caractéristiques définissent un générateur de caractères de haut de gamme, destiné au marché de la photocomposistion, et qui s'avère nettement supérieur à la concurrence.

C - MARCHES

Nous nous sommes rendu compte que le code et l'organisation du décodeur étaient en fait compatibles avec les opérations spéciales que nous ne pensions pas prendre en charge (rotations, inclinaisons, variations de l'épaisseur du trait). Leur réalisation pourrait se faire simplement au prix d'une étape de décodage supplémentaire, et d'une réduction de la vitesse globale de la machine. Il s'agit là d'extensions de ses possibilités sans bouleversement de l'architecture.

L'indépendance des différentes phases du décodage a induit naturellement une structure fortement modulaire, qui sera analysée plus loin. L'existence de cette modularité nous a permis de décrire et d'étudier des variantes déduites de la version de base. Il s'agit alors de s'adapter à des segments de marché différents - écrans, traitement de texte, imprimantes rapides, - avec une compatibilité totale aussi bien au niveau du logiciel que du matériel. Cela est très important lorsque l'on sait que le traitement de texte est de plus en plus utilisé, à un coût de plus en plus bas ; il faut alors retrouver en numérique la souplesse du plomb et la qualité du travail d'un petit atelier de typographie, à un coût bien moindre.

Quelques exemples de variantes étudiées :

. La première expansion peut être supprimée, à condition d'interdire l'utilisation de primitives ; la taille moyenne du code passe alors de 800 à 1600 bits par caractère.

. Il est possible de faire varier le nombre de processeurs chargés du calcul effectif des points des parois, (processeurs de suivi de contour), pour adapter la vitesse de décodage, - et le prix -, à une application précise :

2 processeurs - 200 car/s - 40 kF
10 processeurs - 1000 car/s - 80 kF

. Pour des versions bas de gamme, une diminution importante du coût, et de la vitesse, sera obtenue par changement de la technologie. Ainsi, plusieurs cartes microprogrammées effectuent la deuxième phase du décodage (transformation des interpolants canoniques), de manière à garantir une vitesse de fonctionnement de 1000 car/s. Une autre possibilité utilise un microprocesseur standard et son coprocesseur de calcul, mais limite la vitesse de la machine à environ 50 car/s. Le même type de modification peut être envisagé pour la mise à l'échelle, en remplaçant un composant rapide (multiplieur TRW) par un microprogramme. Il s'agit bien sûr d'aspects divers de l'éternel dilemme matériel-logiciel ; tout algorithme peut être entièrement cablé, ou entièrement programmé ; c'est au concepteur, à l'architecte des systèmes de dégager de l'étude un compromis correct.

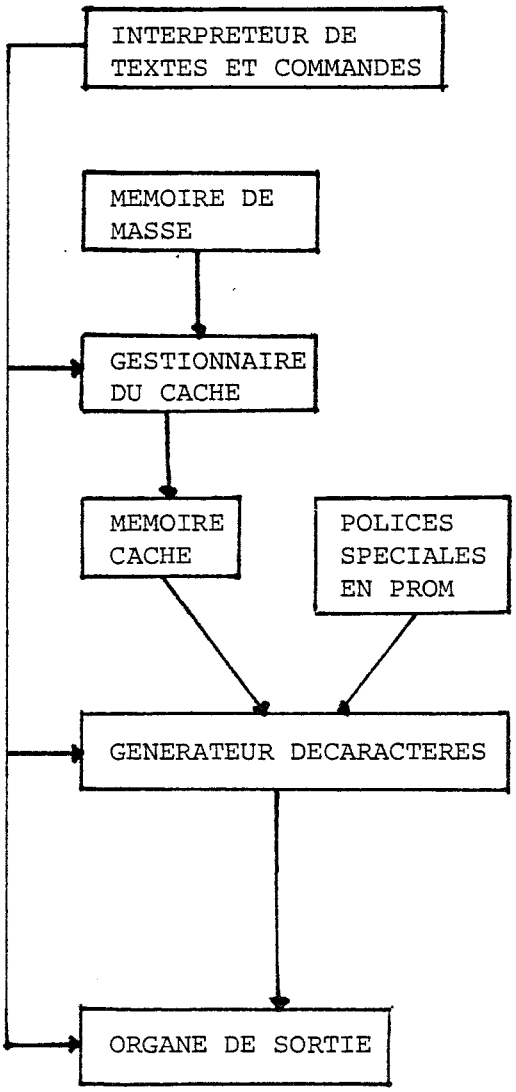
Différentes versions peuvent donc être dérivées du prototype réalisé de manière à optimiser la compacité du code, la quantité de tampon, la souplesse, la vitesse et le coût pour une application déterminée.

Normalement, la configuration complète correspond à un environnement de photocomposeuse (fig. 1-5) ; on porte l'attention sur la qualité (2000 x 2000), et la vitesse (1000 car/s). Au contraire, une version plus lente et meilleur marché peut être utilisée pour engendrer des caractères en code par plages ou matrice de points, pour une imprimante d'ordinateur ou de machine de traitement de texte (figure 1-6) ; on porte alors l'attention sur la souplesse (opérations spéciales) et la taille du code (800/1600 bits/car).

Le tableau 1-7 résume quelques applications élémentaires de notre générateur de caractères.

Type d'application	Vitesse (car/s)	Qualité maximale (points)	Souplesse	Prix (kF)
Photocomposeuse (haut de gamme)	1000	2000	mise à l'échelle X, Y	70 - 100
Photocomposeuse (bas de gamme)	200	2000	"	40 - 70
Imprimante (haut de gamme)	200	200	toutes opérations par materiel	20 - 40
Imprimante (bas de gamme)	50	200	toutes opérations par logiciel	10

Tableau 1-7



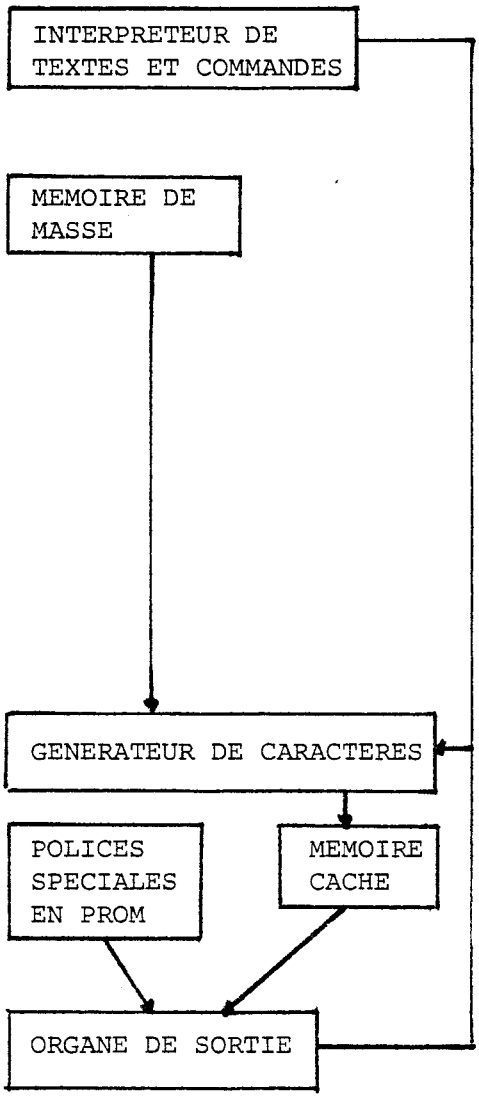
ENVIRONNEMENT DE PHOTOCOMPOSEUSE

Résolution:
De 200 à 640 lignes/cm

Taille des caractères:
De 2 à 180 points par 1/2 point

Vitesse:
De 500 à 1000 car/s

Figure 1-5



ENVIRONNEMENT D'IMPRIMANTE D'ORDINATEUR

Résolution:
De 80 à 200 lignes/cm

Taille des matrices:
De 15 à 300 points par 1

Vitesse:
De 50 à 250 car/s pour remplir la mémoire cache

Figure 1-6

On peut aller plus loin : les caractères sur lesquels nous avons travaillé jusqu'à présent ne sont autre chose que des taches bicolores régulières, et notre décodeur un générateur de taches ; on peut ainsi concevoir une application générale pour tout type de terminal à balayage vidéo : générations de caractères, symboles et logotypes ; modification d'images en noir et blanc simples et régulières par déformations géométriques pour l'animation.

Les algorithmes dégagés peuvent eux-mêmes participer, de manière autonome, aux développements de l'informatique graphique :

. le codage du contour utilise une méthode d'interpolation très simple, bien adaptée à une saisie semi-automatique ou interactive.

. le code retenu contient une information supplémentaire, destinée à rendre compatible la définition séquentielle sur contour avec l'ordre nécessaire au balayage vidéo ; elle est normalement dégagée au moment du codage, hors-ligne, a priori, dans une phase de compilation du caractère. Nous avons longuement étudié cette structure, et essayé de la déterminer de manière optimale. Ce travail n'est pas gratuit : dans le cas de certaines opérations spéciales, elle doit être recalculée en temps réel à partir de la description du contour après transformation.

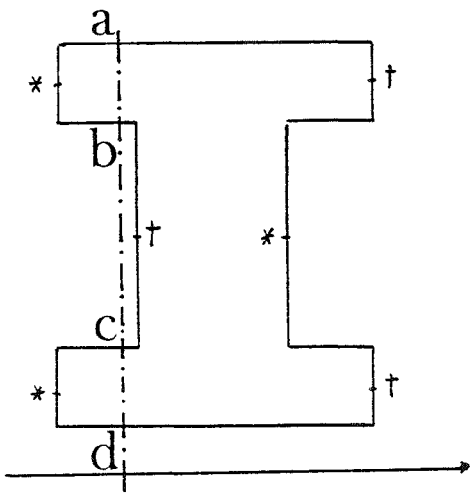
3 - PRINCIPES DE REALISATION

A - INTERPOLATION DU CONTOUR

1°) Codage du contour :

Nous avons vu que l'un des objectifs du projet est d'augmenter la finesse du caractère tout en diminuant la taille des fichiers du code en entrée. Pour arriver à une densité permettant de respecter ces deux contraintes seul un codage du contour est envisageable. Si l'on regarde la manière dont le caractère est créé

sur un tube cathodique, les seuls points qu'il est nécessaire de connaître à un instant donné sont les points d'intersections du contour et de la droite de balayage en cours. Ainsi, il est nécessaire de suivre en temps réel abscisse après abscisse les parois, portions du contour représentables comme une fonction univoque de l'abscisse. Le débit maximum est, pour une force de corps donnée, fonction de la résolution désirée, qui détermine le nombre d'abscisses à calculer, et du nombre de parois à suivre. Les opérations de suivi des parois étant rendues totalement indépendantes l'une de l'autre, il est possible de rendre le temps de calcul indépendant du nombre de parois à calculer en plaçant en parallèle autant de processeurs qu'il y a de parois. Le débit de la machine doit être de 1000 car/s pour une matrice de 100 x 100 ce qui donne un temps de calcul de 10 s par point. Les seules courbes utilisables sont les droites et les cercles pour lesquelles il existe des algorithmes rapides et précis permettant de trouver l'accroissement sur un axe en connaissant la variation sur l'autre (HOR 77) (BRE 77). Les contours d'un caractère peuvent être codés sous la forme d'une structure d'anneaux, chaque anneau est la description d'une composante connexe découpée en une suite de droites et de cercles. Sous cette forme le codage d'un signe nécessite en moyenne 3200 bits soit 64 K octets pour une police de 160 signes. Nous allons maintenant examiner deux méthodes pour augmenter la densité du code.



* : points de naissance de paroi
† : points de mort de paroi
a, b, c, d : points à calculer pour le balayage en cours

2°) Compression numérique :

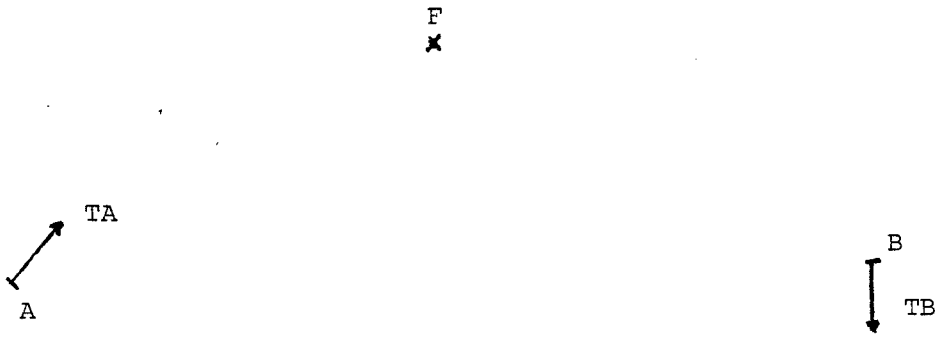
La difficulté de trouver des arcs de cercle réalisant une bonne approximation d'une courbe quelconque lors de la saisie du contour des caractères, a conduit à étudier les méthodes de découpage automatique de courbe en arcs de cercle ; c'est la notion de courbe évoluée introduite par Marc Hourdequin (HOU 78).

Le codage d'une courbe évoluée demande six mots alors que le codage de quatre cercles en nécessite seize, il apparaît donc possible de comprimer le code dans un rapport deux au prix de nombreux calculs numériques à effectuer lors du décodage. La taille des fichiers d'entrée est ainsi ramenée à 32 K octets par police. De plus, l'utilisation de mots de 16 bits alors que les coordonnées de grille sont sur 11 bits permet d'améliorer la densité du code en utilisant les quatre bits de poids forts pour coder le cas très fréquent où les tangentes des extrémités de la courbe sont verticales ou horizontales.

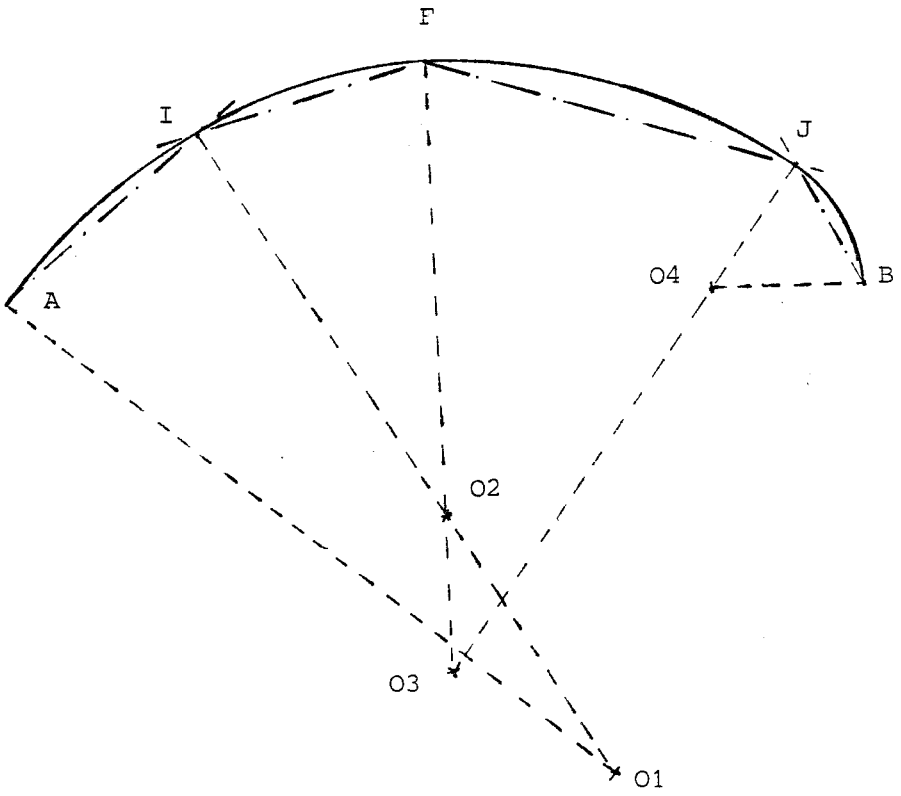
3°) Compression Linguistique

Cette compression est liée aux polices utilisées qui présentent de nombreuses portions de contour identiques. Ces éléments graphiques répétitifs, appelés primitives, peuvent être codés séparément. Lors de la rencontre d'un de ces éléments dans un contour, le codage du contour est remplacé par une simple référence à la primitive. Il est également possible de préciser des opérations simples telles que la translation, les symétries par rapport à X et Y et l'inversion du code afin d'utiliser au mieux cette notion. L'efficacité de cette compression est très variable d'une police à l'autre et suivant celle-ci la taille du fichier variera de 12 à 16 K octets (fig. 1-8).

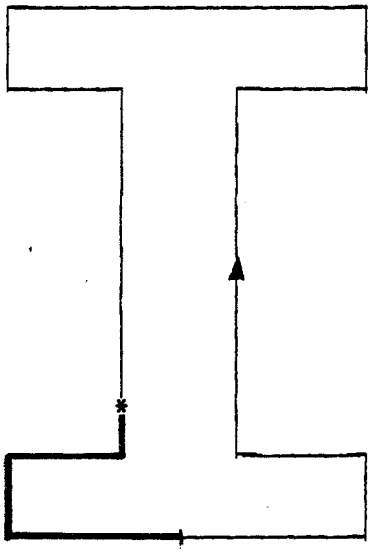
PARAMETRES D'ENTREE



RESULTATS



COURBE EVOLUEE



→ sens de parcours

* début primitive

PRIMITIVE EMPATTEMENT

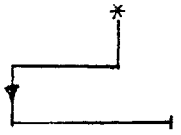
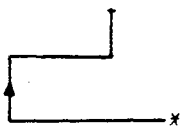
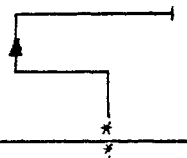
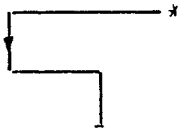
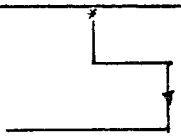
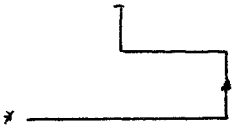
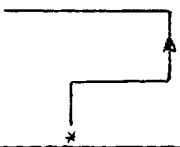
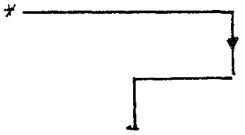
	CODE NON INVERSE	CODE INVERSE
SANS SYMETRIE		
SYMETRIE /OX		
SYMETRIE /OY		
SYMETRIE /O		

Figure 1-8

STRUCTURE D'ARBRE

1 POINTEUR
N° ORDRE N° PAROI N° ORDRE N° PAROI
0 ABSCISSE DE NAISSANCE
1 POINTEUR
N° ORDRE N° PAROI N° ORDRE N° PAROI
1 POINTEUR
N°ORDRE N° PAROI N° ORDRE N° PAROI
"
0 ABSCISSE FIN DE LETTRE

STRUCTURE D'ANNEAU

000 X DROITE
Y DROITE
100 XA
YA
TA
XF
YF
TB
XB
YB
(COURBE EVOLUEE)
111 REFERENCE
TRANSLATION X
TRANSLATION Y
(PRIMITIVE)
010 FIN D'ANNEAU
"
110 FIN DE LETTRE

(ANNEAU)

CODE D'ENTREE

B - STRUCTURES DE PARALLELISME

1°) Mise en parallèle des parois

L'opération de suivi des parois du caractère à partir de la seule description des contours contenue dans la structure d'anneau demanderait une exploration préliminaire du fichier afin d'acquérir des informations sur la structure globale de la lettre. La vitesse demandée à la machine ne permet pas ces calculs préliminaires. Une structure d'arbre placée en tête du fichier, contient toutes les informations nécessaires pour initialiser les processeurs de suivi de contour, et les rendre autonomes. Ces informations sont l'abscisse de naissance, un pointeur dans la structure d'arbre, le numéro de paroi et le numéro d'ordre.

- L'abscisse de naissance indique aux processeurs à quel moment ils doivent commencer le calcul de la paroi.

- Le pointeur dans la structure d'anneau indique où se trouve les paramètres concernant la paroi.

- Le numéro de paroi permet au processeur qui doit prendre en charge la paroi de se reconnaître. Ce numéro évite d'avoir une gestion dynamique de la charge des processeurs de suivi de contour. On remplace une allocation dynamique par une auto-allocation précalculée. Chaque processeur reçoit à la mise en route le nombre de processeurs présents et un numéro de référence puis il calcule les numéros des parois qu'il devra reconnaître.

- Le numéro d'ordre permet de classer les ordonnées avant de les transmettre à l'organe de visualisation.

Toutes ces informations sont calculées lors de la saisie du caractère et, à part les abscisses de naissances qui sont mises à l'échelle, elles ne sont pas modifiées lors du décodage.

Les pointeurs dans la structure d'anneau sont déterminés de façon à pointer les coordonnées du point de naissance après les expansions du code correspondant aux deux compressions.

2°) Pipe-line de décodage et modularité

Il est impossible de concevoir une machine à un seul étage capable de décoder les caractères en respectant le débit de 1000 car/s. Le décodage peut facilement être découpé en trois étapes totalement indépendantes et séquentielles, ce qui permet d'utiliser une structure pipe-line dans laquelle chaque étage correspond à une étape d'expansion du code.

La machine a donc été construire sous une forme très modulaire. La seule étape nécessaire pour garder un codage par contour est l'étape 3 qui réalise le suivi des contours. Les autres étapes réalisent une expansion du code et ne sont intéressantes que pour obtenir une densité très élevée en entrée. L'étape intermédiaire qui réalise la mise à l'échelle est bien entendu obligatoire dans toute application industrielle ; il faut remarquer également que la mise à l'échelle indépendante en abscisse et en ordonnée rend nécessaire l'étape 2 qui transforme les courbes évoluées en cercles.

En plus de cette modularité en nombre d'étage, il existe une modularité au niveau de l'étape 3 qui autorise de deux à dix cartes travaillant en parallèle avec, bien entendu, une répercussion sur le débit de cette étape. Une autre possibilité, que nous réalisons, est de remplacer l'étape 2 relativement complexe (4 cartes) mais rapide par une carte utilisant le 8087, coprocesseur du microprocesseur 8086 d'Intel, ce qui entraîne un débit vingt fois plus faible. Les versions qu'il est possible de réaliser à partir de la machine vont de deux cartes si on accepte une densité de code non maximale à 18 cartes si on désire un code très compact et une machine très rapide. Les diverses versions sont indiquées dans le tableau qui suit. A l'intérieur d'une colonne, le code des caractères en entrée est inchangé.

Pour arriver à la première ligne, la fonction de contrôle général est incluse dans la carte de mise à l'échelle et la carte de tri en sortie est supprimée en considérant que l'unique processeur de suivi de contour effectue les calculs des ordonnées dans l'ordre de leur transmission.

Densité du code

	800 bits/car	1600 bits/car	3200 bits/car (1)
CAR/s	nombre de cartes		
20	4	3	2
50	7	6	-
200	10	9	5
300	12	11	7
1000	18	17	13

(1) impose échelle X = échelle Y

3°) Variantes

La structure pipe-line et le code utilisé offrent une grande souplesse et permettent de rajouter des opérations spéciales sans bouleverser la partie déjà réalisée.

Toutes ces opérations : épaississement, inclinaison et rotation imposent de modifier la structure d'anneau et la structure d'arbre. Dans la structure d'anneau elles modifient les coordonnées et les angles et la rotation impose en plus une rotation de chaque anneau afin qu'il débute par un point de mort ce qui simplifie leur exploitation. Dans la structure d'arbre les deux

premières opérations n'introduisent qu'une modification des abscisses de naissance alors que la rotation demande de recalculer les points de naissance, dont le nombre peut varier, ainsi que les pointeurs et numéros d'ordre et de paroi qui leur sont associés. Si l'on étudie l'effet produit par toutes ces opérations appliquées à un caractère on constate que celles-ci ne sont pas commutatives, il est donc nécessaire de fixer une priorité entre elles de façon à obtenir un résultat qui semble naturel à l'utilisateur. La modification de la structure d'arbre pour toutes ces opérations peut être réalisée sur une seule carte placée, dans la structure de la machine réalisée, après l'étape de calcul des courbes évoluées.

Les modifications dans la structure d'anneau peuvent être réalisées grâce à une carte par opération placée dans l'ordre des priorités entre opérations. Pour des versions lentes ou n'utilisant que rarement ces opérations spéciales, il est possible des les regrouper sur une ou deux cartes en fonction des caractéristiques souhaitées. L'épaississement et l'inclinaison peuvent être rajoutés à la maquette sans modifier le travail réalisé ; pour introduire la rotation il serait nécessaire de modifier le logiciel de l'étape de transformation de courbes évoluées en cercles afin de calculer les points à tangentes verticales (naissances ou morts) ou à tangentes horizontales (changement de cercle) à l'intérieur des courbes évoluées.

4 - RESULTATS OBTENUS

A - STRUCTURE EFFECTIVE

Pour certifier le générateur de caractère décrit plus haut, il a été décidé de se limiter à un débit de 500 car/s. C'est la vitesse des photocomposeuses modernes sur le marché (LINOTRON 202, COMPUGRAPHIC 8600). La description de la maquette est donnée étape par étape et le schéma fonctionnel est détaillé au niveau de la carte.

1°) Expansion linguistique (Etape 1)

Au cours de cette étape les références aux primitives indiquées dans le fichier d'entrée sont remplacées par le code du contour correspondant, translaté, modifié par les symétries et inversé si nécessaire. Cet étape réalise le transfert d'environ 200 mots avec addition sur 100 ; si nous considérons un temps d'addition égal au temps de transfert, cela nous donne 300 opérations élémentaires à réaliser en 2 ms. Cette carte est réalisée à partir d'un microprocesseur 16 bits (le 8086 d'Intel) qui nous permet de manipuler l'information par mot et rend l'accès à la base de données très simple.

Pour une application utilisant plus d'une dizaine de polices la taille du fichier d'entrée impose de le placer sur disque, sur disquette ou sur mémoire à bulles. Un gestionnaire de la mémoire de masse recopie à la demande une police dans une mémoire tampon qui sera exploitée par l'étape 1. Pour la maquette nous n'avons qu'un petit nombre de caractères d'essai, ce qui nous a permis de placer le fichier d'entrée sur mémoires EPROM implantées sur la carte de l'étape 1.

2°) Expansion numérique (Etape 2)

Au cours de cette étape chaque courbe évoluée est remplacée par quatre arcs de cercles. Une courbe évoluée est une courbe de classe C^1 sans point d'inflexion et dont la variation de l'angle du vecteur tangent est inférieure à π. Elle est définie dans le fichier d'entrée par ses deux extrémités A et B, les angles des tangentes en ces points TA et TB, et le point de carrure F. Le remplacement de la courbe par quatre arcs de cercles demande le calcul de deux points intermédiaires I et J respectivement compris entre AF et FB, et des centres des quatre cercles. Les calculs préliminaires donnent les angles des tangentes en F, I et J ; ces deux derniers points étant considérés comme les points de carrure des courbes AF et FB. Le calcul des coordonnées des 6 points est effectué par la résolution de 6 systèmes linéaires 2x2 dont les coefficients sont fonction des angles des tangentes à la courbe.

Afin de respecter le débit de la machine on a réalisé un sous pipe-line de calcul comprenant 2 cartes microprogrammées à base de microprocesseurs en tranche (2901 de AMD). La première réalise les calculs préliminaires c'est-à-dire le calcul des tangentes en F, I, J par division et l'accès à des tables trigonométriques pour obtenir les coefficients des systèmes linéaires. La deuxième résoud les systèmes linéaires par une suite de cinq multiplications qu'elle sous-traite à une carte réalisant la multiplication en virgule flottante. Ce sous pipe-line est alimenté par une carte de lecture et de tri qui transmet à la partie calcul les paramètres d'entrée des courbes évoluées et poursuit, en parallèle avec la partie calcul, l'exploitation du fichier d'entrée.

3°) Suivi de contour (Etape 3)

Au cours de cette étape la représentation séquentielle du contour est transformée en une représentation par matrice creuse créée colonne après colonne.

Cet étage est composé de 2 à 10 processeurs de suivi de contour ; afin de pouvoir suivre toutes les parois, chaque processeur en calcule un nombre variable en fonction de la configuration présente. A la mise en route de la machine chaque processeur reçoit un numéro et le nombre de processeurs présents et détermine les numéros des parois qu'il prendra en charge. Cette procédure de repliement présente 3 avantages :

- La possibilité d'effectuer la mise au point avec seulement 2 cartes ;

- Le repliement automatique qui lors de la mise en route de la machine permet par un programme d'autotest de configurer cet étage en fonction des cartes saines ;

- La possibilité de produire des machines ayant des débits variant dans un rapport 10 sans changer le matériel ni le logiciel.

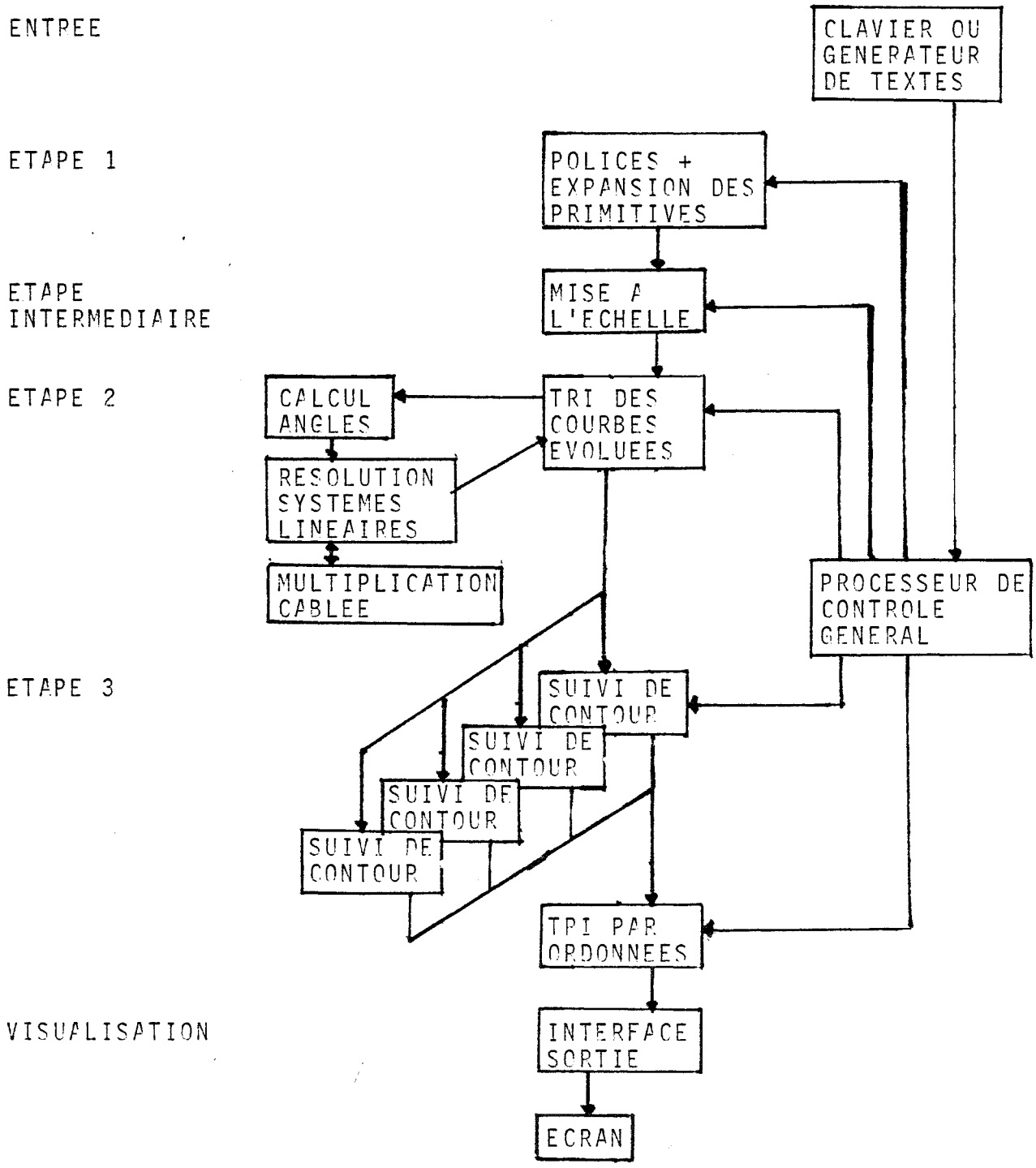
En contrepartie, il est nécessaire d'associer à chaque paroi un numéro (numéro de paroi) qui permet au processeur concerné de se reconnaître. De même, le problème de classement des ordonnées a été résolu en associant à chaque paroi un numéro d'ordre. Une carte réalisée en logique câblée effectue rapidement le tri par adressage à l'aide du numéro d'ordre lors de la saisie des résultats et adressage en séquence lors de leur transmission.

4°) Mise à l'échelle (Etape intermédiaire)

Au cours de cette étape le caractère est réduit afin d'être placé dans une grille de taille désirée ; le format de la grille peut aller de 2048 x 2048 à 30 x 30. Le nombre élevé de coordonnées à mettre à l'échelle nous a conduit à utiliser un multiplieur rapide, la division par un nombre constant etant remplacée par une multiplication suivie d'un décalage. La possibilité d'avoir une échelle différente en X et en Y empêche d'effectuer la mise à l'échelle après l'étape 2 à cause des cercles, qui seraient transformés en ellipses ; celle-ci est donc placée après l'étape 1. Il est alors nécessaire de modifier les angles en fonction du rapport des échelles ; cette modification est réalisée grâce à une table et limite à 32 le nombre de rapports d'échelle possibles, pris entre 0,5 et 2.

5°) Synchronisation

Les temps de calcul des étapes 1, 2 et intermédiaires ne dépendent que du caractère à traiter ; par contre, le temps de calcul de l'étape 3 dépend également de l'échelle adoptée. De plus, l'origine du temps de calcul critique est fonction de la suite de caractères engagés dans le pipe-line ; il est ainsi impossible de connaître a priori l'étape critique. En outre, pour des raisons de simplicité la synchronisation a été réservée à un processeur indépendant PCG. Ce dernier supervise les tests de



SCHEMA FONCTIONNEL DE LA MAQUETTE

fonctionnement correct à la mise en route de la machine, il transmet aux processeurs de suivi de contour les informations nécessaires au recouvrement, il indique à l'étape 1 le numéro du caractère désiré, il calcule le rapport d'échelle, de plus il pourra gérer la mémoire de masse des polices située sur disque et le positionnement des caractères sur la page (justification, passage à la ligne).

6°) Réalisation

La technique du wrapping a été utilisée pour le cablage afin d'autoriser facilement les modifications de câblage. Les cartes de l'étape 3 et PCG ont été réalisées à partir de cartes prépercées au format européen 6U (160 mm x 233 mm). Les autres cartes ont été réalisées à partir de cartes (230 mm x 233 mm) ayant un plan de masse côté composants et les alimentations imprimées côté cablage. Ces cartes ont été placées dans deux racks, l'un contenant l'étape 3 formée de 2 à 8 cartes de processeurs de suivi de contour, l'autre contenant le reste de la machine.

La connexion entre cartes d'un même rack a été réalisée par un fond de panier wrappé. La connexion entre les deux racks ou avec la boîte de commande et le kit d'interface en sortie a été réalisée par des câbles plats à 16 fils.

B - PERFORMANCES

Le code utilisé est pratiquement identique à celui certifié par Marc HOURDEQUIN (HOU 78) grâce à un programme de simulation réalisé sur un ordinateur PDP 11/40. D'après les estimations réalisées sur les polices Baskerville et Univers, ce code atteint une compacité de 800 bits/car en moyenne. La principale différence entre le code utilisé et le code de Marc HOURDEQUIN est l'introduction du numéro de paroi et du numéro d'ordre. La qualité apportée par l'utilisation d'une grille de définition de 2000x2000 peut être appréciée sur les planches de caractères présentées et en particulier sur le U Baskerville.

La souplesse du codage et l'effet produit par la mise à l'échelle différente en X et en Y peuvent être appréciés sur la planche présentant le sigle de l'Ecole des Mines de Saint-Etienne.

La mesure de la vitesse de la machine est rendue complexe par le fait que plusieurs paramètres interfèrent à chaque étape pour donner le temps de calcul : cette remarque nous a conduits à concevoir des caractères spéciaux pour effectuer les mesures. Les principaux paramètres sont :

- Pour l'étape 1
  - La longueur totale du fichier du caractère
  - Le nombre de primitives utilisées pour coder le caractère

- Pour l'étape intermédiaire
  - La longueur totale du fichier du caractère
  - Le nombre de droites sur le contour
  - Le nombre de courbes évoluées sur le contour

- Pour l'étape 2
  - La longueur du fichier
  - Le nombre de droites
  - Le nombre de courbes évoluées

- Pour l'étape 3
  - Le nombre de points de naissances
  - L'échelle en X (nombre d'abscisses à engendrer)
  - Le nombre de droites
  - Le nombre de cercles
  - Le taux moyen de recouvrement (nombre de parois par processeur)

Ces mesures ont été effectuées avec les temps de cycles suivants :

- Etape 1		250 ns
- Etape intermédiaire		300 ns
- Etape 2	Partie supérieure	210 ns
	Partie 1	260 ns
	Partie 2	230 ns
- Etape 3		250 ns

L'interprétation des mesures a conduit aux résultats suivants :

- Etape 1

  L : longueur en mots de 16 bits de la structure d'arbre
  l : nombre de mots hors primitive
  p : longueur totale en mots du code des primitives
  n : nombre de primitives

  $T1\ (\mu S) = 4L + 51 + 30n + 15p$

- Etape intermédiaire

  a : longueur en mots de la structure d'arbre + nombre de contours distincts
  b : nombre de droites
  c : nombre de courbes évoluées

  $TI\ (\mu s) = 1,5a + 3,75b + 9c$

- Etape 2

  a : longueur en mot de la structure d'arbre + nombre de contours distincts
  b': nombre de droites avant la 1ère courbe évoluée + pour chaque courbe évoluée le nombre de droites au dessus de 15 avant la courbe suivante (en général b' est réduit au nombre de droites avant la 1ère courbe évoluée).
  c : nombre de courbes évoluées

  $T2\ (\mu s) = 1,2a + 3,8b' + 42 \times (C \neq 0) + 58c$

- Etape 3

e : nombre de points de naissance
X : nombre d'abscisses
M : nombre moyen de paroi par abscisses
N : nombre de processeurs disponibles
b": nombre de droites après mise à l'échelle (b"=b pour l'échelle 1x1)
c": nombre de cercles après mise à l'échelle (c"=4c pour l'échelle 1x1)

dans le cas N=10 $T3\ (\mu s)=105e + 1/M(32b"+82c")+18X$ (1)

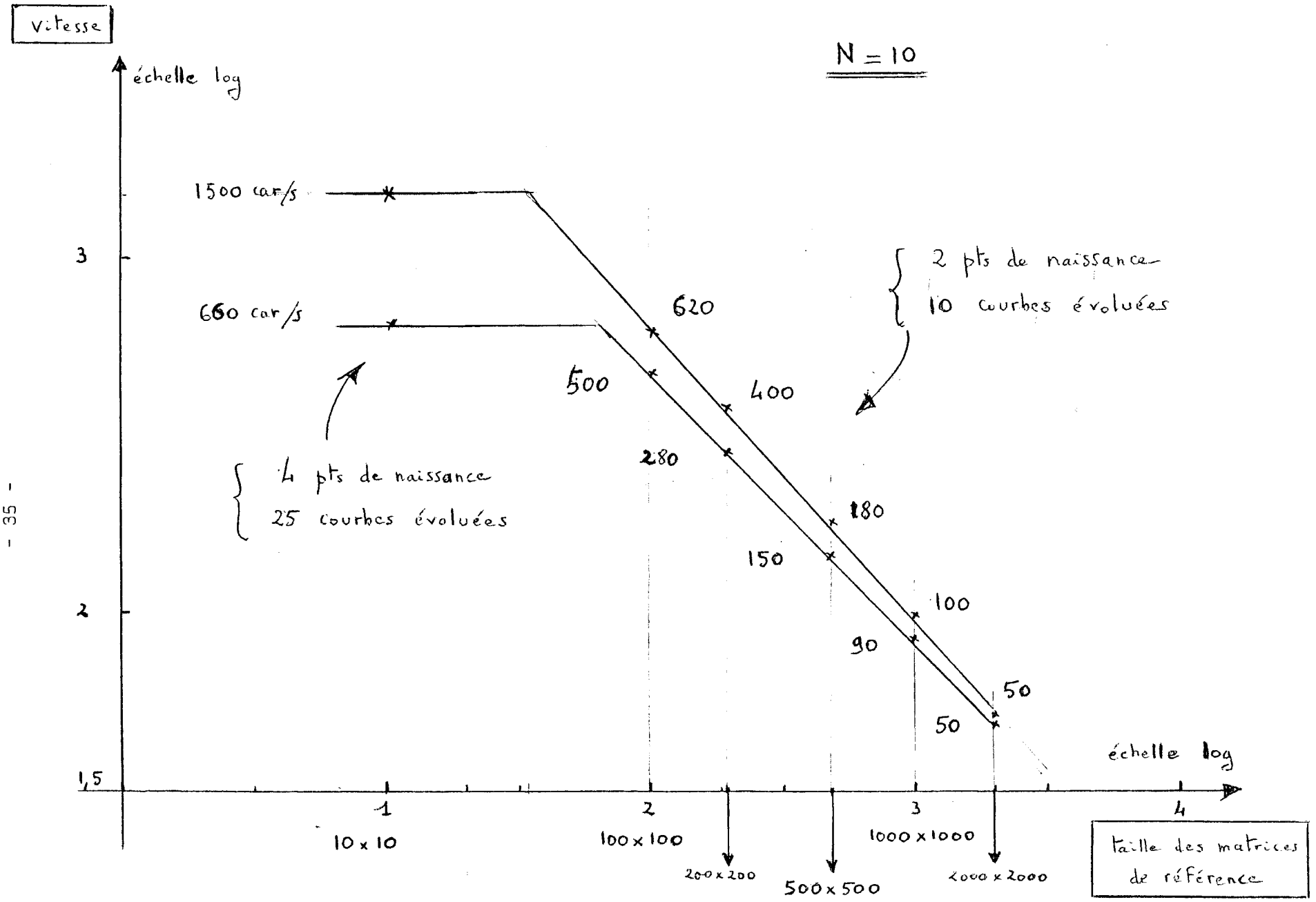
dans le cas $N < M$ $T3\ (\mu s)=250e + 1/N(32b"+82c")+M/N\ 38X$ (2)

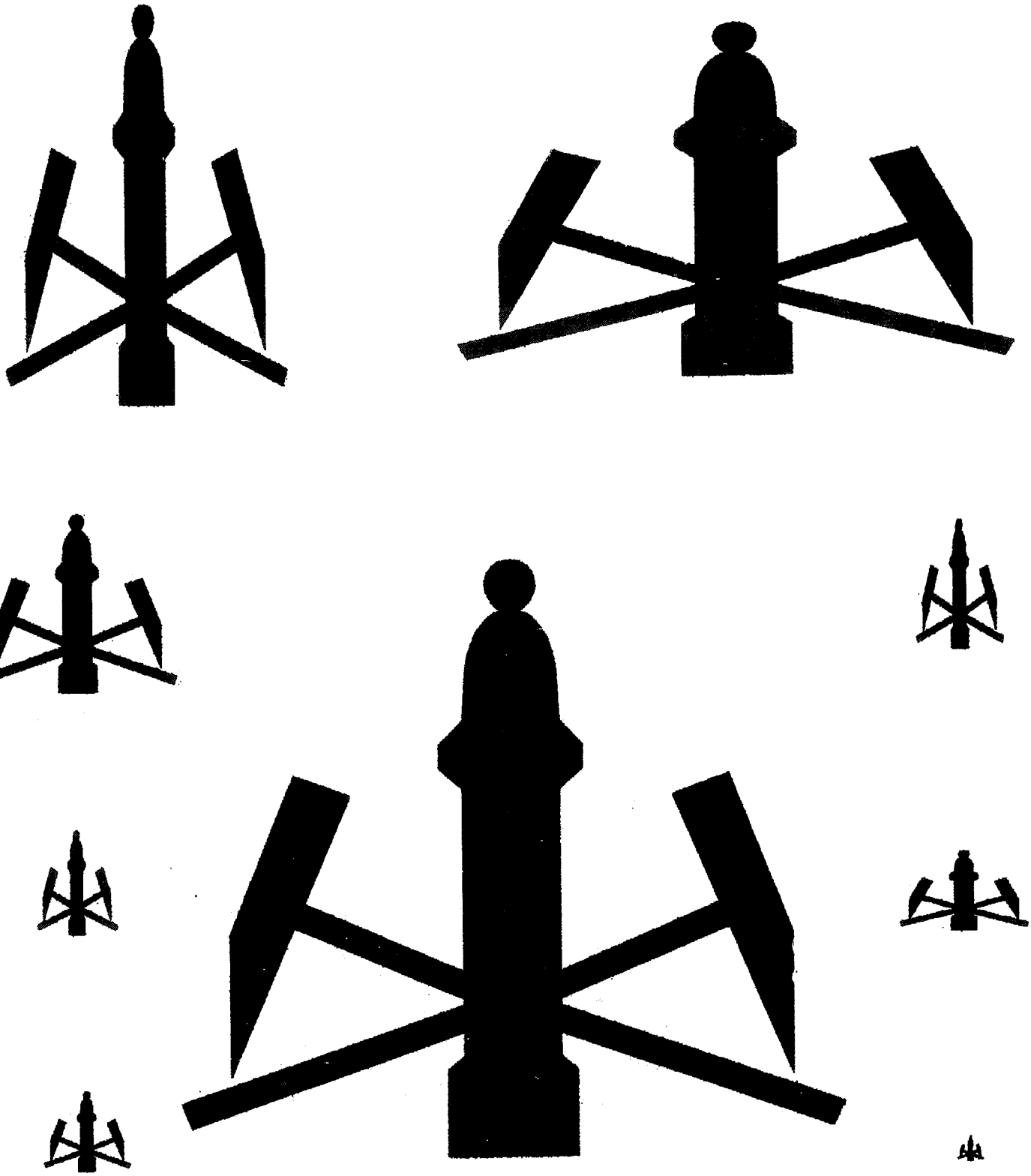
La comparaison de (1) et (2) montre que la gestion de plusieurs parois par processeur dans l'étape 3 est pénalisante en temps. De fait, le contexte des calculs ne peut plus être constamment gardé dans les registres internes du microprocesseur et la vitesse de calcul est réduite de moitié. Ce facteur s'ajoute évidemment au surcroit de travail de chaque processeur mesuré en moyenne par M/N. Si l'on estime M voisin de 5, la vitesse est divisée approximativement par un facteur 3 pour N=4 et par un facteur 4 pour N=2.

Bien qu'elles soient approchées, les formules des temps de calculs sont représentatives des performances du générateur de caractères.

Afin de présenter les résultats sous une forme plus lisible, on a tracé les courbes de vitesse, pour le cas N=10 en fonction d'une échelle homogène en X et en Y, pour deux caractères de complexités différentes couvrant bien la gamme des polices de corps de texte.

On voit que pour une matrice de référence de 100x100, la vitesse du générateur de caractère est de 500 caractères par seconde, ce qui atteint l'objectif visé. D'autre part, la maquette fait appel à la version lente des microprocesseurs utilisés ; le gain théorique est de 2 si on utilise les versions rapides des microprocesseurs et des mémoires associées ; la vitesse du générateur de caractère pourrait donc atteindre les 1000 caractères à la seconde, vitesse de référence des photocomposeuses de haut de gamme.


Vitesse
échelle log
N = 10
1500 car/s
660 car/s
3
2
1,5
620
500
400
280
180
150
100
90
50
50
2 pts de naissance
10 courbes évoluées
4 pts de naissance
25 courbes évoluées
échelle log
1
10 x 10
2
100 x 100
200 x 200
500 x 500
3
1000 x 1000
2000 x 2000
4
taille des matrices de référence



ABCDEFGHIJKLMN
OPQRSTUVWXYZ

abcdefghijklmnop
qrstuvwxyz

Epreuve imprimante, rapport de rēduction 1/1
Rēduction photographique 1/9

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrs tuvwxyz

Epreuve imprimante, rapport de rēduction 1/1
Rēduction photographique 1/18

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvw xyz

Epreuve imprimante, rapport de rēduction 1/2
Rēduction photographique 1/9

ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz

Epreuve imprimante, rapport de rēduction 1/2
Rēduction photographique 1/18

UNIVERS MEDIUM 689

&ÆŒœ 0123456789

nopqrstuv ".,:!?

GHIJKLMNO

Epreuve imprimante, rapport de réduction 1/1
Réduction photographique 1/9

&ÆŒœ ".,:!?? 0123456789

nopqrstuv GHIJKLMNO

Epreuve imprimante, rapport de réduction 1/1
Réduction photographique 1/18

&ÆŒœ ".,:!? 0123456789

nopqrstuv GHIJKLMNO

Epreuve imprimante, rapport de réduction 1/2
Réduction photographique 1/9

&ÆŒœ ".,:!?? 0123456789 *nopqrstuv* *GHIJKLMNO*

Epreuve imprimante, rapport de réduction 1/2
Réduction photographique 1/18

ABCDEFGHI JKL
MNOPQ RSTUV
WXYZ

abcdef ghi jklmn op
qrstuvwxyz

Epreuve imprimante, rapport de réduction 1/1
Réduction photographique 1/9

ABCDEFGHI JKLMN OPQ RST

UVWXYZ

abcdef ghi jklmnopqrstuvwxyz

Epreuve imprimante, rapport de réduction 1/1
Réduction photographique 1/18

BASKERVILLE 169

ABCDEFGHIJKLMNOPQRSTU

VWXYZ

abcdef ghi jklmn opqrstuvwxyz

Epreuve imprimante, rapport de rēduction 1/2
Rēduction photographique 1/9

ABCDEFGHIJKLMN OPQRST UVWXYZ

abcdef ghi jk lmn opqrst uvwxyz

Epreuve imprimante, rapport de rēduction 1/2
Rēduction photographique 1/18

BASKERVILLE 169

Chapitre 2

CHAPITRE 2.

INTRODUCTION AUX PROBLEMES DE NATURE ORDINALE.

1.- POSITION DU PROBLEME.

2.- LA NOTION DE PAROI .

A.- Definition.

B.- Indépendance logique des parois.

C.- Parallélisme effectif du suivi des parois.

3.- ASPECTS METHODOLOGIQUES.

A.- Un exemple d'architecture.

B.- Intérêt et difficulté d'un repliement.

C.- Architecture retenue.

- I Allocation parois-processeurs.
- II Tri des parois.
- III Calcul de l'information supplémentaire.

D.- Autres archictectures possibles.

4.- CONCLUSION.

1.- POSITION DU PROBLEME

Dans une machine de composition, le générateur de caractères ne remplit qu'une fonction, même si elle est essentielle : il doit donc être considéré comme une boîte noire dont les entrées et les sorties s'adaptent à un environnement particulier. Les spécifications d'entrée semblent bien définies : indication du caractère (police et lettre dans la police), et des opérations à lui faire subir (mise à l'échelle par exemple). Par contre les spécifications de sortie dépendent fortement de la technique de visualisation envisagée ; nous avons pensé que la production d'un code par plages nous offrait la plus grande généralité, puisqu'il permet la génération à grande vitesse des caractères en une seule passe de balayage sur un support à mémoire (film pour les photocomposeuses, tampon de mémoire intermédiaire pour les imprimantes rapides et les écrans de traitement de texte.)

Le générateur de caractère que nous avons réalisé produit du code par plages à partir d'une description très dense du contour, et c'est l'aval de la machine, l'Etape 3, qui, d'un code interpolé du contour où ne figurent plus que des paramètres d'arcs de cercle et de segments de droite, effectue le décodage final (fig 2-1)

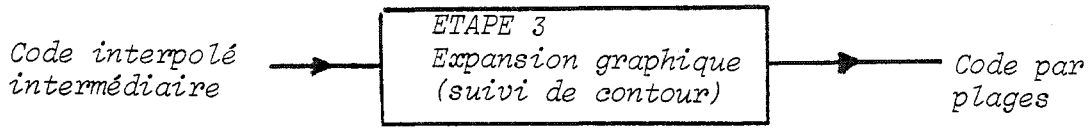


Figure 2-1 : Fonction de l'Etape 3.

Le contour d'un caractère est formé d'une ou plusieurs composantes connexes, les **anneaux**, décrits dans le sens direct, le **sens de description de l'anneau**, à partir d'une origine pour le moment quelconque. Cette structure est insuffisante pour engendrer le caractère en une seule passe de balayage ; elle ne contient pas les informations nécessaires au calcul, à toute abscisse, des ordonnées des points d'intersection du contour avec la droite de balayage, dans l'ordre de balayage.

2.- NOTION DE PAROI

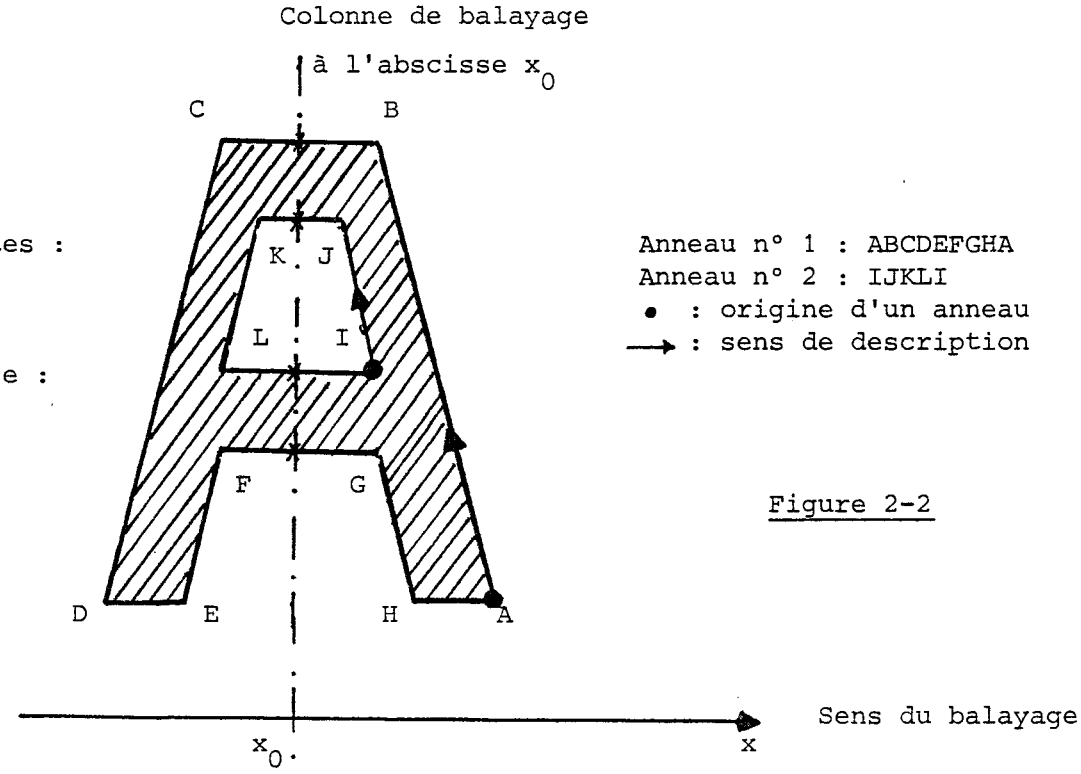
A - Définition.

C'est la notion de paroi, dégagée dans la première partie, qui sera notre outil principal pour résoudre le problème. Une définition mathématique précise en sera donnée dans le chapitre IV ; rappelons simplement que, dans un anneau, c'est une partie maximale monotone en X, comprise entre un point de naissance et un point de mort ; elle est directe si elle doit être calculée dans le sens de description du contour, inverse sinon (figure 2-2).

A une abscisse donnée, une paroi est active si elle a une intersection avec la colonne de balayage ; sinon elle est inactive. Remarquons qu'une fois inactivée, une paroi ne sera jamais réactivée.

B - Indépendance logique des parois.

Pour répérer les abscisses de naissance et les parois qui y sont activées, une information supplémentaire est nécessaire, que le suivi des seules parois actives ne permet pas de calculer au vol ; elle doit être créée **a priori** soit par la station d'édition, soit lors d'une phase préliminaire de l'opération de décodage. Nous supposerons que cette information, appelée structure d'arbre ou liste des naissances, est incluse dans le code, et cela au prix d'une redondance minime. Elle contient la liste des abscisses de naissance, et pour chacune d'elles, pour chaque



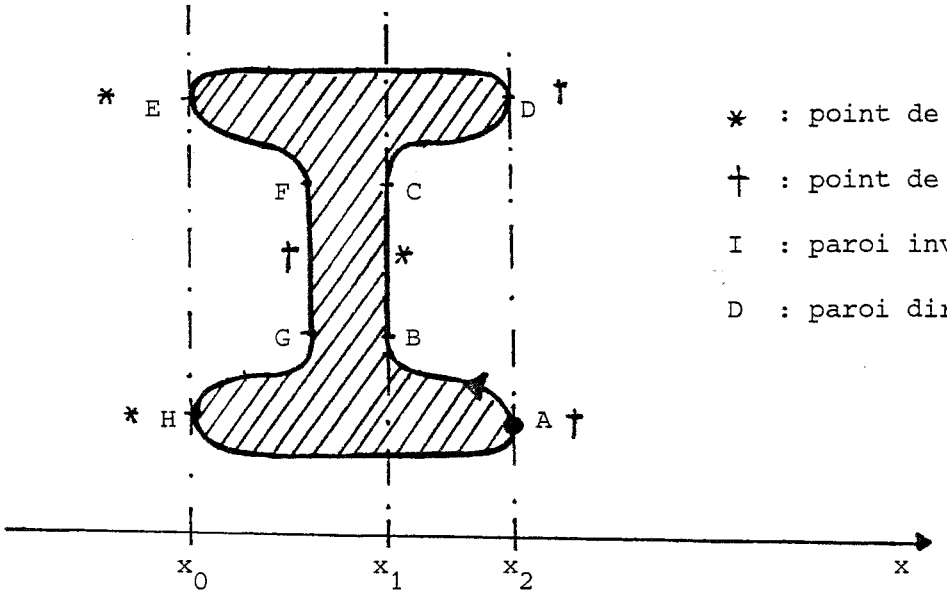
Parois directes :
DEFGHA
LI

Parois inverse :
DCBA
LKJI

* : points à calculer en $x = x_0$

Anneau n° 1 : ABCDEFGHA
Anneau n° 2 : IJKLI
• : origine d'un anneau
⟶ : sens de description

Figure 2-2



* : point de naissance

† : point de mort

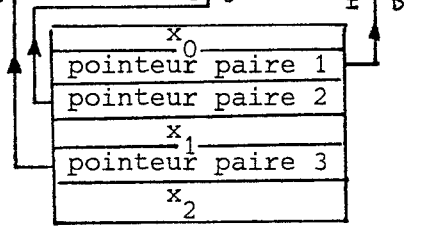
I : paroi inverse

D : paroi directe

Figure 2-3

Description de l'anneau : †$\overset{\frown}{AB}$ * $\overset{\frown}{CD}$ † $\overset{\frown}{DE}$ * $\overset{\frown}{EF}$ † $\overset{\frown}{GH}$ * $\overset{\frown}{HA}$

Structure d'arbre



paire de parois alors activées, un pointeur sur le point de naissance dans l'anneau correspondant, ainsi que des indications diverses qui seront explicitées ultérieurement (figure 2-3). Remarquons que la mort de chaque paroi, donc son point de désactivation, peut en revanche être détectée lors du suivi de cette paroi par passage par un maximum local en X.

C - Parallélisme effectif du suivi des parois.

La liste des naissances permet de considérer les parois indépendamment les unes des autres ; à ce parallélisme logique nous allons faire correspondre un parallélisme d'exécution, au sens informatique du terme. En effet, si l'on veut satisfaire au cahier des charges : 1000 caractères de taille 100 X 100 par seconde, il faut à chaque abscisse calculer en 10 μs environ toutes les intersections de la colonne de balayage avec les parois actives. Or le calcul d'un seul point d'arc de cercle ou de segment de droite, par un algorithme incrémental (HOR 77) (BRE 77) demande environ 10 μs sur un microprocesseur 16 bit récent (nous verrons un peu plus loin qu'en fait l'ensemble des calculs pourrait s'effectuer en 20 μs). Il semble donc naturel d'affecter les différentes parois actives à un moment donné à des processeurs distincts pour permettre leur éxécution simultanée. Bien sûr, si l'on accepte des performances plus faibles, et nous avons remarqué au chapitre I que cela n'était pas sans intérêt, il reste toujours interessant de pouvoir affecter plusieurs parois actives à un même processeur, à condition d'équilibrer les charges entre les microprocesseurs présents.

Ici apparaissent plusieurs problèmes sur lesquels nous aurons à revenir :

- Comment allouer les parois actives aux processeurs effectivement présents, en minimisant le temps de génération ?
- Dans le cas où un processeur prend en charge plusieurs parois actives, quelle est l'organisation logicielle la mieux adaptée à leur éxécution sur ce processeur?
- Comment synchroniser l'ensemble pour obtenir à chaque abscisse, les ordonnées de tous les points calculés ?
- Pour obtenir réellement du code par plages, il faut à chaque abscisse, trier

les ordonnées calculées par les différents processeurs, puisque rien, ni dans les anneaux ni a priori dans la liste des naissances, ne présume de l'ordre des parois.

Pour comprendre comment nous avons résolu l'ensemble de ces points, le meilleur moyen nous semble être la restitution du cheminement méthodologique qui nous a mené des premières études à la réalisation d'un prototype pleinement opérationnel.

3. - ASPECTS METHODOLOGIQUES

A - Un exemple d'architecture.

La première architecture étudiée est contenue en germe dans HOU 78 : le nombre de processeurs est égal au nombre maximum de parois actives simultanément sur le caractère le plus complexe à engendrer, soit 10 pour les polices romaines de corps de texte. Le mécanisme d'allocation qui en découle est extrêmement simple - une paroi au plus par processeur - : un processeur de contrôle de la 3e étape (PC3) examine la liste des naissances et gère une table d'occupation des processeurs de suivi de contours PSC_i (i=1...10). A une abscisse de naissance, il alloue les nouvelles parois aux processeurs libres. La désactivation d'un PSC_i par mort de la paroi qu'il calculait doit être indiquée à PC3 pour mise à jour de la table d'occupation. Le tri est effectué à chaque abscisse par un processeur de tri PT qui récupère et classe les ordonnées des points calculés par les PSC_i. La synchronisation au niveau des abscisses est élémentaire et centralisée.

Dans un environnement multiprocesseur, il faut éviter de perdre du temps en protocoles de cession et de prise de bus ; nous avons donné à ce problème une solution radicale : chaque processeur dispose en propre, dans un tampon d'entrée mis à jour par l'étape précédente, de l'ensemble du code intermédiaire du caractère à engendrer ; ainsi, il n'y a jamais contact entre les bus des différents processeurs. (fig 2-4).

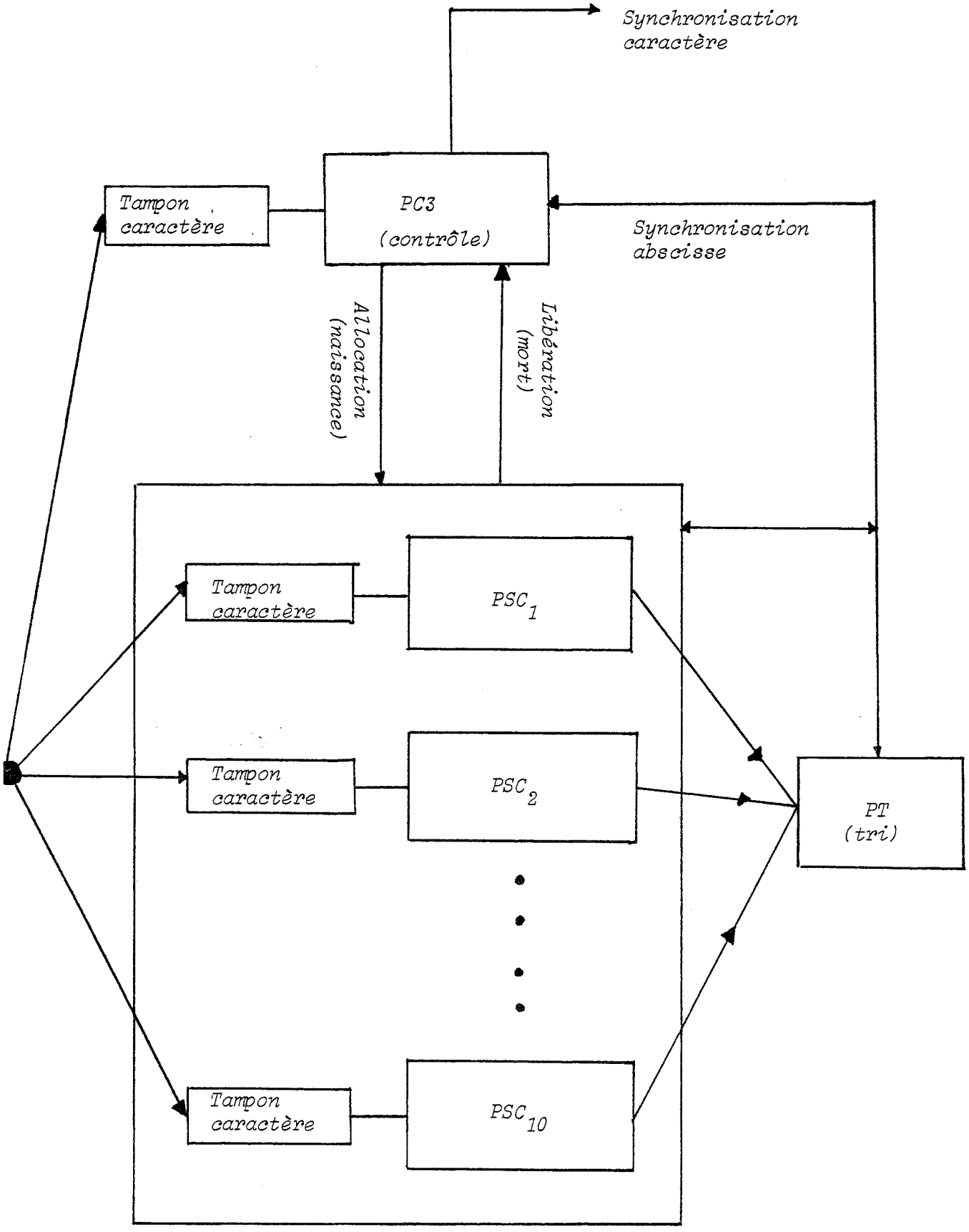


Figure 2-4 : organisation matérielle de l'Etape 3 (1ère étude)

Nous pouvons faire de nombreuses remarques sur cette architecture :

A) L'architecture est centralisée autour de PC3 et requiert de fonctionner tout le temps en configuration maximale (Nombre N de processeurs égal à 10). Or d'une part la plupart des caractères pourraient être générés avec N=6, voire N=4, ce qui traduit en moyenne un gaspillage important ; d'autre part la fiabilité résultante est faible, toute panne de processeur paralysant l'ensemble.

B) Les ressources en processeur sont mal utilisées pour d'autres raisons encore : quand PC3 travaille, les PSC_i sont en attente et réciproquement. D'autre part l'allocation des parois aux points de naissance se fait en série, ce qui amène une perte de temps importante surtout pour des caractères de petite taille.

C) Enfin, entre deux points de naissance, l'ordre des résultats reste constant, et le tri se résume à une indirection à travers une table qui ne doit être mise à jour qu'à chaque naissance. Il vaudrait mieux tenir compte de cette cohérence des calculs pour éviter d'avoir à réorganiser 10 valeurs en 10 μs.

B - Intérêt et difficulté d'un repliement.

Nous avons d'abord tenté de supprimer le processeur de tri PT en reportant la fonction au niveau de PC3, qui est à même de gérer la table d'indirection mentionnée ci-dessus. Malheureusement, cette opération se solde à la fois par une centralisation accrue et une baisse significative de la vitesse.

Une réflexion se basant sur la prise en compte de la remarque A nous a permis, en posant différemment le problème, de le résoudre : que se passe-t-il si l'on ne dispose que de n processeurs de suivi de contour ($1 \leq n \leq 10$)? Pour des caractères complexes, il faut alors allouer plus d'une paroi à un ou plusieurs processeurs. Nous avons appelé cette procédure **repliement** de l'ensemble des parois sur l'ensemble des processeurs ; le mécanisme d'allocation paroi→processeur n'est plus forcément injectif.

Le concept même du repliement est porteur de nombreux avantages :

- Possibilité de mettre au point le générateur avec n<10 (nous avons utilisé 1,

puis 2 processeurs de suivi de contour pour certifier son fonctionnement ; deux processeurs supplémentaires ont été rajoutés pour étudier ses performances).

- Repliement automatique en cas de disfonctionnement d'un PSC.
- Création d'une modularité supplémentaire permettant d'ajuster les caractéristiques du générateur de caractères à des marchés très différents : photocomposition, bureautique.

En revanche, nous avons eu à résoudre les difficultés suivantes pour passer de l'idée à sa réalisation :

- Transparence lorsque n est supérieur au nombre M des parois actives (ce qui est vrai en particulier si n=10).
- Lorsque $n < M$, création d'un mécanisme simple d'allocation des parois aux processeurs, équilibrant au mieux la charge de chacun d'eux.
- Comment récupérer, à chaque abscisse, les ordonnées calculées par les processeurs (chacun peut en émettre plusieurs), et les trier ?

L'architecture décrite plus haut s'est avérée, pour diverses raisons, absolument incompatible avec la notion même de repliement ; politique d'allocation et de désallocation délicates, synchronisations plus complexes, contraintes de temps non respectées.

C - Architecture retenue.

I Allocation parois processeurs

C'est en introduisant des informations supplémentaires dans le code, c'est à dire en modifiant les hypothèses mêmes de notre travail, que nous avons dépassé ces difficultés. Ainsi, pour résoudre le problème de l'allocation, nous avons associé à chaque paroi un **numéro de paroi** dont la seule donnée permet, aux abscisses de naissance, l'allocation automatique, décentralisée et optimale des parois aux processeurs, quel que soit l'environnement.

Allocation décentralisée et automatique :

Chaque PSC_i examine la structure d'arbre aux abscisses de naissance ;

l'application d'une fonction d'allocation aux numéros des parois activées indique celles qui sont à prendre en charge.

Allocation optimale :

Le calcul des numéros de parois, effectué hors ligne par la station d'édition, doit être tel que la charge des processeurs soit à peu près équilibrée ; l'affectation à chaque paroi de son numéro se fait par une fonction de désignation (fig 2-5). Le processus d'allocation devient alors trivial et PC3 est déchargé de sa fonction essentielle ; il ne lui reste guère plus qu'à gérer les diverses synchronisations et à mettre à jour la table d'indirection pour le tri, à chaque abscisse de naissance.

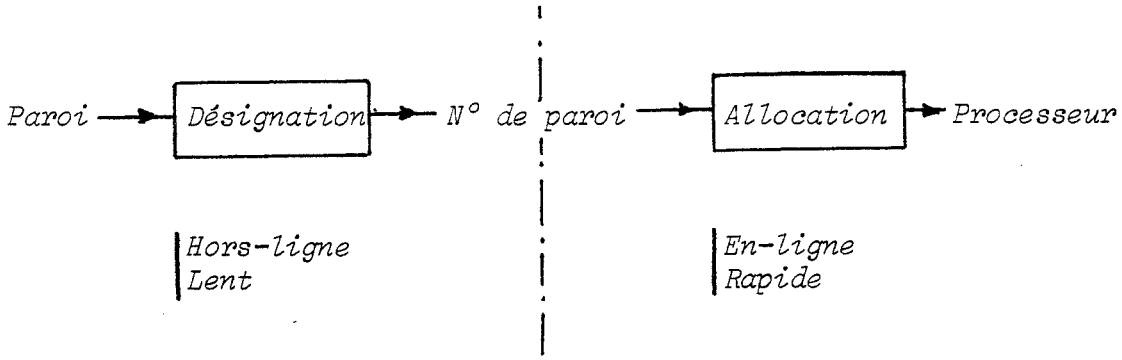


Figure 2-5 : Allocation optimale automatique.

II Tri des parois.

La simplification du problème du tri provient de la remarque suivante : dans la mesure où deux parois ne se croisent pas (et nous supposerons que le contour n'a pas de points doubles), les ordres partiels définis sur l'ensemble des parois actives à une abscisse donnée peuvent être immergés dans un ordre total cohérent. On fait alors correspondre dans le code à chaque paroi un numéro d'ordre qui permet lors du décodage son classement par une carte spéciale réalisée en logique cablée, la carte de sortie CS. A chaque abscisse, l'information émise par un PSC qui suit une paroi comprend d'une part l'ordonnée courante, d'autre part le numéro d'ordre, lequel sert d'adresse dans une RAM rapide où est stockée l'ordonnée. Au passage à l'abscisse suivante, les informations valides dans la RAM sont relues en séquence, puis transmises vers une file d'attente avant d'être exploitées par l'organe de visualisation.

Il faut signaler, pour terminer, que l'ensemble des fonctions de synchronisation ont été déportées au niveau du matériel, pour accélérer l'ensemble du processus.

III Calcul de l'information supplémentaire.

Ainsi, l'architecture de la 3e étape a pu être simplifiée et décentralisée, au prix d'une redondance somme toute minime du code (moins de 10%) ; il n'y a plus de processeur de contrôle, ni de tri, mais des numéros de parois et des numéros d'ordre qu'il s'agit de déterminer pour les inclure dans le code. Les deux problèmes ne sont pas équivalents, ni de même degré de difficulté : un mauvais choix sur l'ensemble des numéros de parois impliquera au pire une mauvaise allocation, donc une dégradation des performances globales ; par contre une erreur sur un numéro d'ordre rend incompréhensible le résultat du décodage.

Dans des conditions normales de fonctionnement, le nombre et la disposition des parois restent constants ; il est donc naturel de calculer les numéros d'ordre et les numéros de parois à priori, hors ligne, par la station d'édition, au cours de la phase de codage. Or nous savons que la structure d'un caractère est modifiée lorqu'on lui applique certaines opérations spéciales, telles que les inclinaisons ou les rotations. Le recalcul de la structure d'arbre doit s'accompagner d'une rédétermination en ligne des caractéristiques des parois. Comme ces applications spéciales tolèrent que la vitesse soit fortement diminuée, il suffit pour les numéros de parois, de donner une solution possible menant à une allocation suboptimale ; pour les numéros d'ordre, il s'agit de minimiser le nombre de tests à effectuer pour classer les parois ; cela ne peut se faire qu'en travaillant sur la structure globale du caractère.

D - Autres architectures possibles.

Chaque PSC_i a été réalisé autour d'un microprocesseur 16 bits, capable d'effectuer un calcul élémentaire en $10\mu s$: il y a ainsi adéquation de la capacité de traitement au schéma logique parallèle de génération.

Or les algorithmes étudiés ne dépendent en rien de la technologie employée. Une seconde architecture, respectant la contrainte de vitesse, pourrait être développée autour de deux processeurs de suivi de contour construits avec des microprocesseurs bipolaires en tranche ; l'allocation des parois serait alors élémentaire, un microprocesseur calculant les parois directes, l'autre les parois inverses. De même, pour toute application où la vitesse n'est pas un caractère dirimant, et où la taille des caractères est faible (imprimantes de traitement de texte), une archictecture simplifiée utilisant un microprocesseur standard permettrait de regrouper sur une seule carte l'ensemble des fonctions de la 3^{e} étape.

4.- CONCLUSION.

Les deux chapitres qui suivent s'attaquent à la détermination des numéros de parois (chapitre III) et des numéros d'ordre (chapitre IV). C'est là l'essentiel du travail théorique de la thèse ; nous avons pourtant tenu à rajouter une partie pour décrire l'arganisation du logiciel d'un PSC_i en cas de repliement (chapitre V), et une dernière pour aider à l'interprétation des mesures effectuées sur le prototype du générateur de caractères (chapitre VI). Chaque fois que cela à été possible, nous avons replacé les problèmes étudiés dans leur cadre le plus général, ce qui risque peut-être de dérouter le lecteur lors d'une première prise de contact.

Chapitre 3

CHAPITRE 3

DETERMINATION DES NUMEROS DE PAROIS

1.- UN PROBLEME PARTICULIER D'ALLOCATION DE RESSOURCES.

A.- Présentation.
B.- Intérêt.
C.- Allocation et désignation.
D.- Détermination d'une solution de base.
E.- La fonction-coût.
F.- Recherche d'un optimum heuristique par permutations.
G.- Recherche d'un optimum par une méthode déterministe.
H.- Conclusion.

2.- APPLICATION AU CALCUL DES NUMEROS DE PAROIS.

A.- Modélisation.
B.- Application au g minuscule Baskerville.

1.- UN PROBLEME PARTICULIER D'ALLOCATION DE RESSOURCES.

A.- Présentation

On considère un travail $\mathcal{T}$ constitué d'un ensemble de tâches indépendantes, similaires et répétitives p_i, $i \in [1,n]$, que l'on appelera **processus** . Chaque processus est donc formé d'une suite finie de calculs élémentaires

$$y_{i,k+1} = f(y_{i,k}) \quad , \forall i \in [1,n] \quad , \forall k \in [d_i, d_i + l_i]$$

L'indice k prend ses valeurs dans $[1,K]$; il pourra être identifié au temps pour plus de compréhension. Pour chaque processus p_i, d_i est l'indice du début de son éxécution, et l_i sa durée. Le calcul de $y_{i,k+1}$ ne pouvant commencer que si le calcul de $y_{i,k}$ est terminé, $\forall j \in [1,n]$, les calculs seront dits synchrones au niveau des indices.

L'execution d'un processus p_i se fera sur un **processeur** P_j , $j \in [1,m]$, dans les conditions suivantes:

* on peut affecter plusieurs processus à un même processeur,
* l'allocation est statique : une fois alloué à un processeur, un processus s'éxécute entièrement sur celui-ci,
* le temps de calcul nécessaire à un processeur P_j pour un indice k dépend linéairement du nombre de processus qui lui sont alloués à cet indice.

L'objectif est de déterminer une **méthode optimale d'allocation** qui minimise le temps total d'éxécution, quels que soient les nombres de processeurs m, de processus n, et le plan de charge $\{d_i, l_i\}$ $i \in [1,n]$.

B.- Intérêt.

Pour illustrer les concepts que nous venons de présenter, nous allons donner l'exemple de deux systèmes complexes qu'une telle approche permet de modéliser:

* Dans une économie planifiée, un grand groupe industriel peut fabriquer le même produit périssable dans m usines U_1, U_2,...U_m. Le service statistique indique qu'il existe un potentiel de n clients C_1, C_2,...C_n ; un client C_i

absorbe une unité de fabrication par mois pendant une certaine période d_i, d_i+l_i de l'année. Comme le produit est périssable, il ne peut y avoir de stockage ni à l'usine, ni chez le client; on considère par contre que la capacité de production d'une usine est grande : chaque usine peut , chaque mois, produire autant d'unités de fabrication qu'il est nécessaire.
La direction de la planification doit indiquer à chaque client C_i l'usine U_j où il devra s'approvisionner, tout en s'arrangeant pour que, chaque mois, la charge de travail de toutes les usines soit à peu près identique. Elle cherche de plus à prévoir une méthode générale pour les années à venir, qui ne dépendra pas du nombre m d'usines que le groupe pourra employer à la fabrication de ce produit.

* Le second exemple décrit l'application du modèle au générateur de caractères: un caractère formé de n parois P_1, P_2,...P_n sera décodé sur une machine possédant m processeurs de suivi de contour PSC_1 , PSC_2 ,...PSC_m. En remplaçant "indice k" par "abscisse x_k", on connaît pour chaque paroi son abscisse de naissance x_i et sa longueur l_i ; la contrainte de synchronisation au niveau des abscisses est nécessaire pour la génération d'un code par plages.
En première hypothèse, nous considérerons que le calcul élémentaire $f(x_k, y_{i,k})$ est le même pour toute paroi i; nous savons qu'il sera nécessaire d'affiner le modèle, puisque les temps d'évaluation d'un point d'une droite ou d'un cercle sont loin d'être équivalents.
Dans ces conditions, comment peut on allouer les parois aux processeurs, en équilibrant leur charge respective à chaque abscisse, afin de minimiser le temps global de génération? Est-il possible de prévoir une méthode d'allocation qui produise un résultat optimal quel que soit le nombre m de processeurs de suivi de contour?

C.- Allocation et désignation.

Pour chaque processus p_i, nous introduisons une fonction booléenne δ_i de $[1,K]$ dans $\{0,1\}$, qui sera son **indicateur d'activité:**

$\delta_i(k)=1$ si $d_i \leq k \leq d_i+l_i$

$\delta_i(k)=0$ sinon

Nous appelerons **complexité locale** à l'indice k:

$\delta(k)= \sum_i \delta_i(k)$ nombre des processus actifs à l'indice k

et **complexité globale** de $\mathcal{T}$: $M=\max_k \delta(k)$.

Supposons d'abord $\delta(k)=M \quad ,\forall k \in [1,K]$. L'ensemble des n processus admet alors une partition en M chaînes couvrant $[1,K]$, et la stratégie optimale d'allocation répartit évidemment les chaînes de façon symétrique sur les m processeurs disponibles. Autrement dit, l'algorithme optimum d'allocation A_{opt} se décompose en :

-une fonction de désignation $\psi: i \rightarrow N_i$ déterminant la chaîne N_i à laquelle rattacher le processus i;

- une fonction d'allocation réduite $\varphi: N_i \rightarrow j$ déterminant le processeur auquel allouer la chaîne N_i.

Dans le cas général, la complexité locale est variable sur $[1,K]$, mais nous chercherons encore à factoriser A_{opt} sous la forme :

$A_{opt}= \varphi \circ \psi$ avec;

φ : fonction d'allocation indépendante des $\{d_i,l_i\}$

ψ : fonction de désignation indépendante de m.

Bien que cette décomposition ne garantisse pas un optimum en un sens mathématique précis, elle a l'avantage de bien séparer dans A_{opt} ce qui, dans le générateur de caractères, doit être effectué à chaque remise en route (φ), de calculs qui peuvent être exécutés entièrement a priori lors du codage de chaque caractère (ψ). Cette stratégie est au moins optimale par sa simplicité et sa généralité.

ψ sert, comme précédemment, à former des chaînes, discontinues en général, à partir de processus totalement disjoints :

$N_i = N_j \Rightarrow \forall k \in [1,K] \ , \ \delta_i(k).\delta_j(k) = 0 \quad (1)$

Le nombre des numéros de chaîne est donc au moins égal à M, la complexité globale de $\mathcal{T}$. Par ailleurs, le nombre de chaînes peut ne pas dépasser M, atteint en particulier si $\delta(k) = M$ sur tout $[1,K]$. Ce nombre doit donc être égal à M, puisque le coût augmente nécessairement avec le nombre de chaînes.

Les N_i prenant donc leur valeur dans $[1,M]$, l'allocation d'un processus à un processeur sera déterminée par la **fonction d'allocation** $\varphi: [1,M] \rightarrow [1,m]$, telle que

$\varphi(N_j)$ soit l'indice j du processeur P_j sur lequel sera calculé le processus p_i.
Lorsque $M \leqslant m$, toute application injective de $[\![1,M]\!] \rightarrow [\![1,m]\!]$ convient; en particulier l'allocation triviale définie par $\varphi(N_j) = j$.
Dans le cas général $M > m$, une fonction d'allocation très simple peut être exhibée, calquée sur la précédente : $\varphi(N_i) \equiv j \pmod m$. Bien que toute surjection de $[\![1,M]\!]$ sur $[\![1,m]\!]$ puisse représenter une fonction d'allocation, cette **"allocation modulo m"** est intéressante dans la mesure où elle minimise la quantité d'information nécessaire à un processeur pour déterminer l'ensemble des processus qui lui sont alloués : il lui suffit de connaître son numéro de processeur j et le nombre m des processeurs actifs ($1 \leqslant j \leqslant m$).

Une fois fixée la procédure d'allocation, il reste à définir la **fonction de désignation** $\psi : [\![1,n]\!] \rightarrow [\![1,M]\!]$

$$p_i \mapsto \psi(i) = N_i$$

qui minimise le temps total d'exécution de $\mathcal{T}$. Il s'agit bien sûr d'un problème de nature combinatoire, et si l'on sait dénombrer l'ensemble des surjections possibles, - il y en a $A_n^M = (n!)/(n - M)!$, il n'est pas envisageable d'exhiber directement une solution optimale. Déjà l'ensemble des fonctions ψ qui vérifient la relation de disjonction (1) est très petit par rapport à l'ensemble de toutes les numérotations; malheureusement l'utilisation et la formalisation de ce critère sont difficiles, puisque la disposition interne des processus est très différente d'un travail $\mathcal{T}$ à un autre $\mathcal{T}'$, et l'on désire un procédé simple et universel.

Nous déterminerons donc une solution de base, vérifiant la relation de disjonction; à partir de là, en effectuant des opérations élémentaires sur cette solution, nous nous rapprocherons d'un optimum ψ^* heuristique; nous allons donc devoir définir une fonction-coût sur l'ensemble des ψ , dont le minimum correspondra au minimum du temps d'exécution de $\mathcal{T}$. Avant d'appliquer le modèle ainsi défini au générateur de caractères, nous donnerons une seconde méthode de résolution, déterministe, de ce problème.

D.- Détermination d'une solution de base

Nous allons nous appuyer sur un exemple dont les données figurent dans le tableau ci-contre :

i	d_i	l_i
1	10	200
3	10	40
5	20	160
7	0	40
9	30	180
11	150	160
13	40	120

On identifie les indices des évènements, c.à.d :

- **indices de naissance** ou indices de début de processus,
- **indices de mort** ou indices de fin de processus,

ce qui permet de définir des intervalles d'indice pendant lesquels l'état du système n'est pas changé; on ramène ainsi l'étude de l'allocation à celle des niveaux d'activité des processus; l'ensemble des indices passe de $[1,K]$ à $[1,K']$ avec $K' \ll K$ en général.

ABSCISSES D'EVENEMENTS (BORNES DES INTERVALLES)

0 10 20 30 40 120 150 160 180 200

LONGUEUR DES INTERVALLES

10 10 10 10 80 30 10 20 20

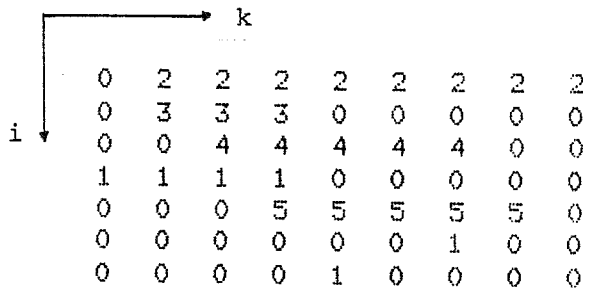
A partir du tableau d'entrée, on fabrique la matrice d'activité $\delta_{i,k}$ telle que $\forall i \in [1,n]$, $\forall k \in [1,K']$ $\delta_{i,k} = \delta_i(k)$.

0	1	1	1	1	1	1	1	1
0	1	1	1	0	0	0	0	0
0	0	1	1	1	1	1	0	0
1	1	1	1	0	0	0	0	0
0	0	0	1	1	1	1	1	0
0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0

Le nombre M nécessaire à l'identification des processus est $\max_k \sum_i \delta_{i,k} = 5$.
L'algorithme de détermination de la solution de base est le suivant :

```
pour k de 1 à K';
    début pour i de 1 à n;
        si mort(i) alors libérer la chaîne Nj;
        finpour;
        pour i de 1 à n;
        si naissance(i) alors Ni = premier n° de chaîne libre;
        finpour;
    fin;
finpour;
```

Cette méthode est très naturelle : on dispose au départ d'une liste triée des numéros libres. A chaque indice d'évènement, on commence par libérer les numéros des processus qui y meurent, puis on affecte à ceux qui naissent les numéros libres, dans l'ordre. L'application de cet algorithme produit la solution de base $B_{i,k}$ suivante :



$B_{i,k} = 0$ si $\delta_{i,k} = 0$
$B_{i,k} = N_i$ si $\delta_{i,k} = 1$

E.- La fonction-Coût.

Pour la détermination d'une solution de base, la connaissance du nombre des processeurs actifs est inutile; posons le maintenant égal à m. Les deux

cas m = 1 et m $\geqslant$ M sont triviaux, puisque pour l'un, tous les processus sont exécutés sur le même processus, et pour l'autre l'unicité des N_i dans chaque intervalle entraîne l'exécution d'au plus un processus par processeur. Dans le cas général où m $<$ M, nous allons introduire les notions suivantes:

* **charge locale** d'un processeur P_j à l'indice k: $C_j(k)$ est le nombre de processus alloués à P_j à l'indice k.

* **charge locale d'équilibre :** $C(k) = \delta(k)/m$.

Des hypothèses que nous avons posées en A découle le fait qu'une condition nécessaire de minimisation du temps d'exécution global de $\mathcal{P}$ est la répartition équilibrée des charges locales à chaque indice autour de la charge locale d'équilibre, ce qui s'écrit ainsi :

$\forall j , \forall k , \; E(C(k)) - 1 \leqslant C_j(k) \leqslant E(C(k))$.

où E(x) représente la partie entière par excès de $x \in R$.

On peut en déduire une **surcharge globale** sur l'intervalle k, par

$S(k) = 0$ si $\forall j \in [1,m] , \; E(C(k)) - 1 \leqslant C_j(k) \leqslant E(C(k))$

$S(k) = \underset{j}{\text{Max}}\, C_j(k) - E(C(k))$ sinon.

Dans ces conditions, si t est le temps d'évaluation d'un calcul élémentaire, et l(k) la longueur de l'intervalle d'indice $k \in [1,K']$, alors le temps d'exécution de l'intervalle k sera:

$t_k = t.l(k).(E(C(k)) + S(k))$, le temps minimum étant :

$t^*(k) = t.l(k).E(C(k))$.

La surcharge ainsi déterminée est une bonne fonction-coût, puisqu'elle s'annule lorsque l'allocation est optimale. Le coût d'une allocation sera donc :

$$C_m = \sum_k S(k).l(k) ,$$

l'indice m étant là pour rappeler le nombre de processeur actifs. Le critère de coût ne devant pas dépendre de m, nous prendrons en fait :

$$C = \sum_m \alpha(M,m).C_m ,$$

les coefficients $\alpha(M,m)$ étant des nombres positifs reflétant la nature du problème.

F.- Recherche d' un optimum heuristique par permutations.

Si l'on désire garder intacte la structure des chaînes, les seules opérations élémentaires qui permettent de passer d'une numérotation valide - respectant la règle de disjonction - , à une autre numérotation valide sont les permutations de $[1,M]$, dont le nombre est M! . Rappelons que le cardinal de l'ensemble des surjections est A_n^M ; le rapport du nombre de surjections au nombre de permutations est $A_n^M / M! = C_n^M \gg 1$ si $n \gg M$.

L'ensemble des permutations (ou substitutions) de $[1,M]$ s'appelle le groupe symétrique d'ordre M, $\mathcal{S}_M$. Soit $\sigma \in \mathcal{S}_M$. On remplace dans la solution de base N_i par $\sigma(N_i)$; on détermine le coût de cette nouvelle numérotation, et s'il est inférieur à celui de la solution de base, on prend comme nouvel optimum l'allocation définie par $\sigma(N_i)$. En parcourant l'ensemble $\mathcal{S}_M$, on exhibera une permutation donnant un coût minimal (elle n'est pas unique en général); la numérotation ainsi obtenue sera considérée comme optimale. Si, en parcourant $\mathcal{S}_M$, on trouve une permutation amenant un coût nul, on s'arrêtera à ce moment.

Lorsque M devient trop grand, il est long d'examiner $\mathcal{S}_M$ en entier. Une simulation a montré qu'en restreignant $\mathcal{S}_M$ à l'ensemble des transpositions $\tau_{i,j}$ ($i,j \in [1,M]$, $i \neq j$) - càd des permutations telle que

$\tau_{i,j}(i) = j$; $\tau_{i,j}(j) = i$; $\forall k \notin \{i,j\}$, $\tau_{i,j}(k) = k$

on arrivait à réduire considérablement le temps de calcul pour arriver à une solution quasi-optimale. Sur le tableau ci-dessous, on trouvera , en fonction de m, les coûts optimaux correspondant soit à une permutation générale σ^*, soit à une transposition τ^*, pour l'exemple précédent. Les coefficients $\alpha(M,m)$ choisis conduisent à une fonction-coût

$C = C_2 + C_3 + C_4$.

m	Solution de base	Meilleure transposition	Meilleure permutation
C_2	120	20	0
C_3	0	0	0
C_4	120	0	0
C	240	20	0

$\tau^* = (4\ 2\ 3\ 1\ 5) = \tau_{1,4}$

$\sigma^* = (3\ 4\ 2\ 5\ 1)$.

Restriction aux transpositions et itération.

Puisque l'application d'une seule transposition à la solution de base permet de réduire fortement le coût et conduit à une solution sub-optimale, on peut se demander ce qui se passe lorsqu'on itère ce raisonnement en appliquant une nouvelle transposition optimale à la solution ainsi déterminée : la récurrence ainsi dégagée converge-t-elle vers l'optimum σ^* ?

La démonstration de cette propriété simplifierait grandement la détermination de l'optimum : lorsque M vaut 5, il y a M! = 120 permutations, et M(M-1)/2 soit 10 transpositions; pour M = 6, il y a 720 permutations et 15 transpositions seulement. Essayons maintenant de formaliser le problème; considérons une application

$$f : \mathcal{S}_M \to \mathbf{N}$$
$$\sigma \mapsto f(\sigma)$$

$f(\sigma)$ est le coût de la permutation σ ; $f(Id) = f_0$ est le coût de la solution de base. Par énumération sur $\mathcal{S}_M$, on détermine une permutation σ^*, en général non unique, telle que

$$f^* = f(\sigma^*) = \min_{\mathcal{S}_M} f(\sigma)$$

σ^* peut être canoniquement décomposée en un produit de transpositions :

$$\sigma^* = \prod_{q \in [1,n]} \tau_q .$$

Mais rien ne nous permet d'affirmer que la suite $\sigma_1 = \tau_1$; $\sigma_2 = \tau_2 \circ \tau_1$; ... $\sigma^* = \tau_n \circ \cdots \circ \tau_2 \circ \tau_1$ définit une suite non croissante $f(\sigma_1), f(\sigma_2) \ldots\ldots f(\sigma^*)$; cela est même faux en général. Réciproquement, si l'on restreint $\mathcal{S}_M$ à l'ensemble des transpositions, peut-on définir une suite $\tilde{\tau}_1, \tilde{\tau}_2, \cdots \tilde{\tau}_n$ telle que $f(\tilde{\tau}_1), f(\tilde{\tau}_2 \circ \tilde{\tau}_1) \ldots\ldots f(\tilde{\tau}_n \circ \tilde{\tau}_{n-1} \ldots \circ \tilde{\tau}_1)$ soit une suite non décroissante ? Dans ces conditions est on assuré de la convergence de cette suite vers f^* ?

La forme complexe de la fonction-coût f ne nous permet pas de résoudre formellement ce problème; c'est pourquoi nous avons mené une simulation en APL sur deux exemples. Dans les deux cas M = 5, les coefficients de la fonction-coût $\alpha(5,m)$ valent (0,0,1,2,4,0,0....).

EXEMPLE 1

$f(\sigma^*) = 0$

ENTREE

1	10	100
2	10	30
3	20	40
4	30	40
5	40	50
6	50	80
7	50	80
8	90	100
9	20	30
10	90	100
11	60	90
12	50	80

SOLUTION_DE_BASE

1	1	1	1	1	1	1	1
2	2	0	0	0	0	0	0
0	3	3	0	0	0	0	0
0	0	2	0	0	0	0	0
0	0	0	2	0	0	0	0
0	0	0	0	2	2	0	0
0	0	0	0	3	3	0	0
0	0	0	0	0	0	0	2
0	4	0	0	0	0	0	0
0	0	0	0	0	0	0	3
0	0	0	0	0	5	5	0
0	0	0	0	4	4	0	0

COUT_DE_LA_SOLUTION_DE_BASE : 50

PASSAGE NUMERO : 1

MEILLEURE_TRANSPOSITION : 1 3

COUT_DE_CETTE_SOLUTION : 10

PASSAGE NUMERO : 2

MEILLEURE_TRANSPOSITION : 1 4

COUT_DE_CETTE_SOLUTION : 10

```
PASSAGE NUMERO : 3

MEILLEURE_TRANSPOSITION     : 2   4

COUT_DE_CETTE_SOLUTION : 10


PASSAGE NUMERO : 4

MEILLEURE_TRANSPOSITION     : 3   4

COUT_DE_CETTE_SOLUTION : 0

SOLUTION_OPTIMALE

1    3
2    2
3    1
4    2
5    2
6    2
7    1
8    2
9    4
10   1
11   5
12   4
```

L'optimum est atteint en
quatre itérations.

EXEMPLE 2

$f(\sigma^*) = 20$

L'optimum est atteint en deux itérations.

ENTREE

1	10	200
3	10	40
5	20	160
7	0	40
9	30	180
11	150	160
13	40	120

SOLUTION_DE_BASE

0	2	2	2	2	2	2	2	2
0	3	3	3	0	0	0	0	0
0	0	4	4	4	4	4	0	0
1	1	1	1	0	0	0	0	0
0	0	0	5	5	5	5	5	0
0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0

COUT_DE_LA_SOLUTION_DE_BASE : 640

PASSAGE NUMERO : 1

MEILLEURE_TRANSPOSITION : 1 3

COUT_DE_CETTE_SOLUTION : 100

SOLUTION_OPTIMALE

1	2
3	1
5	4
7	3
9	5
11	3
13	3

PASSAGE NUMERO : 2

MEILLEURE_TRANSPOSITION : 2 3

COUT_DE_CETTE_SOLUTION : 20

SOLUTION_OPTIMALE

1	2
3	1
5	4
7	3
9	5
11	3
13	3

G.-Recherche d'un optimum de manière déterministe.

L'analyse de la structure du coût nous indique qu'il faut essayer d'assurer, dans chaque intervalle k, la consécutivité des numéros de chaînes actifs : si les indices sautent de N à N + 2 , l'allocation modulo sera mauvaise pour m = 2 (surcharge du processeur N (2) par rapport au processeur N+1 (2)); de même, si les indices passent de N à N+3, l'allocation sera mauvaise pour m = 3 (surcharge du processeur N (3) par rapport aux processeurs N+1 (3) et N+2 (3)).

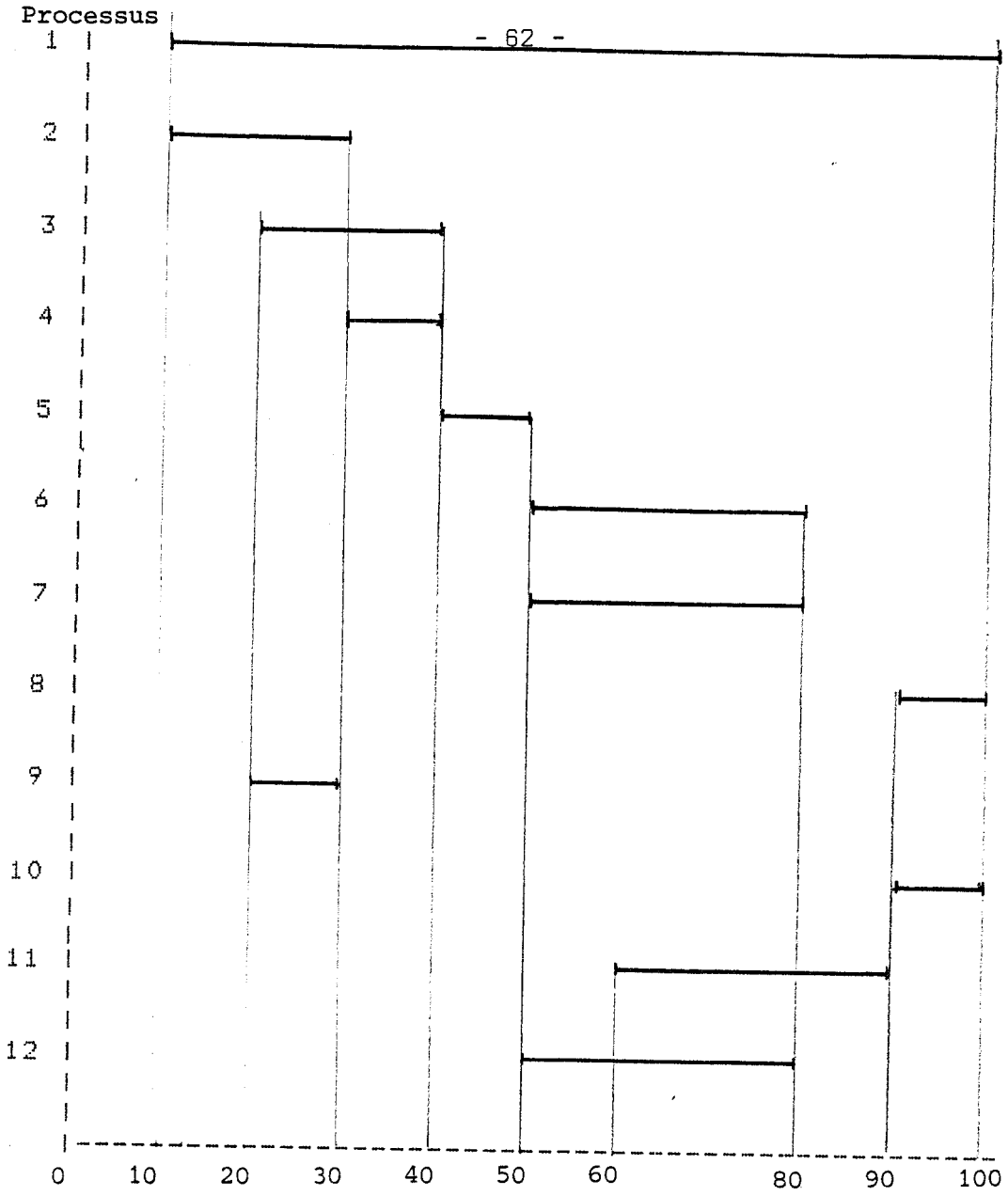
Pour minimiser le coût, il nous est possible d'agir dans deux directions :

- utiliser au mieux les numéros disponibles en compactant les chaînes, ce qui diminue le nombre des trous : il faut essayer d'obtenir les chaînes les plus longues.
- dans chaque intervalle, compacter les numéros de chaînes pour les rendre consécutifs.

Compaction des chaînes.

L'algorithme de détermination de la solution de base conduit à une organisation qui ne minimise pas forcément la longueur des trous; d'autre part, la compaction des numéros de chaînes se fait de manière non déterministe, en parcourant une table de transformations - permutations ou transpositions - et en choisissant alors la solution qui minimise la fonction-coût. La structure même des chaînes n'est pas réorganisée, il n'est pas question , en général, d'affirmer que la numérotation obtenue est optimale (sauf, bien sûr, si son coût est nul).

Pour dégager une solution a priori meilleure que la solution de base, où la construction des chaînes dépend de l'ordre dans lequel sont examinés les processus, il faut à chaque fois qu'il y a une indétermination sur un processus à affecter à une chaîne, choisir celui qui permettra la construction d'une sous-chaîne maximale. On est amené à résoudre un problème récurrent qui s'apparente à ceux que l'on traite par les méthodes de la programmation dynamique; nous allons en préciser les grands traits sur les données de l'exemple 2 du paragraphe précédent.



M = Max $\delta(k) = 5$; il faut donc construire 5 chaînes.

Le processus 1 bloque la chaîne 1 ; tant qu'il n'y a pas de choix, on affecte les numéros de chaînes dans l'ordre consécutif : le processus 2 est affecté à la chaîne 2, le procesus 3 à la chaîne 3, le processus 9 à la chaîne 4. Lorsque se présente le processus 4, on ne sait s'il faudra l'affecter à 2 ou à 4; on lui donne donc le numéro α (α = 2 ou 4) qu'il s'agira de déterminer pour obtenir la chaîne la plus longue possible. De même on affecte au processus 5 le numéro β , dont la valeur est à prendre dans (α,3) ou encore dans (2,3,4). Les processus 6, 7, et 12 sont absolument équivalents, donnons leur les numéros β , β', β'' à prendre dans (2,3,4). Le processus 11 doit être affecté à

la chaîne 5; les processus 8 et 10 à γ , γ' à prendre dans β , β', β''. Il est maintenant possible de revenir en arrière :

$\beta = \gamma$ et $\beta' = \gamma'$ (il y a indifférence)

. $\beta = \alpha = 2$; $\beta' = 3$; $\beta'' = 4$.

La solution obtenue est équivalente à la solution de base.

Poids d'un intervalle - Poids d'une chaîne.

Soit $\delta(k)$ la complexité locale sur un intervalle k de longueur l(k), c.à.d le nombre des chaînes actives sur k, et soit M la complexité globale du travail $\mathcal{T}$. Nous affectons à cet intervalle un coefficient P(M,k), son poids, qui doit posséder les propriétés suivantes :

* dépendre linéairement de l(k),
* refléter, pour chaque repliement possible, la probabilité d'une mauvaise allocation, pondérée par l'importance que l'on porte à ce repliement.

Nous poserons donc :

$$P(M,k) = l(k). \sum_{m=1}^{M} \alpha(M,m).R(M,m,\delta(k)) = l(k).P'(M,\delta(k))$$

où $\alpha(M,m)$ représente le coefficient de pondération associé à un repliement sur m processeurs ($m < M$ dans les cas qui nous intéressent), et $R(M,m,\delta(k))$ représente la fréquence statistique des mauvaises numérotations dans l'ensemble des numérotations possibles, dont le cardinal est C_M^m.

Lorsqu'il ne peut y avoir de mauvaises numérotations, le coefficient R est nul :

$R(M,1,i) = 0 \qquad \forall i \in [1,M]$

$R(M,m,i) = 0 \qquad \forall i \in [1,M] \qquad$ si $m \geqslant M$

$R(M,j,0) = 0 \qquad \forall j$

$R(M,j,1) = 0 \qquad \forall j$

$R(M,j,M) = 0 \qquad \forall j$

Donnons un seul détail de calcul des coeficients R pour M = 5 :

$\delta(k) = 2$:

m = 2 : Mauvais 1-3,1-5,3-5,2-4

R(5,2,2) = 4/10 = 0,4.

m = 3 : Mauvais 1-4,2-5

R(5,3,2) = 0,2

m = 4 : Mauvais 1-5

R(5,4,2) = 0,1

m = 1 et m = 5 donc R(5,m,2) = 0

L'évaluation de ces coefficients pour les valeurs 3 et 4 de $\delta(k)$ conduit au tableau de pondération suivant :

m

$\delta(k)$	1	2	3	4	5
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0,4	0,2	0,1	0
3	0	0,1	0,6	0,2	0
4	0	0,4	0	0,4	0
5	0	0	0	0	0

Les coefficients de pondération des repliements correspondent à ceux qui sont utilisés pour l'évaluation du coût; si l'on prend :

$\alpha(5,1) = \alpha(5,5) = 0$

$\alpha(5,2) = 1 \qquad \alpha(5,3) = 2 \qquad \alpha(5,4) = 4$

on en déduit :

P'(5,1) = P'(5,5) = 0

P'(5,2) = 1,2

P'(5,3) = 2,1

P'(5,4) = 2,0

et le poids des 9 intervalles de l'exemple étudié :

P(5,1) = 0 P(5,6) = 20

P(5,2) = 12 P(5,7) = 0

P(5,3) = 20 P(5,8) = 12

P(5,4) = 21 P(5,9) = 21

P(5,5) = 12

Nous pouvons maintenant définir le poids d'une chaîne i dont l'indicateur d'activité est D_j; c'est un vecteur W_i à M composantes, tel que :

$$W_i(j) = \sum_k D_i(k).P(M,k).\ 1_{j=\delta(k)}$$

Dans notre exemple, nous obtenons le tableau ci-dessous :

i

j	1	2	3	4	5
4	40	40	40	40	0
3	42	42	42	0	0
2	36	24	0	0	12

Détermination de l'optimum

Le coût sera minimal si, pour chaque j, on arrive à rendre consécutifs les numéros de chaînes de poids les plus lourds. Si l'on appelle σ la permutation que nous cherchons, on obtient :

$\sigma(1)$, $\sigma(2)$, $\sigma(3)$, $\sigma(4)$ consécutifs (j = 4)

$\sigma(1)$, $\sigma(2)$, $\sigma(3)$ consécutifs (j = 3)

$\sigma(1)$, $\sigma(2)$, $\sigma(4)$ consécutifs (j = 2)

Les quatre solutions évidentes de ce problème sont :

$\sigma_1 = (3,2,4,5,1)$

$\sigma_2 = (2,3,4,5,1)$

$\sigma_3 = (4,3,2,1,5)$

$\sigma_4 = (3,4,2,1,5)$

Les contraintes de consécutivité étant parfaitement respectées, nous sommes assurés de la nullité du coût de ces différentes numérotations; il est d'ailleurs intéressant de remarquer que σ_3 est la permutation optimale σ^* déterminée par la méthode heuristique.

H.-Conclusion

Nous venons de décrire deux méthodes très différentes pour la résolution du problème de la numérotation des chaînes. La première, à partir d'une solution de base fabriquée par application d'un algorithme naturel, procède par énumération des solutions que l'on peut en déduire; la seconde construit un chaînage plus étudié, et conduit à la découverte d'un ensemble de solutions optimales de manière directe. La profonde dissymétrie entre ces deux démarches nous amène à formuler les remarques suivantes :

La première méthode simple et directe, est particulièrement facile à programmer, et conduit à des programmes itératifs; le nombre de passage dans la boucle principale est M! si l'on désire une solution optimale, ou C_M^2 si l'on se satisfait d'une approximation sub-optimale. Pour des applications en temps réel où l'on se contente d'une solution sub-optimale, la solution de base, ou l'optimum par transpositions, peuvent parfaitement convenir.

La deuxième méthode a le mérite de la directivité et de l'exhaustivité; facile à exécuter à la main lorsque M est assez petit, elle inclut deux algorithmes récursifs délicats à implanter en machine :

* la construction des chaînes par compactage fait appel à la programmation dynamique;
* la résolution des relations de consécutivité se fait par création et parcours d'un arbre des solutions.

La profondeur des récursions est en général assez limitée : de l'ordre de K' pour le premier point, et de M - 2 pour le second. Nous n'avons pas programmé cette seconde méthode; cela serait aisé si nous disposions d'un langage de manipulations de listes du type LISP; nous montrerons pourtant dans l'exemple qui suit, sa grande simplicité.

2.-APPLICATION AU CALCUL DES NUMEROS DE PAROI.

A.-Modélisation.

Le modèle que nous venons de décrire peut s'appliquer de manière immédiate à l'allocation en temps réel des parois aux processeurs de suivi de contour (cf. 1.B). Une procédure de désignation affecte à chaque paroi un numéro qui permettra son allocation à un PSC_i ($i \in [1,m]$) par application d'une fonction d'allocation modulo m. Le problème doit être généralisé pour s'adapter à un ensemble de travaux $\mathcal{T}$, c.à.d à l'ensemble des caractères que l'on veut pouvoir engendrer : on choisira les numéros de paroi dans $[1,M]$, où M est le maximum des complexités globales des différents caractères (M est égal à 10 pour les polices romaines de corps de texte).

Lorsque m = 1 ou m = 10, l'allocation est alors optimale pour tout caractère. Si on suppose que m ne peut prendre que des valeurs paires (2,4,6,8) , nous pouvons décomposer le problème en deux parties disjointes : l'étude de l'allocation des parois directes, qui seront désignées par des numéros pairs et affectées sur un processeur de numéro pair, et celle des parois inverses, désignées par un numéro impair et exécutées par un processeur de numéro impair . Dans ces conditions, l'allocation est toujours optimale pour m = 2.

Les paramètres de pondération de la fonction coût sont choisis de manière à tenir compte de la philosophie commerciale du repliement : il est souhaitable que l'allocation soit d'autant meilleure que le nombre des PSC est plus grand. Une machine où m = 10 doit fonctionner de manière optimale; dans une version moins complète (m =8 ou m = 6), on se préoccupe encore beaucoup des performances de vitesse; dans une machine bas de gamme où m =4, destinée à travailler dans un environnement plus lent, l'optimalité de l'allocation n'est plus forcément une contrainte à respecter. C'est pourquoi les coefficients $\alpha(5,m)$ retenus sont ceux que nous avons exhibés plus haut.

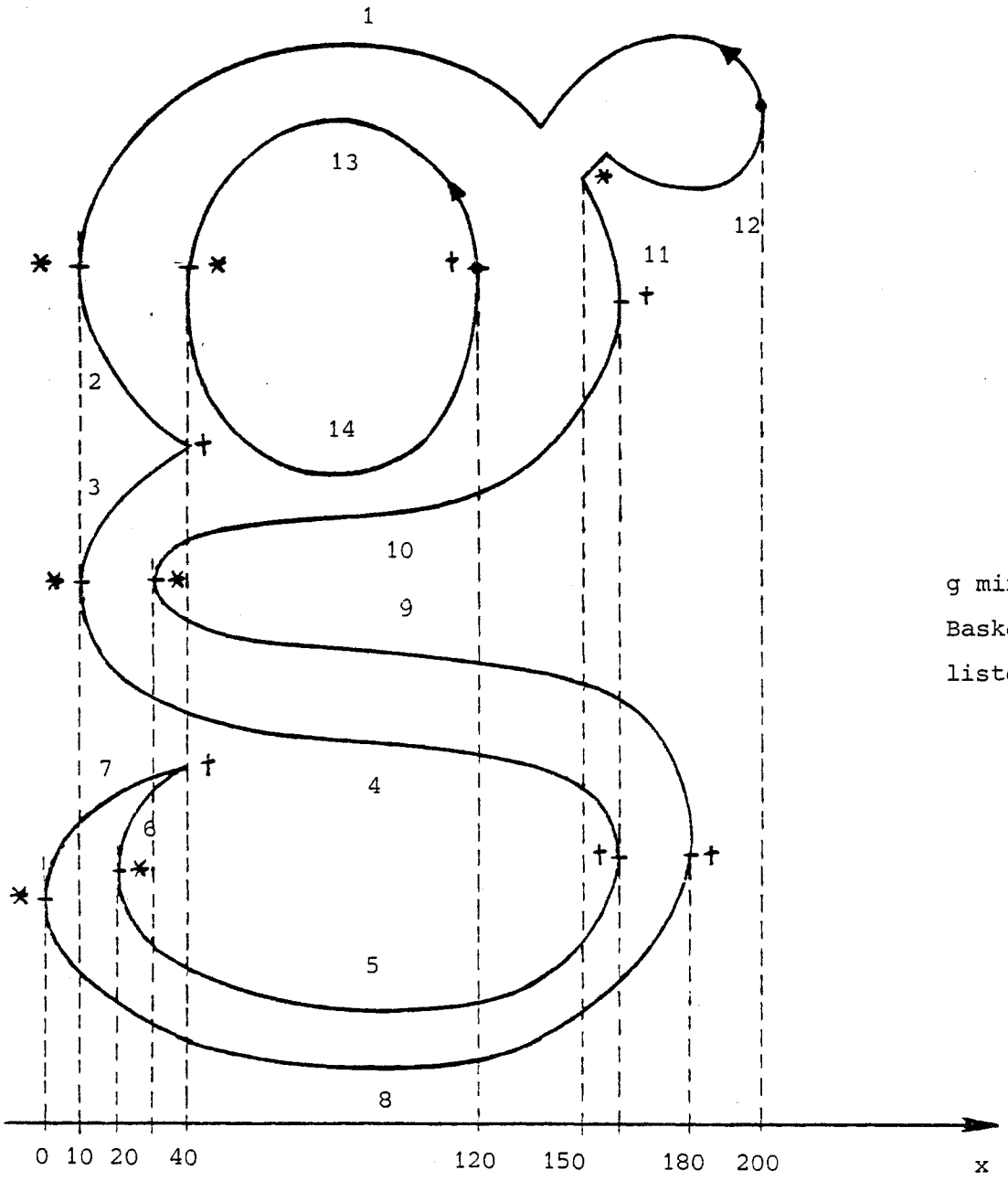
Lorsque la détermination des numéros de parois se fait hors-ligne sur la station d'édition, il est possible de calculer la fonction de désignation optimale, soit par parcours complet de l'ensemble des permutations, soit par activation de

l'algorithme déterministe. Par contre, lorsqu'elle est déterminée en temps réel par une étape qui recalcule la structure d'arbre, il faut s'arrêter dès que la solution de base est trouvée, ou à la limite exécuter une itération de l'algorithme par transpositions. Dans les deux cas, on est assuré d'obtenir une solution sub-optimale.

B.-Application au g minuscule Baskerville.

Pour terminer ce chapitre, nous donnons un exemple complet : l'application des deux modèles à la détermination des numéros de parois du g minuscule Baskerville. Rappelons que dans BLO 79 , nous avions procédé à une numérotation heuristique des parois de ce caractère, en nous basant sur les remarques suivantes :

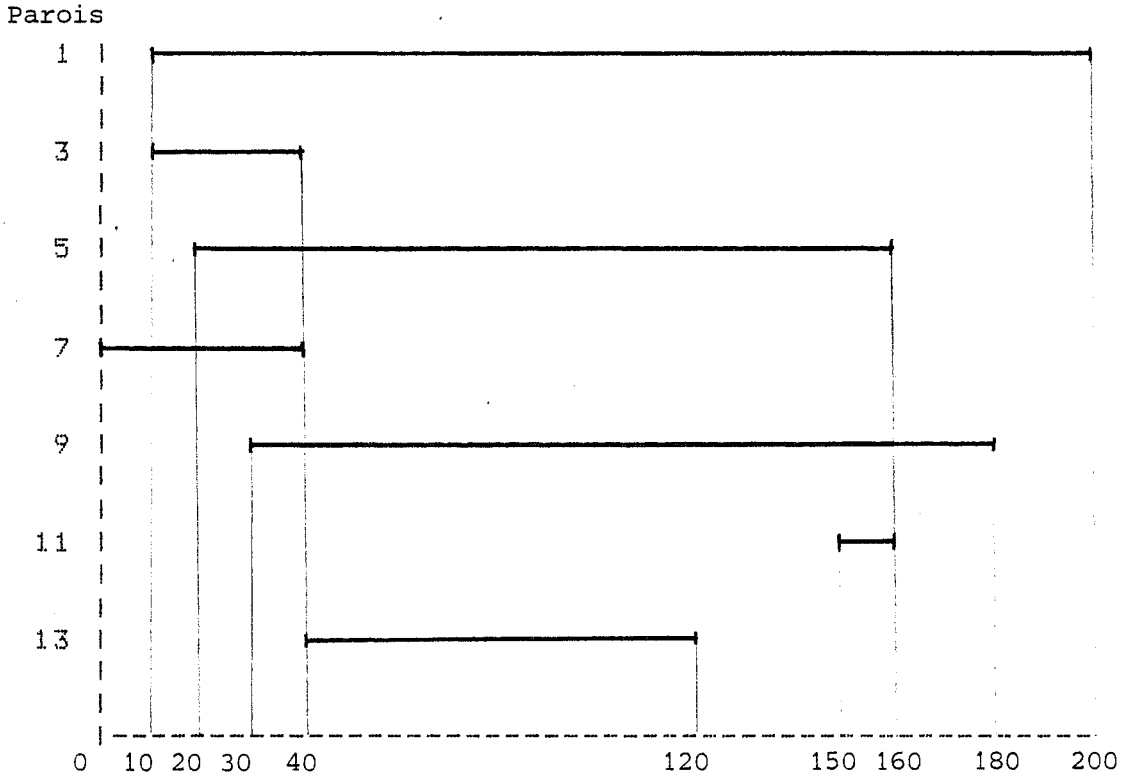
* donner des numéros consécutifs à des parois naissant au même point ; destinée à optimiser a priori le repliement sur deux processeurs, cette idée a été généralisée, puisqu'en fait nous donnons des numéros de parité opposée à de telles parois.
* réutiliser au plus vite les numéros des parois mortes; c'est le principe même de la fabrication des chaînes de la solution de base.
* éviter les trous dans la continuité des numéros correspondant à des parois actives, et cela pour les intervalles les plus longs.



g minuscule
Baskerville :
liste des parois.

Par	d_i	d_i+l_i
1	10	200
2	10	40
3	10	40
4	10	160
5	20	160
6	20	40
7	0	40
8	0	180
9	30	180
10	30	160
11	150	160
12	150	200
13	40	120
14	40	120

PAROIS INVERSES



ENTREE

1	10	200
3	10	40
5	20	160
7	0	40
9	30	180
11	150	160
13	40	120

ABSCISSES D'EVENEMENTS (BORNES DES INTERVALLES)

0 10 20 30 40 120 150 160 180 200

LONGUEUR DES INTERVALLES

10 10 10 10 80 30 10 20 20

MATRICE D'ACTIVITE

0	1	1	1	1	1	1	1	1
0	1	1	1	0	0	0	0	0
0	0	1	1	1	1	1	0	0
1	1	1	1	0	0	0	0	0
0	0	0	1	1	1	1	1	0
0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0

Méthode heuristique

SOLUTION_DE_BASE

0	2	2	2	2	2	2	2	2
0	3	3	3	0	0	0	0	0
0	0	4	4	4	4	4	0	0
1	1	1	1	0	0	0	0	0
0	0	0	5	5	5	5	5	0
0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0

COUT_DE_LA_SOLUTION_DE_BASE : 640

MEILLEURE_TRANSPOSITION : 1 3

COUT_DE_CETTE_SOLUTION : 100

Par	Num
1	2
3	1
5	4
7	3
9	5
11	3
13	3

MEILLEURE_PERMUTATION :4 3 5 2 1

COUT_DE_CETTE_SOLUTION : 20

Par	Num
1	3
3	5
5	2
7	4
9	1
11	4
13	4

Méthode déterministe

L'algorithme de construction des chaînes, en raison de la simplicité du problème, aboutit à la solution de base.

Tableau de pondération des chaînes :

j \ i	1	2	3	4	5
4	200	200	20	200	180
3	21	84	21	63	63
2	0	24	0	0	24

Contraintes de consécutivité :

$\sigma(1), \sigma(2), \sigma(4)$ j = 4

$\sigma(2), \sigma(4), \sigma(5)$ j = 3 (ne pourra être parfait)

$\sigma(2), \sigma(5)$ j = 2

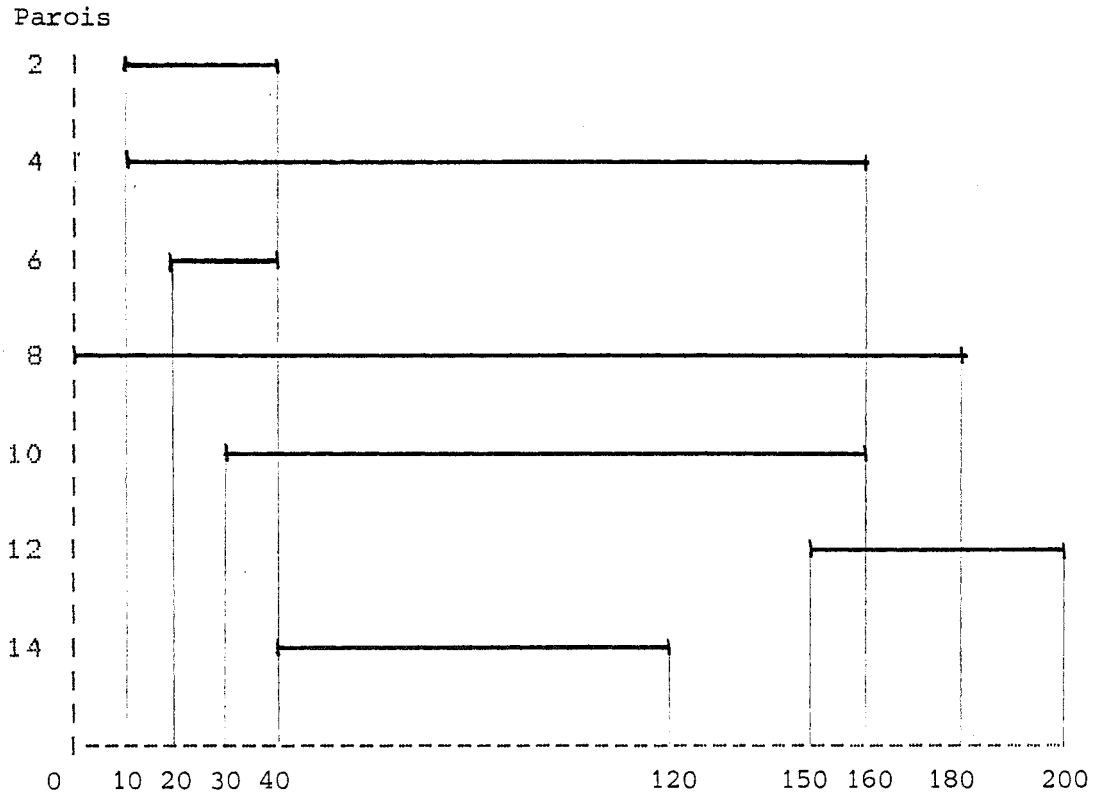
Solutions :

$\sigma(3), \sigma(1), \sigma(4), \sigma(2), \sigma(5)$ $\sigma_1 = (2\ 4\ 1\ 3\ 5)$

$\sigma(5), \sigma(2), \sigma(4), \sigma(1), \sigma(3)$ $\sigma_2 = (4\ 2\ 5\ 3\ 1)$

$$f(\sigma_1) = f(\sigma_2) = 20$$

PAROIS DIRECTES



ENTREE

2	10	40
4	10	160
6	20	40
8	0	180
10	30	160
12	150	200
14	40	120

ABSCISSES_D'EVENEMENTS (BORNES DES INTERVALLES)

0 10 20 30 40 120 150 160 180 200

LONGUEUR_DES_INTERVALLES

10 10 10 10 80 30 10 20 20

MATRICE_D'ACTIVITE

0	1	1	1	0	0	0	0	0
0	1	1	1	1	1	1	0	0
0	0	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	0
0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	1	1	1
0	0	0	0	1	0	0	0	0

Méthode heuristique

SOLUTION_DE_BASE

0	2	2	2	0	0	0	0	0
0	3	3	3	3	3	3	0	0
0	0	4	4	0	0	0	0	0
1	1	1	1	1	1	1	1	0
0	0	0	5	5	5	5	0	0
0	0	0	0	0	0	2	2	2
0	0	0	0	2	0	0	0	0

COUT_DE_LA_SOLUTION_DE_BASE : 630

MEILLEURE_TRANSPOSITION : 1 4

COUT_DE_CETTE_SOLUTION : 20

Par	Num
2	2
4	3
6	1
8	4
10	5
12	2
14	2

MEILLEURE_PERMUTATION : 3 2 4 1 5

COUT_DE_CETTE_SOLUTION : 0

Par	Num
2	2
4	4
6	1
8	3
10	5
12	2
14	2

Méthode déterministe

L'algorithme de construction des chaînes aboutit à la solution de base de la méthode heuristique.

Tableau de pondération des chaînes :

j \ i	1	2	3	4	5
4	200	200	200	180	20
3	84	21	84	0	63
2	24	24	0	0	0

Contraintes de consécutivité :

$\sigma(1), \sigma(2), \sigma(3)$ $\quad j = 4$

$\sigma(1), \sigma(3), \sigma(5)$ $\quad j = 3$

$\sigma(1), \sigma(2)$ $\quad j = 2$

Solutions :

$\sigma(4), \sigma(2), \sigma(1), \sigma(3), \sigma(5)$ $\quad \sigma_1 = (3\ 2\ 4\ 1\ 5)$

$\sigma(5), \sigma(3); \sigma(1), \sigma(2), \sigma(4)$ $\quad \sigma_2 = (3\ 4\ 2\ 5\ 1)$

$f(\sigma_1) = f(\sigma_2) = 0 \quad ; \quad \sigma_1 = \sigma^*$

Chapitre 4

CHAPITRE 4

DETERMINATION DES NUMEROS D'ORDRE DES PAROIS.

I.- POSITION DU PROBLEME.

1.- Point contre paroi.

2.- Local et global.

II.- DEFINITION ET RESULTATS GENERAUX.

1.- Représentation intrinsèque des caractères.

2.- Parois.

3.- L'ordre de balayage.

III.- ALGORITHMES.

1.- Ordre local - Lobes.

2.- Ordre global - Chameaux.

3.- Ordre global - Spirales.

4.- Forme réduite.

5.- Réintroduction des chameaux.

6.- Résumé de la méthodologie.

IV.- APPLICATION DU MODELE AU GENERATEUR DE CARACTERES.

1.- Première étude : g minuscule Baskerville.

2.- Deuxième étude : E majuscule Baskerville.

V.- ETUDE SOMMAIRE DES ROTATIONS.

1.- Représentation intrinsèque.

2.- Mort et naissance des extrema locaux.

I.- POSITION DU PROBLEME.

1.-Point contre paroi.

La découverte de la décomposition en parois nous a fourni un shéma logique bien adapté à la transformation d'un code du contour en un code par plages, et a induit naturellement l'architecture matérielle et logicielle de la 3° étape : parallélisme d'exécution, repliement... Le classement à chaque abscisse des ordonnées calculées a été simplifié par l'attribution d'un **numéro d'ordre** à chaque paroi; le tri est alors remplacé par une indirection réalisée en logique câblée sur la carte de sortie.

La détermination des numéros d'ordre nous semblait a priori extrêmement simple; il suffit de disposer d'une liste des parois triées suivant les abscisses de naissance croissantes et l'algorithme est alors le suivant :

```
pour toute abscisse de naissance entre abs-début et abs-fin;
    pour tout point de naissance;
        pour toute paroi active à cette abscisse;
            tester le point de naissance contre la paroi;
        finpour;
        insérer les deux parois activées dans la liste triée;
    finpour;
finpour;
```

Précisons ce que signifie "test d'un point contre une paroi active". La seule position des noeuds d'interpolation ne suffit pas à déterminer la position d'un point par rapport à une paroi; la figure 4-1 montre une paroi ABCD formée de la concaténation de trois interpolants canoniques. A l'abscisse x_N démarre en N une nouvelle paroi. La donnée des points C, D, et du centre C_{CD} est a priori insuffisante pour positionner N au dessus ou en dessous de l'arc de cercle CD (points N_1 et N_2); il faut alors calculer la puissance du point par rapport au cercle, ou bien décoder partiellement la courbe d'interpolation jusqu'à l'abscisse x_N pour comparer les positions respectives des points N et M (sur l'arc CD).

Lorsque la recherche des numéros d'ordre est conduite sur la station

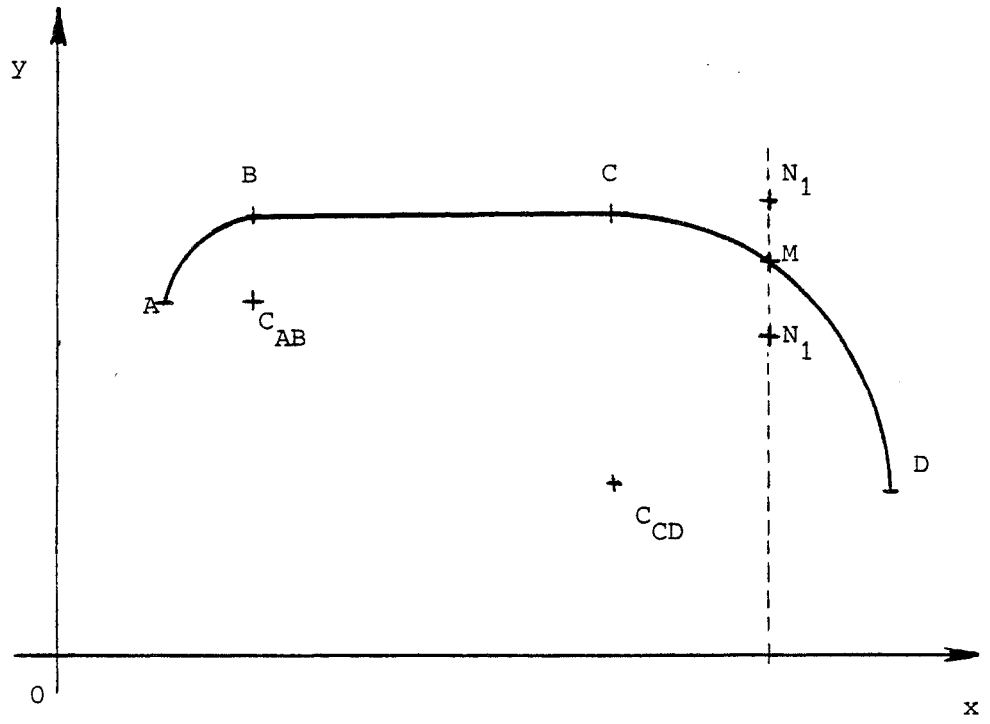


Figure 4-1

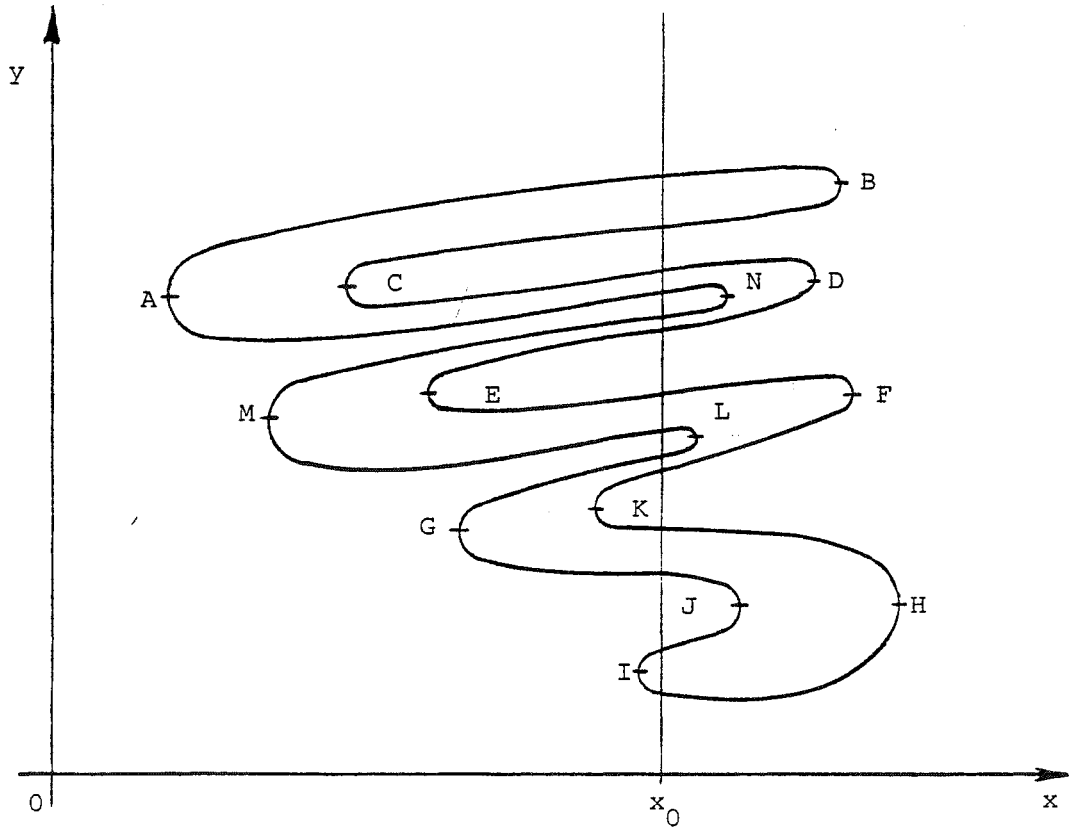


Figure 4-2

d'édition, on dispose de tout le temps nécessaire pour l'une ou l'autre des méthodes; par contre, si l'on veut les déterminer en temps réel, après une rotation du caractère pour fixer les idées, on se rend compte qu'elles sont aussi coûteuses l'une que l'autre : multiplications dans un cas, nombreuses itérations d'un algorithme incrémental de type Bresenham dans l'autre cas. Or le nombre des tests à effectuer est loin d'être négligeable, nous allons le montrer sur l'exemple de la figure 4-2.

Liste des parois : AB,CB,CD,ED,EF,GF,GH,IH,IJ,KJ,KL
ML,MN,AN (14 parois)

Liste triée des points de naissance : A,M,C,E,K,G,I (7 points)

Le tableau 4-3 résume le déroulement de l'algorithme et comptabilise à la fois le nombre de tests réellement effectués, et le nombre maximum dans le cas théorique le plus défavorable selon que l'on utilise un tri par bulles ou un tri dichotomique.

Si l'on suppose qu'il y a K abscisses de naissance distinctes, le nombre maximum de tests dans le cas le plus défavorable, où les K naissances précèdent la première mort, est $K(K-1)$ - tri par bulles - ou $K\text{Log}_2(K)$ - tri dichotomique -; même lorsque K est assez petit (4 à 6 pour des caractères de police romaine de corps de texte), l'ensemble de ces opérations semble difficile à mener en temps réel.

2.-Local et Global.

Pourtant, lorsque nous examinons la figure 4-2, nous nous demandons pourquoi tant de tests sont indispensables; cette impression provient de l'appréhension globale que nous avons du caractère, alors que le code utilisé par l'algorithme n'en décrit qu'un aspect local lié à une progression le long du contour; la liste des naissances, bien qu'elle contienne déjà une certaine relation du local au global, se révèle insuffisante.

Le nombre des tests serait réduit si nous pouvions déduire simplement, à partir des informations connues localement sur le contour, une vision globale du caractère sous la forme d'un ordre partiel des parois. Nous démontrons que cette condition est satisfaite dans le corps de ce chapitre; ainsi, pour le caractère de la figure 4-2, le nombre des tests de type Bresenham a été ramené de 40 (ou 19) à 4!

Abscisse	Tests réels	Tests (max)	Liste triée (ordre croissant)
x_A	0	0	AB-AN
x_M	2	2	AB-AN-MN-ML
x_C	2	4	AB-CB-CD-AN-MN-ML
x_E	6	6	AB-CB-CD-AN-MN-ED-EF-ML
x_K	8	8	AB-CB-CD-AN-MN-ED-EF-ML-KL-KJ
x_G	10	10	AB-CB-CD-AN-MN-ED-EF-ML-KL-GF-GH-KJ
x_I	12	12	AB-CB-CD-AN-MN-ED-EF-ML-KL-GF-GH-KJ-IJ-IH
Cumul :	40	42	

Tableau 1-3

Remarque : Le tri des parois est effectué par un algorithme séquentiel. L'utilisation d'un algorithme de tri rapide entraînerait 19 tests sur la tache de la figure 1-2, mais semble peu justifiée en regard du petit nombre d'éléments à classer.

Certes la détermination de la structure globale prend elle-même du temps, car la manipulation logique des êtres géométriques que nous allons définir n'est pas gratuite... Bien qu'elle n'ait pas fait l'objet d'une réalisation sur machine, elle constitue malgré tout une voie très prometteuse.

Le lecteur attentif se rendra compte de l'analogie profonde entre ce problème et les grands problèmes classiques d'élimination des faces cachées : si nous considérons un caractère comme la section plane à $Z = Z_0$ d'un volume de R^3, la détermination des numéros d'ordre correspond alors à un calcul a priori des numéros des faces,tels qu'ils sont définis dans l'algorithme de NEWELL. Tant que la structure globale que nous allons décrire ne varie pas sur un intervalle $[Z_0,Z_0+H]$, on s'aperçoit qu'il est inutile de recalculer à chaque pas les numéros des faces. Pour la plupart des solides à représenter, cette propriété sera vérifiée sur de larges plages. Signalons qu'à notre connaissance, aucune référence bibliographique n'existe sur le sujet. Citons cependant (BEG 72) pour les généralités de géométrie différentielle.

II.- DEFINITIONS ET RESULTATS GENERAUX.

1.- Représentation intrinsèque des caractères.

Nous nous plaçons dans R^2 muni de sa structure euclidienne orientée, et d'une base orthogonale directe. Le choix des axes de la figure 4-4 est peu habituel, nous le justifions par le fait qu'il représente bien un balayage vidéo par ligne, du type de celui de la télévision. Dans ce qui suit, un **caractère** désignera un compact C de R^2, dont le bord $\dot{C}$ est formé d'une seule composante connexe de classe C^∞, sans points doubles et de longueur finie (fig. 4-4). C'est une sous-variété différentielle simplement connexe de dimension 2. Cette définition très formelle amène les remarques ci-dessous :

* La notion de caractère doit être vidée de toute connotation d'ordre sémantique pour garder le maximum de généralité; nous réintroduirons plus loin certaines restrictions destinées à simplifier les algorithmes.

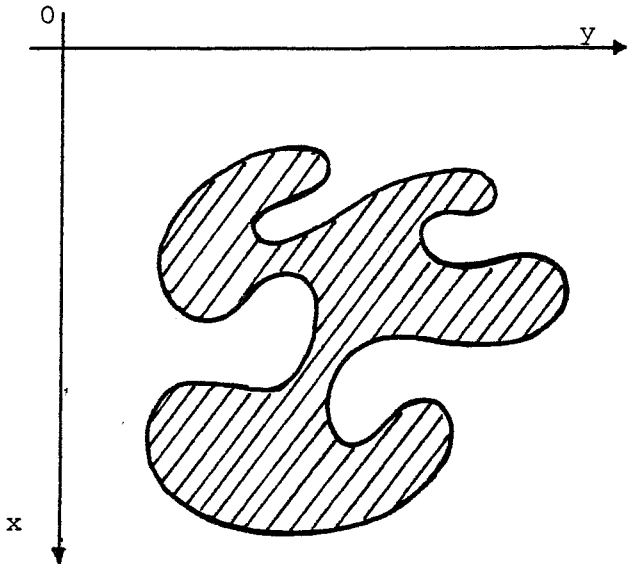


Figure 4-4

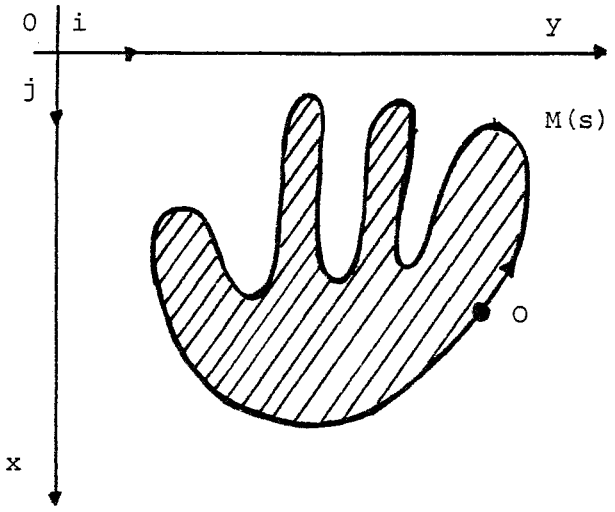


Figure 4-5

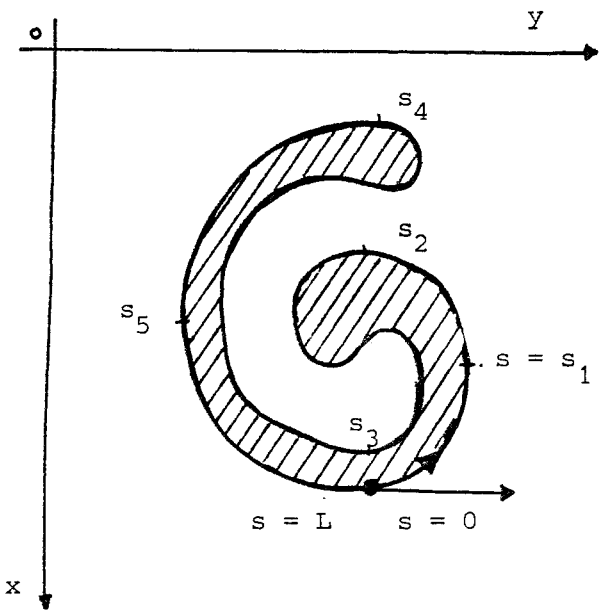


Figure 4-6

* L'interdiction de points doubles sur ∂C est à la base même de la théorie que nous allons développer. Dans notre cas, ∂C est donc une courbe fermée simple (un lacet).
* L'hypothèse ∂C de classe C^∞ est bien trop forte, mais permet d'utiliser simplement les théorèmes de la géométrie différentielle pour ramener le raisonnement à l'essentiel. En fait, tous les résultats pourraient être facilement extrapolés à un bord de classe C^0, C^1 par morceaux, cas des contours codés sous forme d'une suite d'arcs de cercle et de segments de droite.
* Le bord d'un caractère au sens ordinaire du terme peut comporter plusieurs composantes connexes. Toutefois les caractères romains de corps de texte n'admettent qu'une seule composante non convexe, et nous montrerons ci-après comment traiter de telles formes.

Supposée connexe, la frontière ∂C d'un caractère peut être représentée comme une fonction périodique C^∞ de **l'abscisse curviligne** s, à partir d'une **origine** O quelconque, dans le **sens canonique**, qui laisse l'intérieur à gauche lorsqu'on se déplace dans le sens des s croissants (fig. 4-5). La période de M(s) est égale à la longueur L de C, supposée finie et différente de zéro.

La représentation intrinsèque est bien adaptée au problème que nous posons, dans la mesure où elle met en valeur l'information locale, telle que nous pouvons l'extraire facilement de l'anneau codant le contour du caractère.
Le passage au global peut alors se faire par intégration de caractéristiques locales le long de ∂C. Nous utiliserons surtout la **variation totale de la tangente** (ou angle de variation de la tangente) $\theta(s) \in \mathbf{R}$ qui représente la variation totale de l'angle (M'(0),M'(t)) pour s passant de 0 à t (fig. 4-6). Nous supposerons que pour une orientation donnée, l'origine des abscisses curvilignes soit telle que $(M'(0),j) = 0$.

2.-Parois.

Nous appelons **point de naissance** (resp. **point de mort**) un minimum local en x sur ∂C (resp. maximum). Si l'on fait intervenir le signe de la courbure algébrique K en ces points, on obtient les 4 cas de la figure 4-7. En tout point

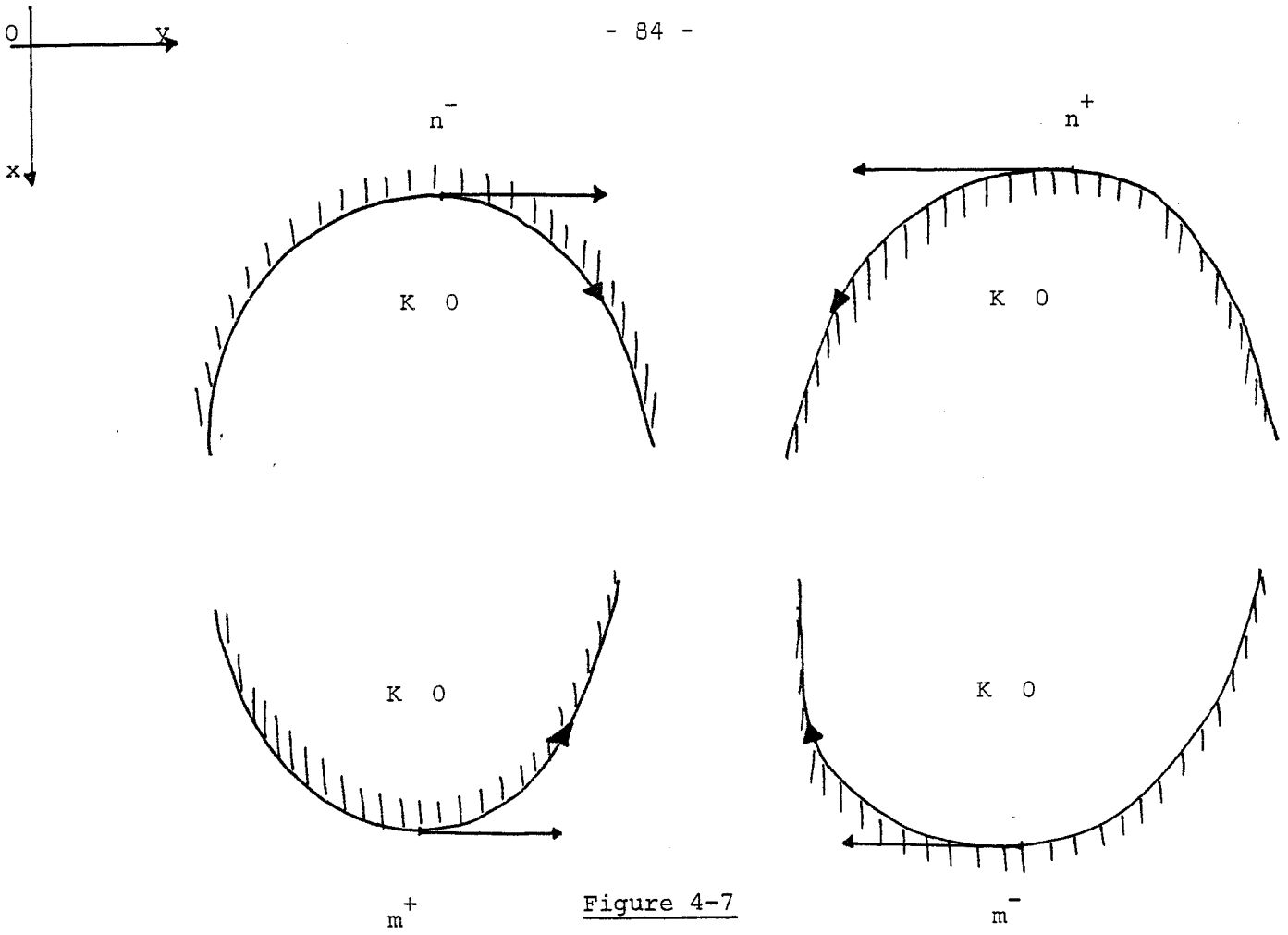
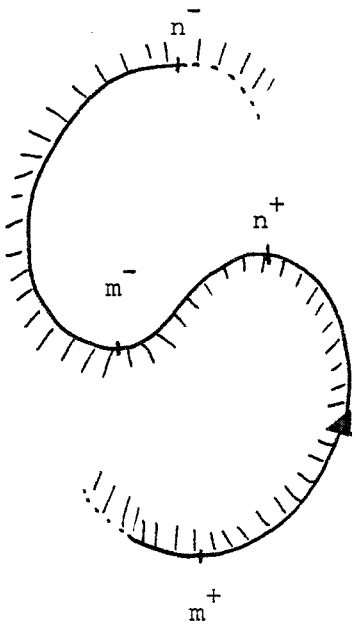


Figure 4-7



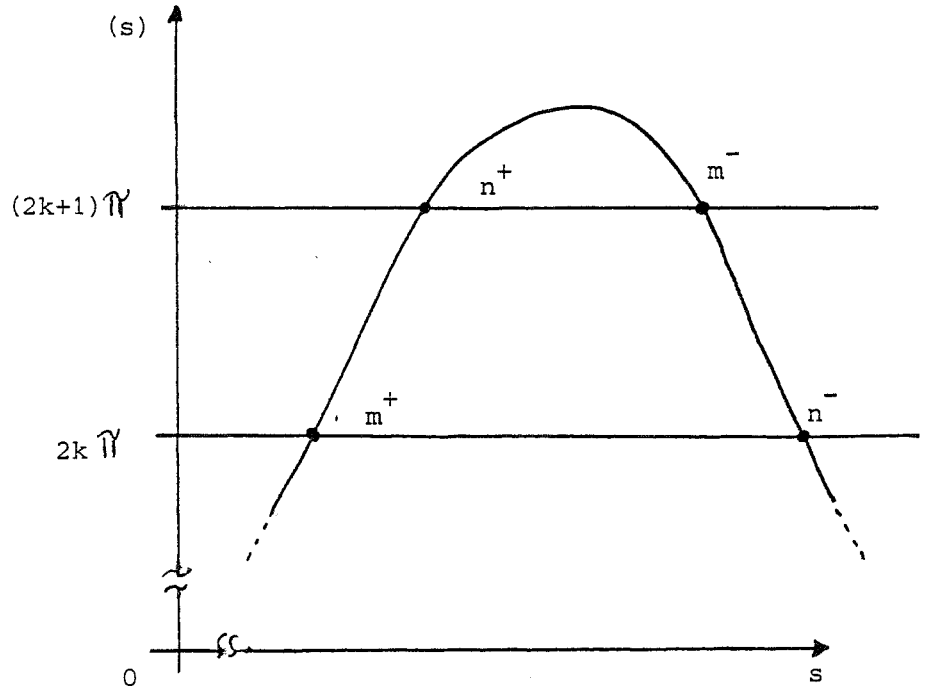


Figure 4-9

θ	θ' 0 <	θ' 0 >
2kπ	m^+	n^-
(2k+1)π	n^+	m^-

Tableau 4-8

extrémal M(s), nous trouvons $\theta(s) = n\pi$. Par contre, il peut exister des points non extrémaux qui vérifient $\theta(s) = n\pi$ et qui correspondent aux points d'inflexion de ∂C à tangente horizontale. Dans la mesure où C est compact, il y a au moins un point de mort de type m^+ sur ∂C supposé C^∞; nous le prendrons comme origine du contour. Si l'on trace la fonction $\theta(s)$, les points de mort et de naissance correspondent alors aux points d'intersection de son graphe avec les droites $\theta = n\pi$; le tableau 4-8 donne en fonction de la parité de n et du signe de θ' le type de l'extremum. La figure 4-9 illustre ces conventions et ces résultats.

Un théorème célèbre de géométrie différentielle, le **théorème de la tangente tournante** ou **Umlaufsatz** va nous être utile plus loin; il indique que le long d'une courbe fermée simple de classe C^2 de $\mathbf{R}^2$, la variation totale de la tangente vaut 2π. Dans notre cas, nous avons donc

$$\theta(L) = \theta(0) + 2\pi = 2\pi.$$

La représentation $\theta(s)$ permet de déduire, à partir de l'Umlaufsatz, et par de simples considérations de continuité et de périodicité, les résultats suivants :

* Le nombre de points de mort est fini et au moins égal à 1
* Entre deux points de mort (resp. naissance) consécutifs, il y a un point de naissance (resp. mort).
* Il y a autant de points de mort que de points de naissance.
* Lorsqu'il n'y a qu'un seul point de mort, et donc un seul point de naissance, ils sont tous deux de type +. On montre que c'est une condition nécessaire et suffisante pour que C soit convexe.
* Le nombre d'intersections du graphe de $\theta(s)$ avec les droite $\theta = k\pi$ est pair sur $[0,L[$, à condition de compter pour deux les points doubles

  $\theta(s) = k\pi$ et $\theta'(s) = 0$

  qui correspondent aux points d'inflexion de ∂C à tangente horizontale.

Un arc de contour compris entre un point de naissance et un point de mort s'appelle une **paroi.** Les parois sont les plus grands arcs de ∂C monotones en x. Une paroi est **directe** (resp. **inverse**) si elle est parcourue dans le sens canonique de son point de naissance à son point de mort (resp. de son point de mort à son point de naissance). En un point de naissance (resp. de mort) se concatènent deux parois; l'une est directe,l'autre inverse. Nous en déduisons que le nombre des parois est pair. Le tableau 4-10 classifie les parois en fonction du type des points

Tableau 4-10

De \ Vers	n^+	n^-	m^+	m^-
n^+			SOURCE D.	BOSSE G.D.
n^-			BOSSE D.D.	PUITS D.
m^+	PUITS I.	BOSSE D.I.		
m^-	BOSSE G.D.	SOURCE I.		

G. : GAUCHE
D. : DROITE
D. : DIRECT
I. : INVERSE

Tableau 4-11

	SD	SI	PD	PI	BDD	BDI	BGD	BGI
SD				X		X		
SI			X		X			
PD		X						X
PI	X						X	
BDD				X		X		
BDI			X		X			
BGD		X						X
BGI	X						X	

S : SOURCE
P : PUITS
B : DEMI-BOSSE

X : Concaténation possible

extrema; nous y avons indiqué une mesure importante, la variation $\Delta\Theta$ de la tangente le long de la paroi : si on rentre dans la paroi en $s = s_{min}$ et qu'on en sort en $s = s_{max}$ (dans le sens canonique, $s_{max} > s_{min}$), alors

$$\Delta\Theta = \Theta(s_{max}) - \Theta(s_{min}) \quad .$$

Le long d'une paroi, $\Delta\Theta$ prend ses valeurs dans $(+\pi, -\pi, 0)$. Les concaténations valides de deux parois sont résumées dans le tableau 4-11, que nous utiliserons souvent par la suite en tant que référence. Signalons quelques résultats intéressants à propos des parois :

* Le nombre de points d'inflexion de ∂C le long d'un puits ou d'une source est pair; le nombre de points d'inflexion sur une $\frac{1}{2}$-bosse est impair.
* Appelons point de naissance orthogonal $N^{\perp}$ et point de mort orthogonal $M^{\perp}$ les extrema locaux en y sur ∂C; alors
  le long d'une source : $\#N^{\perp} = \#M^{\perp} + 1$
  le long d'un puits : $\#M^{\perp} = \#N^{\perp} + 1$
  le long d'une $\frac{1}{2}$-bosse : $\#M^{\perp} = \#N^{\perp}$

Soit P une paroi de ∂C; nous noterons ΔX_P sa projection sur l'axe Ox.

Définition : Deux parois P et Q de ∂C sont **indifférentes** si et seulement si

$$\Delta X_P \cap \Delta X_Q = \{\emptyset\} \quad .$$

3.-L'ordre de balayage.

Appelons **droite de balayage en x** la droite d'équation $x = \hat{x}$. Soient $M(s_0)$, $M(s_1)$,...$M(s_{2k-1})$ les points d'intersection de ∂C avec la droite de balayage en $\hat{x}$, indiqués dans le sens des s croissants. Nous dirons que $M(s_i) = M_i(\hat{x},y_i)$ est devant $M(s_j) = M_j(\hat{x},y_j)$, et nous noterons $M_i < M_j$, si $y_i < y_j$. Cette relation définit **l'ordre de balayage;** c'est un ordre total dans $M_0, M_1, \ldots M_{2k-1}$.

Soient deux parois P et Q non indifférentes, et $x_0 = \text{Min}\,(\Delta X_P \cap \Delta X_Q)$. La droite de balayage en x_0 coupe P en P_{x_0} et Q en Q_{x_0} (en fait l'un de ces deux points est un point de naissance). Supposons que l'on ait $P_{x_0} < Q_{x_0}$. Alors le fait que le contour ∂C n'ait pas de points doubles se traduit par la propriété :

$$\forall x \in \Delta X_P \cap \Delta X_Q \quad , \; P_x < Q_x \; .$$

La décomposition du contour en parois est compatible avec l'ordre de balayage : nous dirons que P est devant Q, et nous noterons $P \prec Q$.

Nous venons de définir une relation d'ordre sur l'ensemble des parois d'un caractère. Le problème principal peut maintenant être posé :

> Etant donné un caractère C et ses parois $P_0, P_1, \ldots P_{2k-1}$, pouvons-nous déterminer une relation d'ordre total sur l'ensemble des parois, compatible avec la définition précédente, à partir de la description séquentielle du contour sous la forme d'un anneau ?

Nous savons qu'une telle relation existe, puisque nous sommes en mesure de l'exhiber (cf. I.1). Par contre, nous sommes à la recherche d'une méthode rapide et logique, qui minimise le nombre de tests "point contre paroi" en utilisant toute l'information dont nous disposons sur le contour.

III.-ALGORITHMES.

1.-Ordre local - Lobes.

Deux parois qui se concatènent en un point de naissance (n^+, n^-) ou de mort (m^+, m^-) sont toujours comparables (fig. 4-12). Nous en déduisons le tableau 4-13 qui exprime, en fonction du type d'une paroi, l'information sur les deux parois qui lui sont adjacentes; nous vérifions qu'il est toujours possible d'ordonner trois parois adjacentes entre elles lorsque la paroi centrale est une $\frac{1}{2}$-bosse.

Un cas particulier important est celui des **lobes** : ils sont composés de trois parois, un puits, une $\frac{1}{2}$-bosse et une source, et présentent un **creux** et une **bosse** (fig. 14). Les types de $\frac{1}{2}$-bosses déterminent quatre types de lobes, dont les trois parois sont automatiquement triées entre elles. Les lobes interviendront principalement dans l'étude de la forme réduite du caractère.

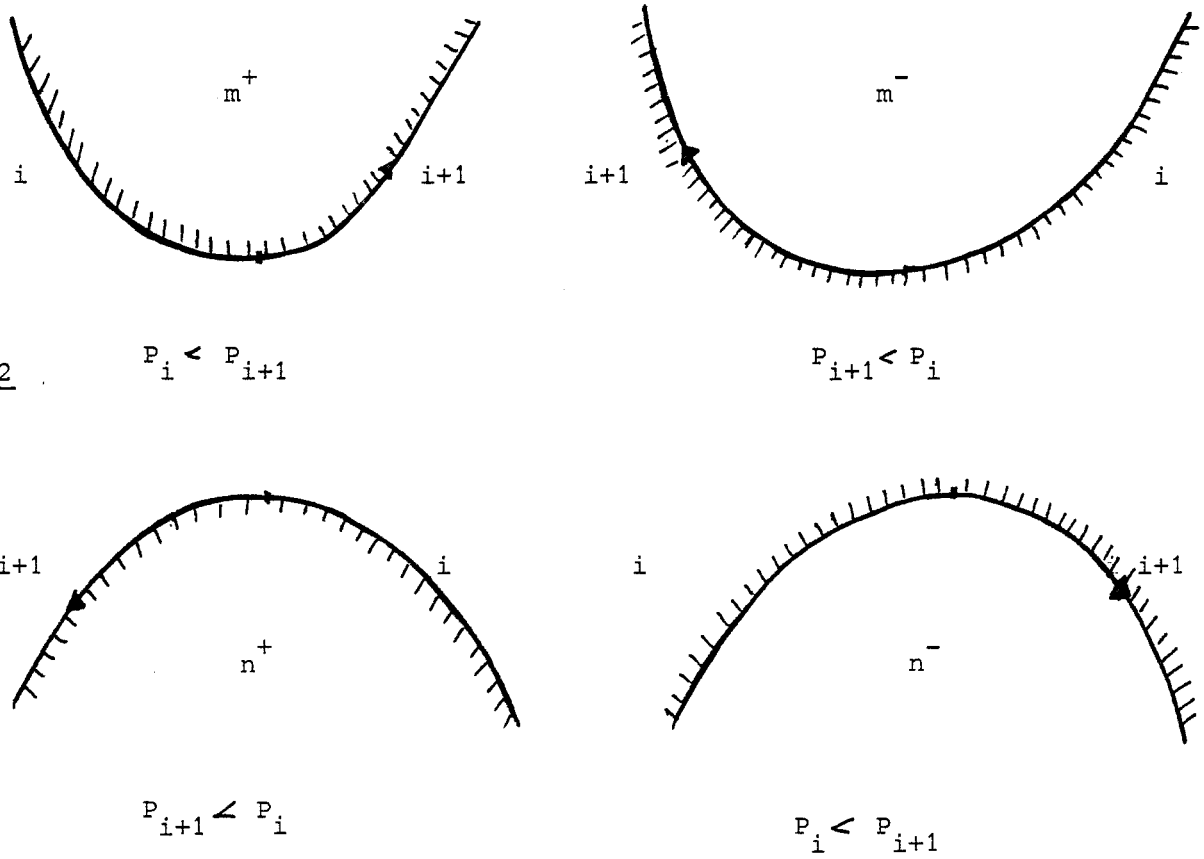


Figure 4-12

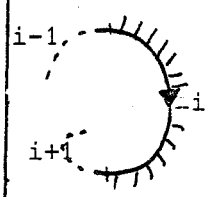
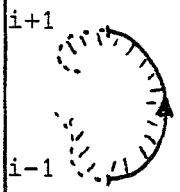
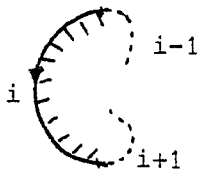
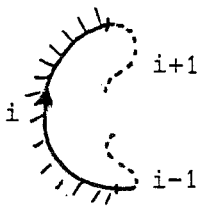
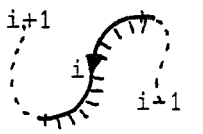
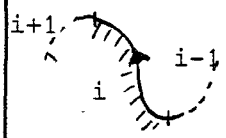
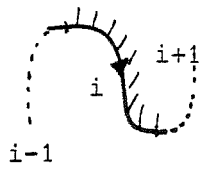
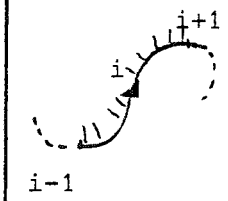
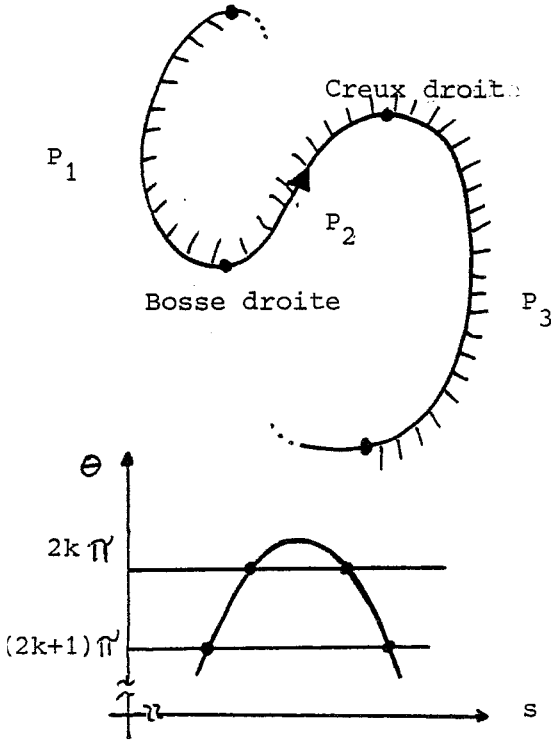
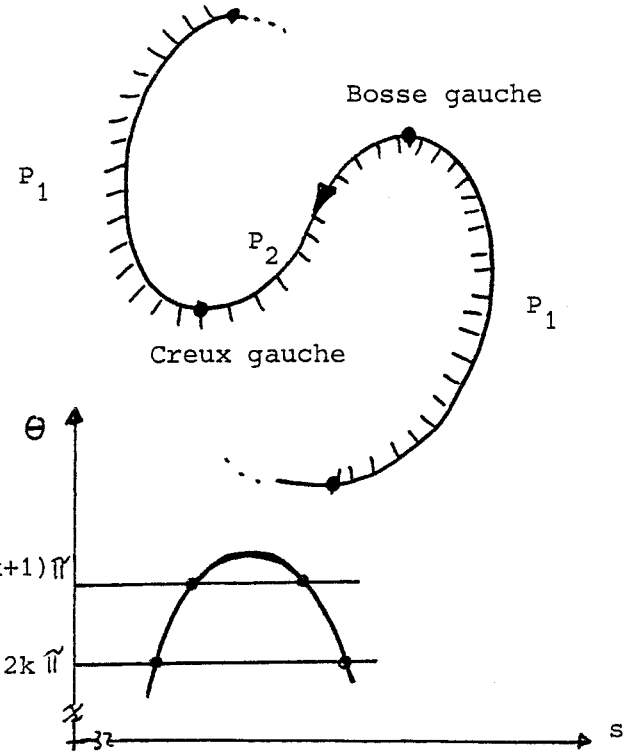
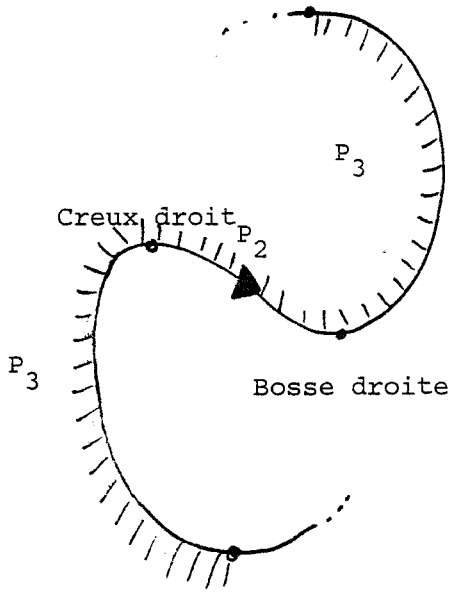
Direct	Inverse	Type de paroi	Ordre local
		PUITS	P_i maximum local
		SOURCE	P_i minimum local
		DEMI-BOSSE GAUCHE	$P_{i+1} < P_i < P_{i-1}$
		DEMI-BOSSE DROITE	$P_{i-1} < P_i < P_{i+1}$

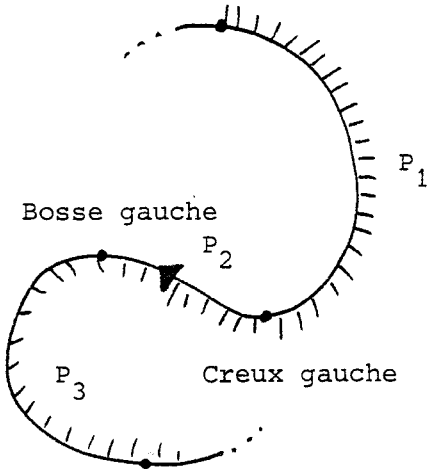
Tableau 4-13

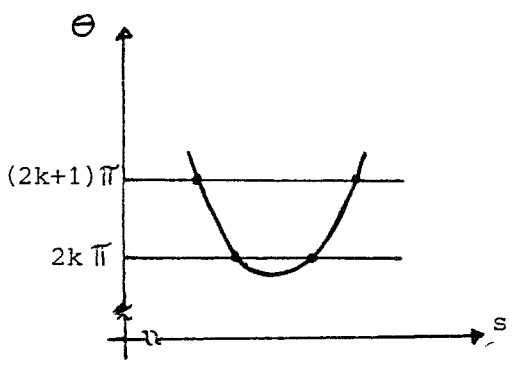
Figure 4-14

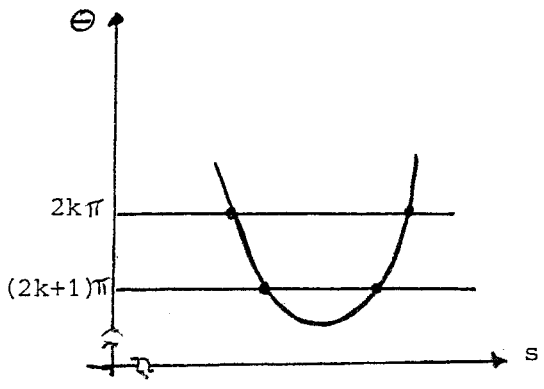

Creux droit
P1
P2
Bosse droite
P3
θ
2kπ
(2k+1)π
s


Bosse gauche
P1
P2
P1
Creux gauche
θ
(2k+1)π
2kπ
s


P3
Creux droit
P2
P3
Bosse droite


P1
Bosse gauche
P2
P3
Creux gauche


θ
(2k+1)π
2kπ
s


θ
2kπ
(2k+1)π
s

2.-Ordre global - Chameaux.

Un **chameau gauche** (resp. **chameau droit**)est une suite P_i, P_{i+1}, ...P_{i+k} de $\frac{1}{2}$-bosses gauches (resp. droites). La figure 4-15 représente deux chameaux et les graphes $\theta(s)$ correspondants. Remarquons que le long d'un chameau, la variation totale de la tangente $\Delta\theta$ est nulle.

Un chameau gauche est totalement ordonné dans l'ordre inverse :

$$P_{i+k} < P_{i+k-1} < \ldots\ldots < P_i .$$

Un chameau droit est totalement ordonné dans l'ordre direct :

$$P_i < P_{i+1} < \ldots\ldots < P_{i+k} .$$

En suivant le contour ∂C, on rentre dans un chameau gauche par un puits P_{i-1}, et l'on en sort par une source P_{i+k+1}; respectivement, on rentre dans un chameau droit par une source P_{i-1}, et l'on en sort par un puits P_{i+k+1}. Il est intéressant de remarquer que les parois P_{i-1} et P_{i+k+1} sont également rangées dans l'ordre (direct ou inverse) avec les parois proprement dites du chameau.

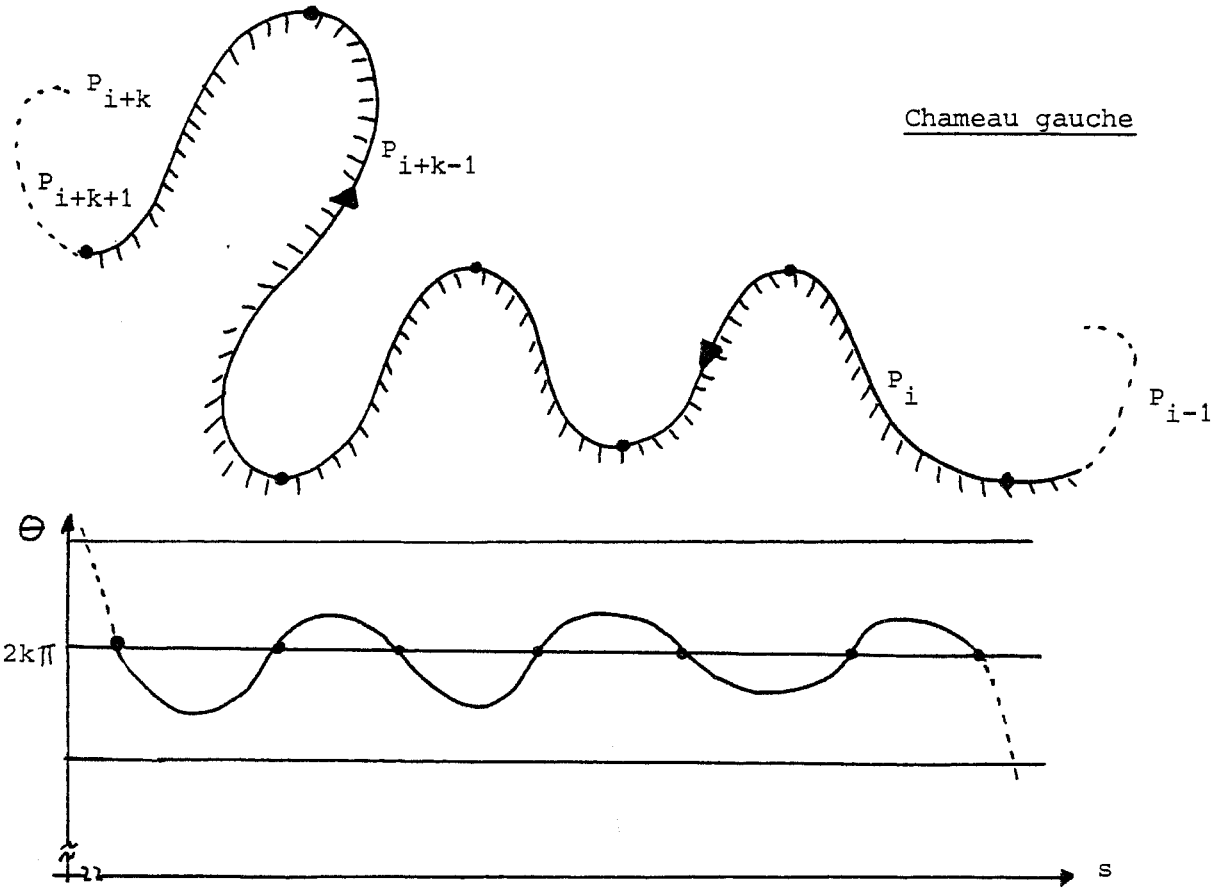
Si nous appelons **graphe dual** de C le graphe orienté de la relation d'ordre $<$ dans l'ensemble des parois - une paroi est alors représentée par un sommet du graphe -, les sources et les puits de ∂C sont les sources et les puits du graphe dual, du moins à l'étape de l'analyse où nous sommes; cela justifie la dénomination.

3.- Ordre global - Spirales.

Nous appelons **spirales** une alternance régulière de puits et de sources; nous allons montrer que les parois d'une spirale peuvent être classées sans effectuer de tests de type Bresenham, ainsi que les parois de deux spirales adjacentes. Nous définissons donc une seconde structure - après les chameaux - dont les éléments sont naturellement classés.

La figure 4-16 représente les deux classes de spirales, et les graphes $\theta(s)$ leur correspondant : spirales positives, formées de sources directes et de puits inverses, et spirales négatives, composées de sources inverses et de puits directs. Une spirale peut présenter une ou deux **volutes.** Le point initial (resp. final) d'une

Figure 4-15


Chameau gauche
P_{i+k}
P_{i+k+1}
P_{i+k-1}
P_i
P_{i-1}
Θ
$2k\pi$
s

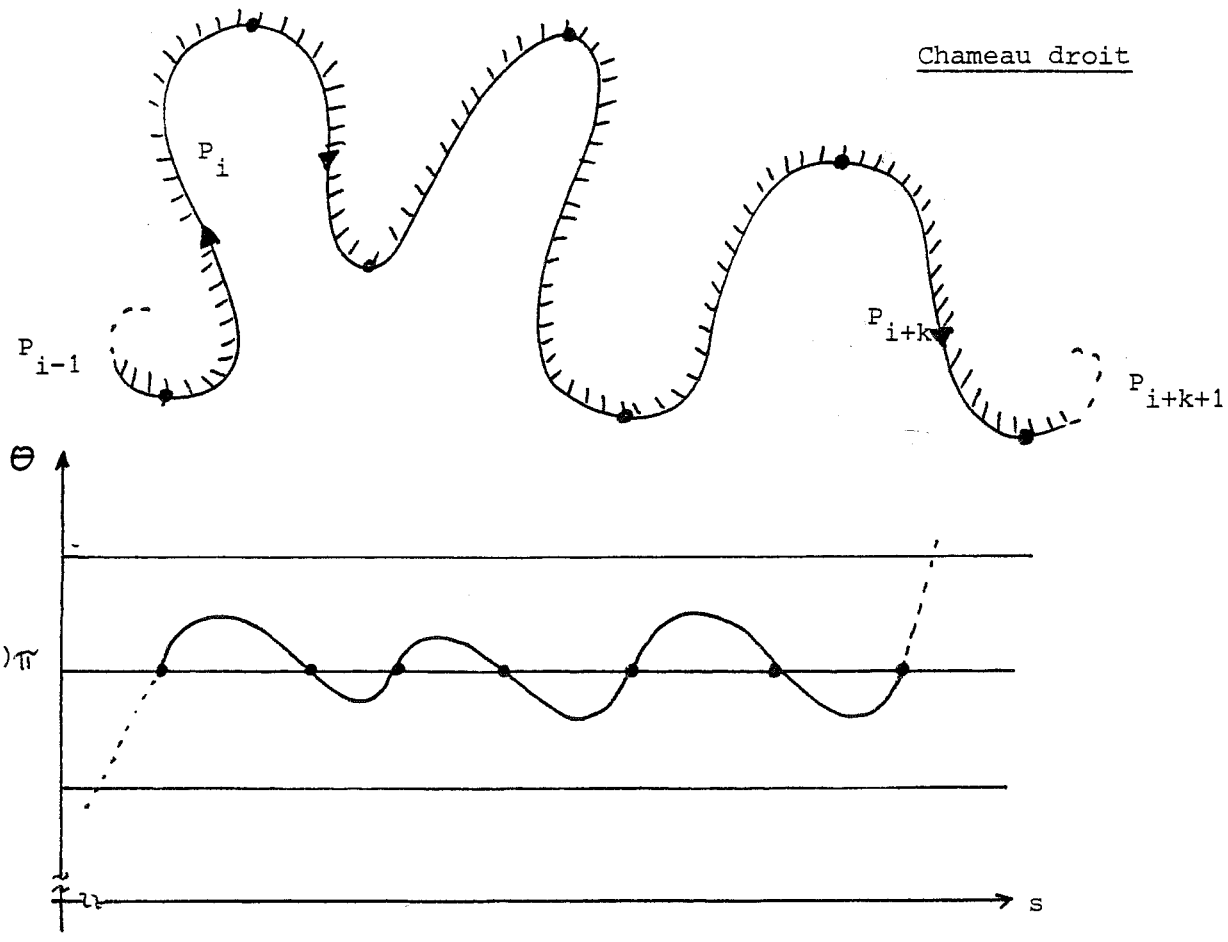
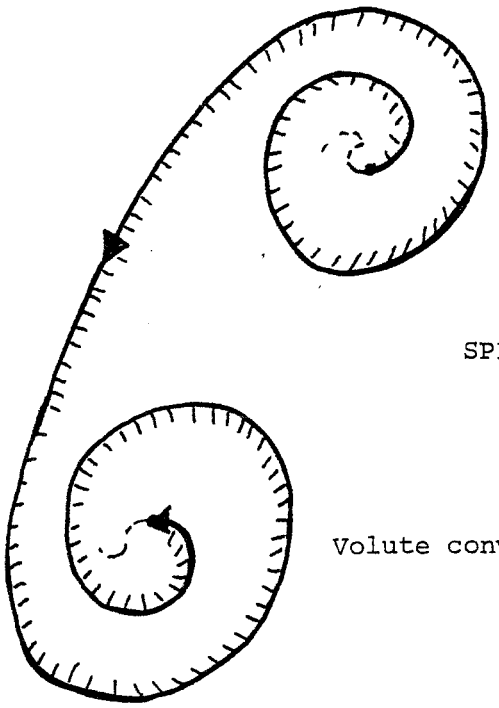

Chameau droit
P_i
P_{i-1}
P_{i+k}
P_{i+k+1}
Θ
$(2k+1)\pi$
s

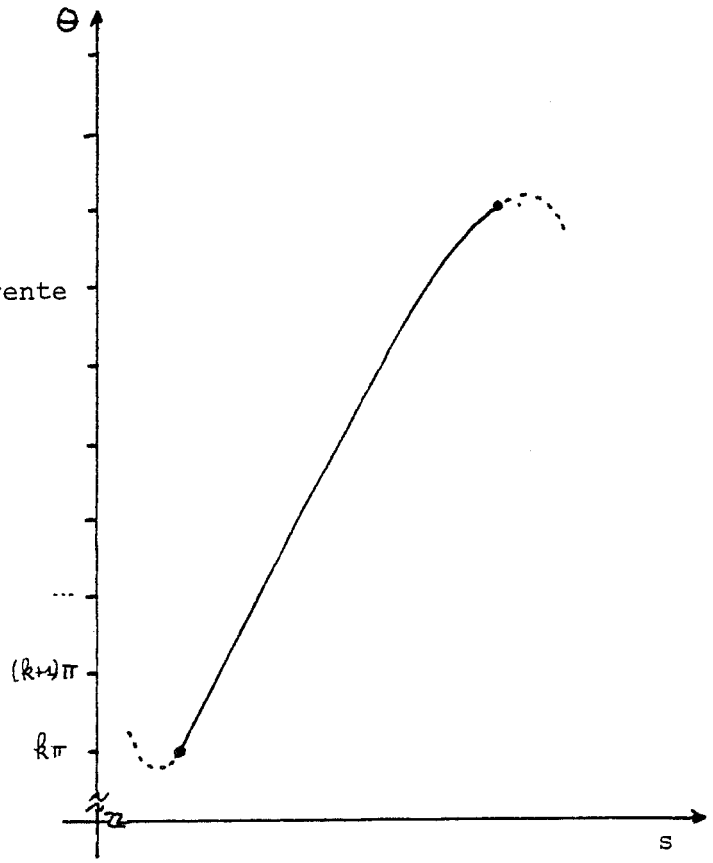
Figure 4-16

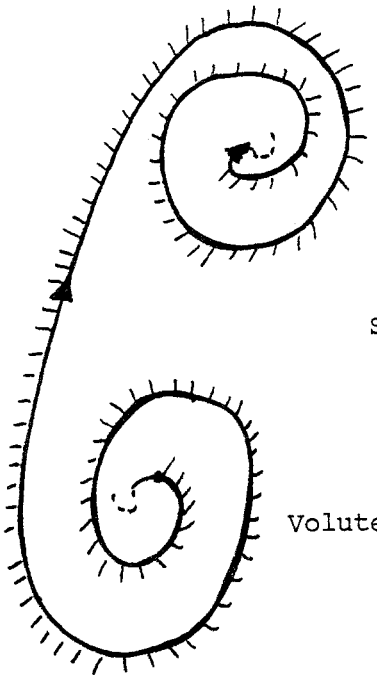


Volute divergente

SPIRALE POSITIVE

Volute convergente

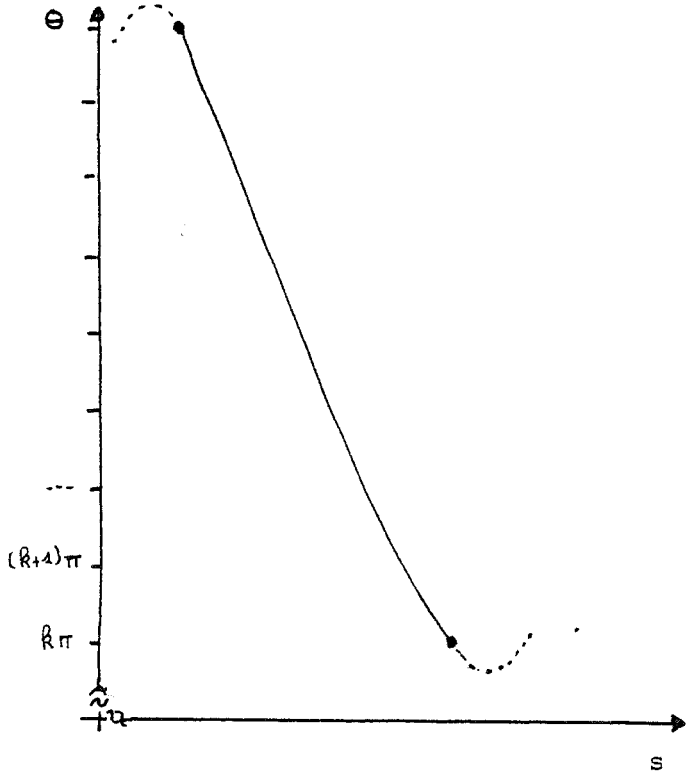




Volute convergente

SPIRALE NEGATIVE

Volute divergente



volute divergente (resp. convergente) est son **ombilic;** lorsque la spirale est composée d'une seule volute, une de ses extrémités est un ombilic, et l'autre est **libre.**

Le long d'une spirale positive (resp. négative) formée de n parois, la variation totale de la tangente $\Delta\theta$ vaut $+n\pi$ (resp. $-n\pi$).

Classement des parois d'une spirale.

Nous sommes placés au début d'une spirale; de simples tests sur les points de naissance et de mort (et éventuellement sur les points de naissance et de mort orthogonaux) vont nous permettre d'en déduire la nature, puis de la trier. L'algorithme commence par détecter la paroi fondamentale de la spirale : s'il y a deux volutes, c'est la paroi qui les relie; dans les autres cas, c'est celle qui contient l'extrémité libre. Il reste ensuite quelques tests à opérer sur les extrema orthogonaux lorsque la spirale présente deux volutes non indifférentes.

A.- Détection du point haut M_H :

Appelons M_0, M_1, ..., M_n les extrémités des parois constituant une spirale, et x_0, x_1, ... , x_n leurs abscisses. Le point haut M_H est le point de naissance qui vérifie

$$x_H = \text{Min } x_i \quad .$$

On le détermine en activant l'algorithme suivant :

si M_0 est un point de naissance alors $x_H = x_0$

sinon $x_H = x_1$;

si x_H x_{H+2} alors **volute convergente;** on s'arrête.

sinon **volute divergente :**

tant que x_{H+2} x_H faire $H = H + 2$.

B.- Spirales à une volute

* M_H est le premier point de la spirale, qui est alors formée d'une seule volute convergente. Le tableau 4-17 résume les différentes dispositions

Tableau 4-17

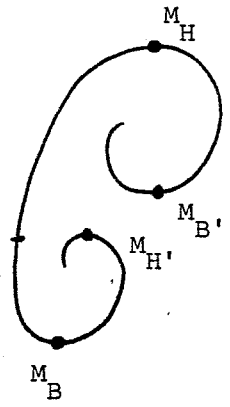
Sens \ Ombilic	Naissance	Mort
Positif	 2n parois $[P_{2m+1}] < P_{2m-1} < \dots < P_1 < [P_{-1}] < P_2 < \dots P_{2m}$	 2n+1 parois $[P_{2m+2}] < P_{2m} < \dots < P_2 < [P_{-1}] < P_1 \dots < P_{2m+1}$
Négatif	 2n parois $P_{2m} < P_{2m-2} \dots < P_2 < [P_{-1}] < P_1 \dots < P_{2m+1} [P_{2m+1}]$	 2n+1 parois $P_{2m+1} < P_{2m-1} \dots < P_1 < [P_{-1}] < P_2 \dots < P_{2m} < [P_{2m+2}]$

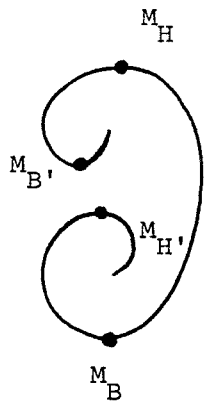
M_H est le dernier point : volute divergente.

Sens \ Ombilic	Naissance	Mort
Positif	 2n parois $P_{-1} < P_3 \dots P_{2m-1} < [P_{2m+2}] < P_{2m} < P_{2m-2} \dots < P_2 < [P_1]$	 2n+1 parois $P_1 < P_3 \dots P_{2m+2} < [P_{2m+2}] < P_{2m} < \dots P_2 < [P_{-1}]$
Négatif	 2n parois $[P_{-1}] < P_2 \dots < P_{2m} < [P_{2m+2}] < P_{2m-2} \dots < P_1$	 2n+1 parois $[P_{-1}] < P_2 < \dots P_{2m} < [P_{2m+2}] < P_{2m+1} < \dots P_3 < P_1$

M_H est le premier point : volute convergente.

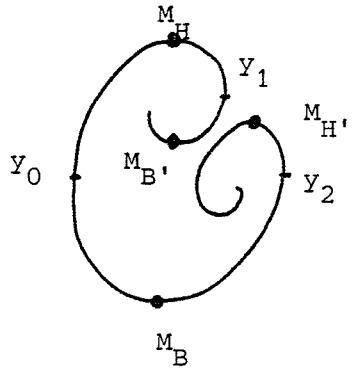
Remarque : Les parois de raccord P_{-1} et P_{2n+1} (ou P_{2n+2}) sont classées avec la volute.

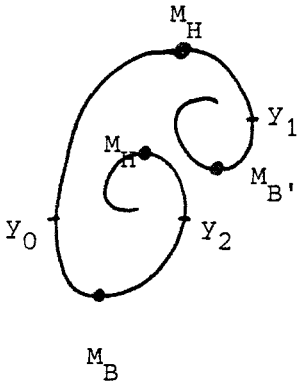




INDIFFERENCE

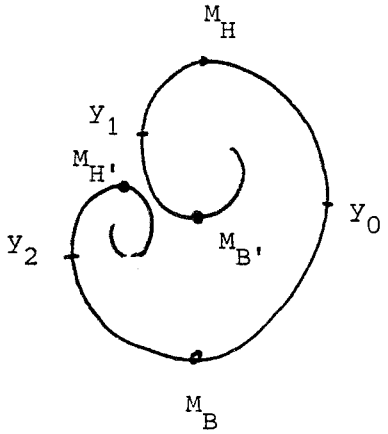
x_H $x_{B'}$ $x_{H'}$ x_B

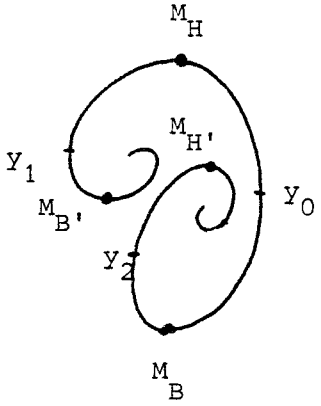




$M_H M_B$ minimum

(2 cas)





$M_H M_B$ maximum

Figure 4-18

possibles, en fonction du sens de la spirale et de la nature de l'ombilic.

* M_H est le dernier point de la spirale, qui est alors formée d'une volute divergente; (cf. tableau 4-17).

C.- Spirales à deux volutes

Si M_H n'est ni premier, ni dernier, définissons M_B tel que

$$x_B = \text{Max} (x_{H+1}, x_{H-1})$$

Si M_B est le premier point de la spirale, ou le dernier, une des volutes de la spirale est dégénérée et réduite à la paroi $M_H M_B$.

Dans les autres cas, définissons les points $M_{B'}$ et $M_{H'}$ (fig. 4.-18). La spirale est alors formée de deux volutes.

Si $x_H < x_{B'} < x_{H'} < x_B$, alors on a deux volutes indifférentes.

Si $x_H < x_{H'} < x_{B'} < x_B$, l'une des volutes est entièrement à gauche de l'autre.

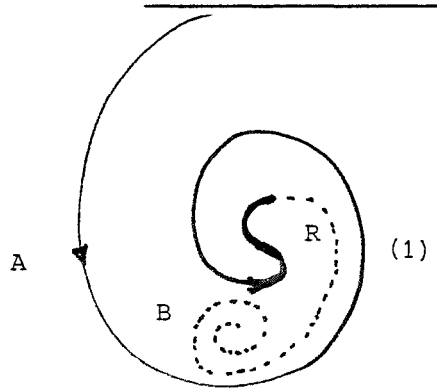
Comparons les positions des points de naissance orthogonaux y_0 (sur $M_H M_B$), y_1 (sur $M_H M_{B'}$), y_2 (sur $M_B M_{H'}$). Ces points sont en effet connus a priori comme noeuds d'interpolation de la représentation canonique que nous utilisons, et les deux tests nécessaires se font sans activer de Bresenham. Nous distinguerons donc 4 cas, suivant la position de l'extremum y_0 d'abord, puis celles des points y_1 et y_2.

Signalons pour terminer que les deux parois qui limitent la spirale (ce sont des $\frac{1}{2}$-bosses), peuvent être également incluses dans l'ordre partiel engendré, sans faire de test.

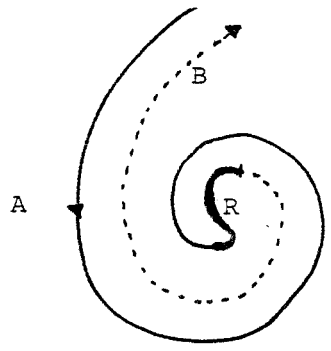
D.- Classement des parois de deux volutes adjacentes.

Deux volutes adjacentes sont nécessairement reliées par une $\frac{1}{2}$-bosse, et tournent en sens contraire. Leur concaténation peut se faire de trois manières :

* ombilic contre ombilic :



(1)



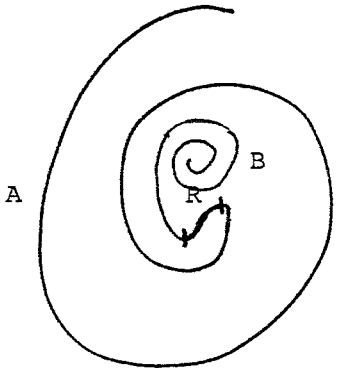
(2)

La volute B peut s'enkyster dans la volute A (1), ou bien permettre de sortir de A (2), cela à condition bien sûr que la variation de la tangente le long de B soit suffisante. Une condition nécessaire pour qu'une volute B déroule une volute A est que ;

$$\Delta\theta_B \geqslant \Delta\theta_A - \pi .$$

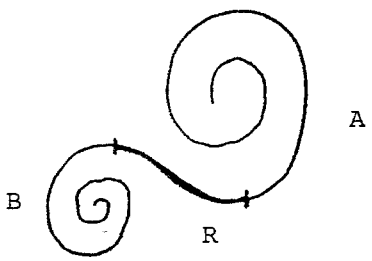
Nous formulons maintenant une hypothèse simplificatrice : il n'y a pas de spirales enkystées dans les formes que nous cherchons à reconnaître; toute volute est suivie (ou précédée) d'une autre volute qui permet d'en sortir (ou d'y rentrer). Cette hypothèse définit, en fait, une classe très large de formes, qui comprend toutes les formes usuelles.

* ombilic contre extrémité libre :

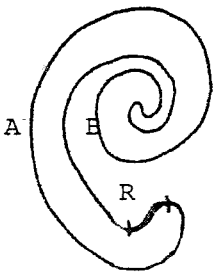


La spirale B est contenue dans la dernière spire de A et s'y enkyste. Ce cas est donc exclu par hypothèse.

* extremité libre contre extrémité libre :



(1)



(2)

Dans le cas (1), la $\frac{1}{2}$-bosse sépare les deux volutes; le classement est alors

trivial.

Dans le cas (2), l'une des spirales s'emboîte partiellement dans l'autre. Dès lors, l'hypothèse simplificatrice demande à ce que les deux spirales se referment complètement. On est donc ramené au cas "ombilic contre ombilic", seule possibilité à prendre finalement en compte. Si deux volutes sont connectées par leurs ombilics, sans kyste, les spirales A et B sont classées en prenant alternativement les parois triées de l'une et de l'autre; la disposition des deux parois de part et d'autre de leur raccord entraîne l'ordre d'emboîtement des deux volutes.

4.- Forme réduite.

Les deux paragraphes précédents ont décrit la résolution du classement des parois d'un chameau, ou de spirales par paires adjacentes. Ce n'est pas toujours suffisant; pour trier les parois résiduelles, nous pouvons remarquer qu'il suffit de les tester au niveau des points de naissance seulement (resp. des points de mort) : cela nous amène à comparer creux droits (n^-) contre bosses gauches (n^+) (respectivement creux gauches (m^-) contre bosses droites (m^+)).

Dans la mesure où $\Delta\Theta$ y est nul, nous pouvons considérer que les chameaux sont des "accidents" le long du contour; la structure générale du caractère subsiste lorsqu'on les élimine : un chameau formé d'un nombre pair de parois sera supprimé; un chameau formé d'un nombre impair de parois sera remplacé par une paroi unique, dite **paroi de raccord** (tableau 4-19). Dans la forme réduite que l'on obtient, ne subsistent que des spirales et des lobes qu'il reste bien sûr à classer. Pour ce faire, il est souvent impossible d'éviter des tests de type Bresenham, puisqu'il faut comparer "point contre paroi"; les figures sont souvent simples, et l'information disponible sur les parois (extrema orthogonaux par exemple) permet de prendre une décision rapide (fenêtrage). Les caractères de la figure 4-20 ont tous la même structure formelle : spirales, lobes, parois, et seuls les tests que nous venons de mentionner permettent de les distinguer; nous allons en faire une analyse complète.

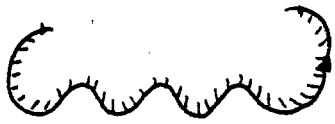
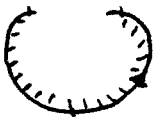
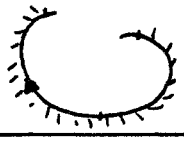
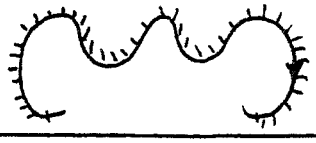
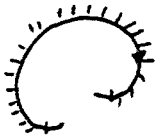
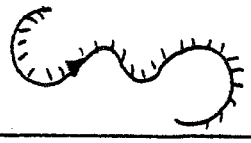
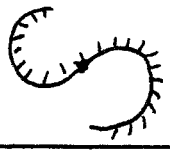
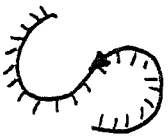
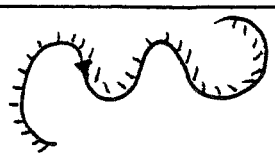
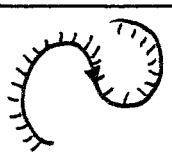
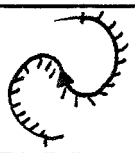
CHAMEAU	REDUCTION	RACCORD
		/
		/
		/
		/
		DEMI-BOSSE DROITE INVERSE
		DEMI-BOSSE GAUCHE DIRECTE
		DEMI-BOSSE DROITE DIRECTE
		DEMI-BOSSE GAUCHE INVERSE

Tableau 4-19

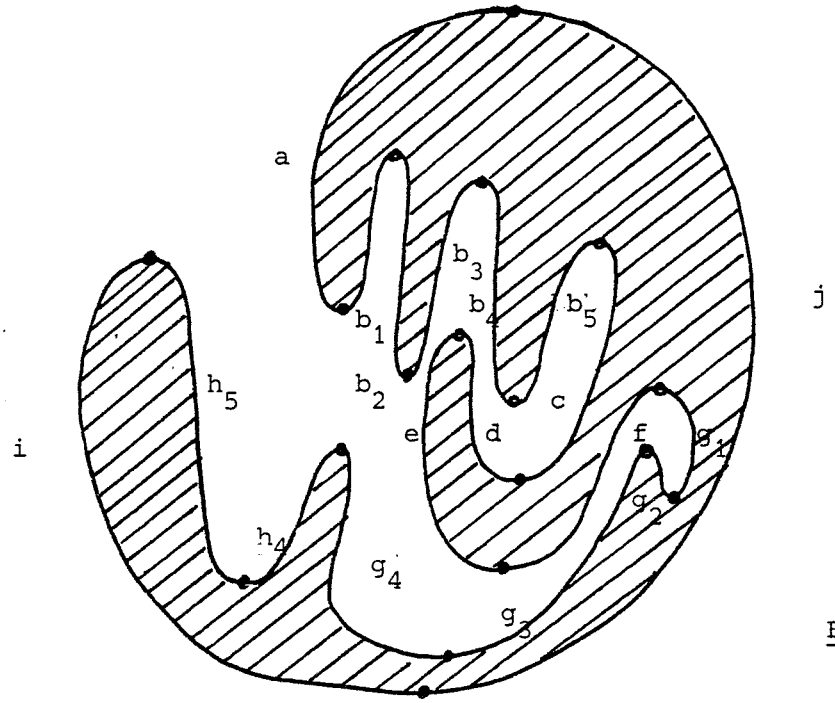
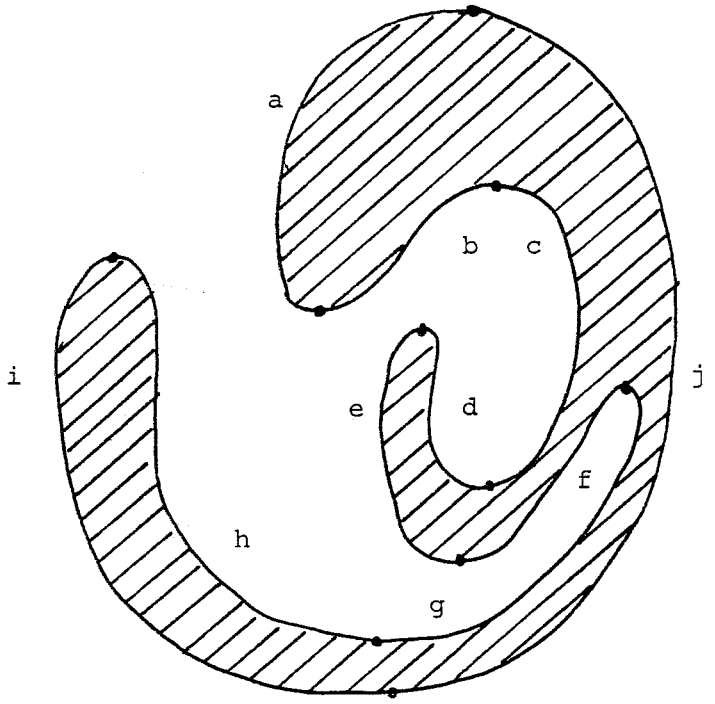
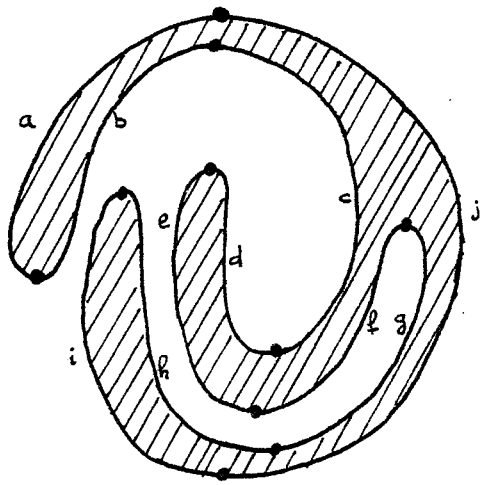


Figure 4-20 (1)

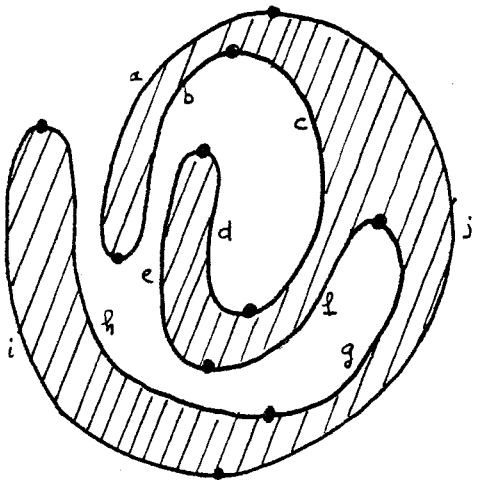
Un caractère



Forme réduite

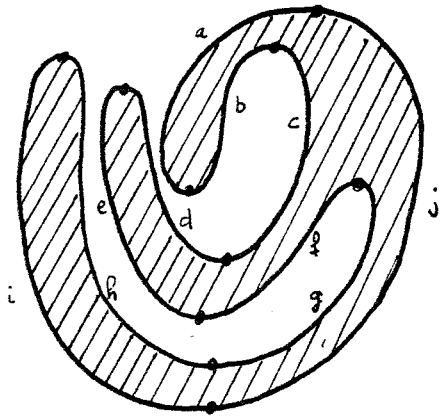


1

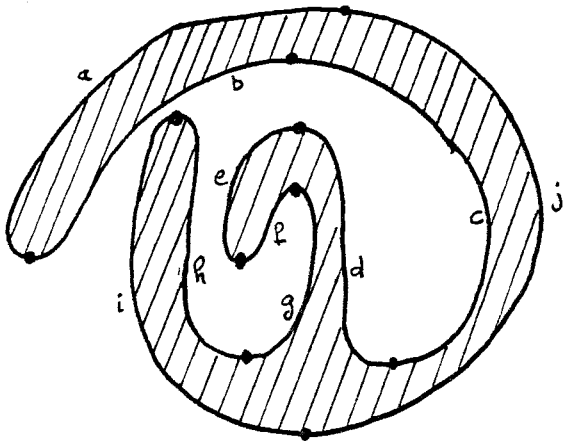


5

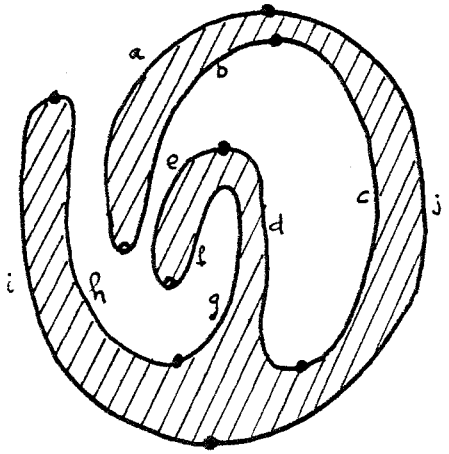
Figure 4-20 (2)



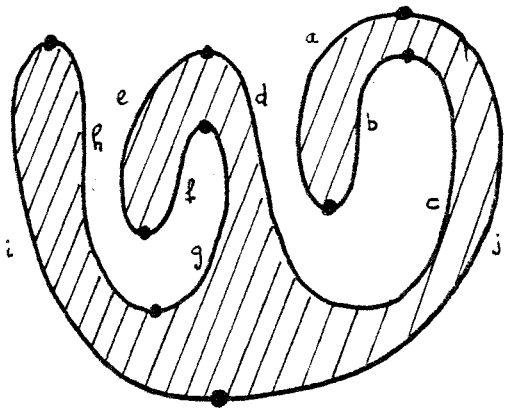
2



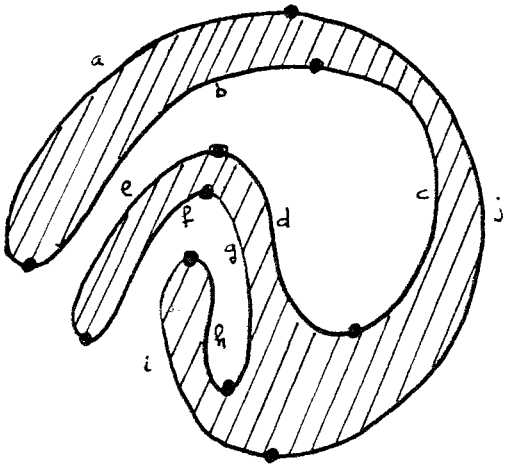
6



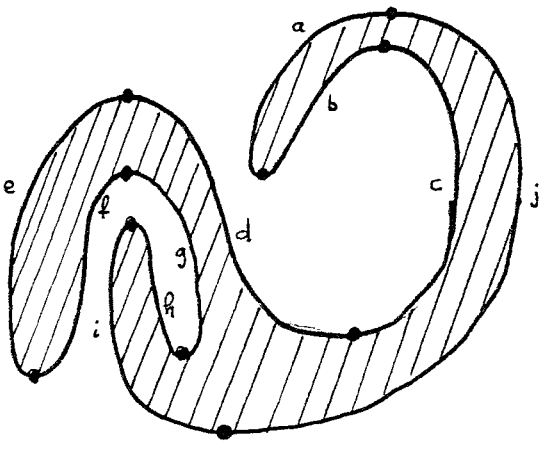
3



7



4



8

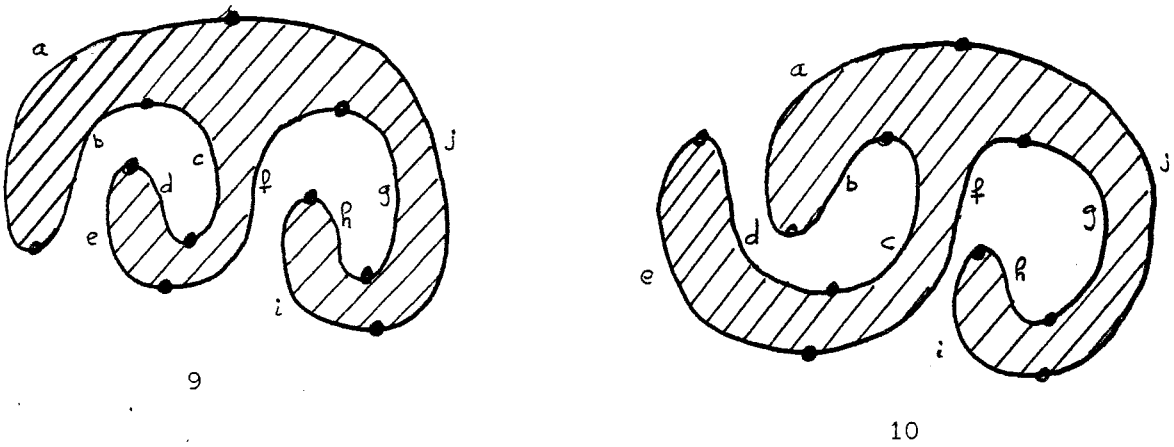


Figure 4-20 (3)

i-j-a : spirale positive

a-b-c ; e-f-g : lobes inverses droits (d'après le type de la $\frac{1}{2}$-bosse)

c-d-e ; g-h-i : lobes inverses gauches ("" "")

L'étude de la spirale i-j-a permet de déterminer les positions respectives des parois i, j, a, b et h :

soit A : $a \prec b \prec i \prec h \prec j$

soit $\overline{A}$: $i \prec h \prec a \prec b \prec j$

Les quatre lobes sont classés a priori :

$a \prec b \prec c$

$e \prec d \prec c$

$e \prec f \prec g$

$i \prec h \prec g$

h et b sont classés avec la spirale ; il ne reste donc que trois lobes indépendants, qu'il suffit de tester deux à deux.

* test a-b-c contre e-d-c :

B: $a \prec b \prec e \prec d \prec c$

$\overline{B}$: $e \prec d \prec a \prec b \prec c$

* test c-d-e contre e-f-g :

C: $e \prec f \prec g \prec d \prec c$

$\overline{C}$: $e \prec d \prec c \prec f \prec g$

* test e-f-g contre g-h-i :

D: $i \prec h \prec e \prec f \prec g$

$\overline{D}$: $e \prec f \prec i \prec h \prec g$

Les seize combinaisons possibles des résultats des tests n'aboutissent qu'à dix formes correctes; dans six cas, il y a apparition d'un cycle dans la description du classement, qui correspond, en fait, à une incohérence géométrique.

Nous énumérons ci-dessous la description des seize modalités, ainsi que les références correspondantes sur la figure 4-20

ABCD $i \prec h \prec a \prec b \prec e \prec f \prec g \prec d \prec c \prec j$ 3

ABCD cycle ($e \prec f \prec i \prec h \prec a \prec b \prec e$)

ABCD $i \prec h \prec a \prec b \prec e \prec d \prec c \prec f \prec g \prec j$ 5

$AB\bar{C}\bar{D}$	cycle (i<h<a<b< e<f<i)	
$A\bar{B}CD$	i<h< e<f<g<d<a<b<c <j	7
$A\bar{B}C\bar{D}$	e<f<i<h<g<d<a<b<c <j	8
$A\bar{B}\bar{C}D$	i<h< e<d<a<b<c <f<g <j	2
$A\bar{B}\bar{C}\bar{D}$	cycle (e<f<i<h<a<b<c <f)	
$\bar{A}BCD$	a<b< i<h<e<f<g<d<c<j	6
$\bar{A}BC\bar{D}$	a<b< e<f <i<h<g<d<c <j	4
$\bar{A}B\bar{C}D$	a<b<i <h <e<d <c <f< g <j	1
$\bar{A}B\bar{C}\bar{D}$	a<b<e <d <i<h<c <f<g<j	9
$\bar{A}\bar{B}CD$	cycle (a<b<i<h<e<d<a)	
$\bar{A}\bar{B}C\bar{D}$	cycle (a<b<i<h<g <d<a)	
$\bar{A}\bar{B}\bar{C}D$	e<d<a<b<c<f=i<h<g<j	10
$\bar{A}\bar{B}\bar{C}\bar{D}$	cycle (i<h< a <b <c<f<i)	

Dans le cas général où des lobes sont séparés par des spirales, il est nécessaire de tester tout lobe d'un sous-ensemble de lobes consécutifs contre tout lobe d'un autre sous-ensemble de lobes consécutifs. Les hypothèses de non-enkystage que nous avons formulées plus haut évitent les tests "lobe contre spirale".

Nous supposons que les caractères que nous traiterons comportent au plus deux spirales consécutives (l'une refermant l'autre); les k lobes résiduels de la forme réduite (mis à part l'ombilic) sont alors consécutifs; ils sont alors classés en au plus (k-1) tests.

5.- Réintroduction des chameaux.

L'étude que nous avons menée, porte sur la forme réduite du caractère; la réinsertion d'un chameau se fait par fusion de deux ordres partiels en un ordre unique, en exécutant des tests en nombre très limité, comme nous le verrons au cours des applications pratiques.

Un chameau $a\text{-}p_1\text{-}p_2\text{.....-}p_k\text{-}b$ avait été supprimé (ou remplacé par un raccord r) pour le tri de la forme réduite; le graphe totalement ordonné, avant la réintroduction du chameau, indique :

$$x < a < y \ldots\ldots\ldots\ldots\ldots x' < b < y'$$
$$\wedge \qquad\qquad\qquad\qquad\qquad \swarrow$$
$$p_1 < p_2 < \ldots\ldots p_{k-1} < p_k$$

Il suffit alors d'intercaler les parois du chameau en testant "creux contre bosse". La méthode appelle les remarques suivantes :

* Peu de tests sont en réalité nécessaires, de nombreuses parois étant soit indifférentes, soit automatiquement classées par fenêtrage.
* La détermination de la position de la bosse ou du creux d'un lobe par rapport à un chameau à n creux ou n bosses demande au plus n tests si l'on procède de manière séquentielle, $\log_2 n$ si l'on procède de manière dichotomique.
* La détermination des positions d'un chameau à n bosses contre un chameau à m creux demande au plus (n+m) tests de type Bresenham.

6.- Résumé de la méthodologie.

A) **Un parcours de** ∂C fournit une liste de parois avec toutes les caractéristiques que nous utiliserons :

* **type** : puits, source, $\frac{1}{2}$-bosse droite ou gauche
* **sens** : direct, inverse
* **projections** sur Ox (ΔX) et sur Oy (ΔY)
* **extrema orthogonaux**

B) Détection, tri et élimination des chameaux.

C) Etude et positionnement des spirales

D) Tris résiduels sur la forme réduite

E) Réintroduction des chameaux.

L'outil logiciel le mieux adapté à la résolution de ce problème semble être la matrice de représentation du graphe dual, ou matrice de précédence. Soient P_0,

P_1, ..., P_{2n-1} , les parois composant ∂C. La matrice B booléenne de dimension 2n x 2n définie par

$B(i,j) = 1$ si $P_i \prec P_j$

$B(i,j) = 0$ si P_i et P_j indifférentes ou si $P_j \prec P_i$.

Au départ, on a B = 0; au fur et à mesure de la détermination des ordres partiels, on les inscrit dans B et on calcule la clôture $\hat{B}$; on recommence en prenant $B = \hat{B}$. Les tests de type Bresenham sont à garder pour la fin, dans la mesure où la prise en compte de la structure globale du caractère permet d'en éviter le plus grand nombre.

Compte tenu de nos hypothèses de départ (coût élevé des tests), la méthode semble optimale; mais elle induit des manipulations combinatoires et logiques très coûteuses en temps. Un compromis paraît souhaitable entre tests supplémentaires et calculs de la clôture de B. Une évaluation devrait être faite afin de déterminer la validité concrète de notre modèle; toutefois cela n'entre pas dans le cadre de cette thèse. Nous nous contenterons de démontrer sa puissance sur quelques exemples pris parmi les caractères d'imprimerie.

IV.- APPLICATION DU MODELE AU GENERATEUR DE CARACTERES.

L'étude que nous achevons a été menée dans un cadre théorique à la fois plus large et plus étroit que celui des caractères d'imprimerie : plus large, puisque nous définissons comme caractère, toute tache simplement connexe assez régulière, et plus étroit en raison des restrictions apportées dès la définition (une seule composante connexe, contour de classe C^∞...). Or, ces hypothèses simplificatrices peuvent être éliminées si l'on réintroduit la nature véritable des caractères romains; par exemple, et comme nous l'avons déjà mentionné, lorsque le bord C d'un caractère comprend plusieurs composantes connexes, une seule n'est pas convexe. L'insertion d'un convexe, c'est à dire d'une association puits-source, ajoute un creux et une bosse qu'il suffit de tester face aux parois non indifférentes.

1.- Première étude : g minuscule Baskerville. (fig. 4-21)

En partant de l'origine 0 dans le sens direct, nous recensons et déterminons les parois suivantes :

c_1:	$\frac{1}{2}$-bosse droite inverse
c_2:	$\frac{1}{2}$-bosse droite directe
c_3:	$\frac{1}{2}$-bosse droite inverse
c_4:	$\frac{1}{2}$-bosse droite directe
d:	puits inverse
e_1:	$\frac{1}{2}$-bosse gauche directe
e_2:	$\frac{1}{2}$-bosse gauche inverse
e_3:	$\frac{1}{2}$-bosee gauche directe
f:	source inverse
g:	$\frac{1}{2}$-bosse droite directe
a:	puits inverse
b:	source directe

Les deux chameaux sont naturellement triés :

$e_3 < e_2 < e_1$ $\qquad$ $c_1 < c_2 < c_3 < c_4$

Dans la forme réduite subsiste une spirale a'-b'-d' et deux lobes f'-g'-a' , d'-f'-a' :

lobes : $f' \prec g' \prec a'$

$f' \prec e' \prec d'$

test de la spirale et de ses raccords :

$b' \prec g' \prec a' \prec e' \prec d'$ (tests sur y_0, y_1, y_2)

Pour respecter une alternance paroi directe-paroi inverse dans la liste classée qui tienne compte de la cohérence du caractère, nous avons obligatoirement la relation :

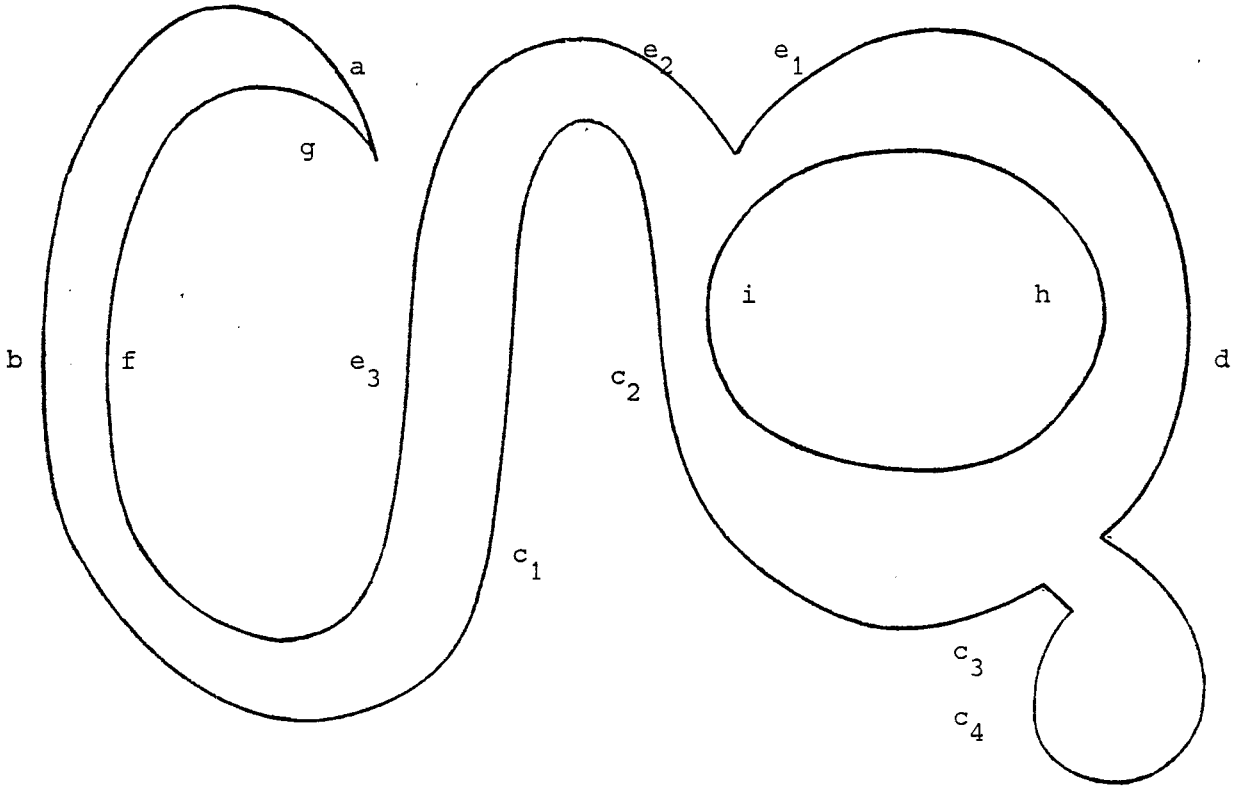
$b' \prec f' \prec g'$

Aucun test de type Bresenham n'est donc nécessaire pour ordonner les parois de la forme réduite.

La réintroduction du chameau e_1-e_2-e_3 appelle au plus un test de Bresenham :

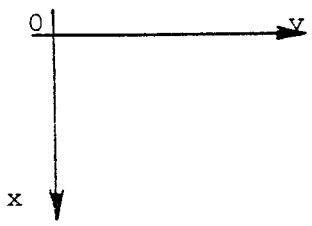
test bosse (g,a) contre creux (f,e_3)

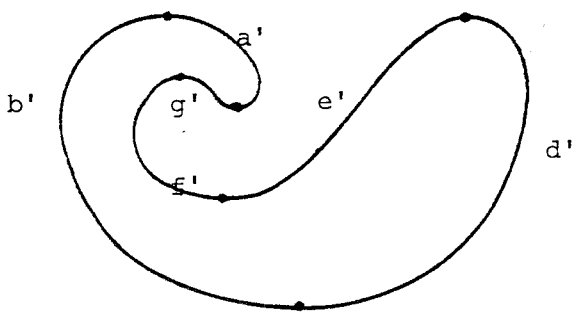
Le résultat partiel est $b \prec f \prec g \prec a \prec e_3 \prec e_2 \prec e_1 \prec d$.



g minuscule Baskerville Figure 4-21

Forme réduite





La réintroduction du chameau c_1-c_2-c_3-c_4 est facilitée par l'indifférence de la bosse (c_3,c_4) vis à vis de tout creux existant. Il reste à effectuer le test de (c_1,c_2) contre (e_3,e_2) dont le résultat est $e_3 < c_1 < c_2 < e_2$, d'où l'ordre total suivant sur la composante connexe principale :

$b < f < g < a < e_3 < c_1 < c_2 < e_2 < e_1 < c_3 < c_4 < d$

Nous prenons en compte le convexe i-h en testant le creux et la bosse contre les autres bosses et creux du caractère. De nombreuses indifférences et restrictions simplifient l'opération; en fin , ne restent à tester que (i-h) contre (c_2,c_3) et (i-h) contre (e_1,d), avec les résultats suivants :

$c_2 < i < h < c_3$ et $e_1 < i < h < d$

Pour terminer nous opérons la fermeture du graphe :

$b < f < g < a < e_3 < c_1 < c_2 < e_2 < e_1 < i < h < c_3 < c_4 < d$

2.- Deuxième étude : E majuscule Baskerville. (fig. 4-22)

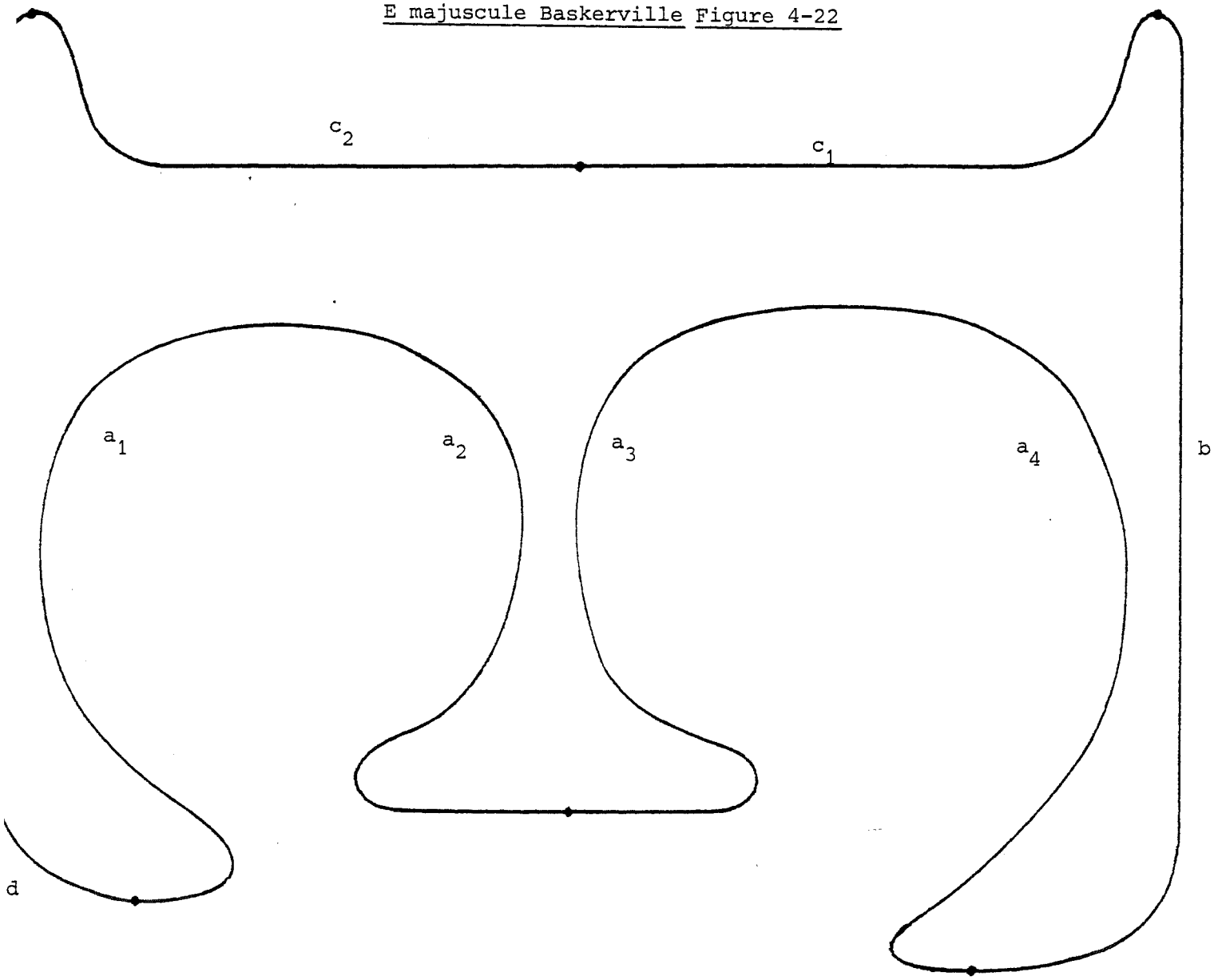
Type des parois :

a_1: $\frac{1}{2}$-bosse droite inverse
a_2: $\frac{1}{2}$-bosse droite directe
a_3: $\frac{1}{2}$-bosse droite inverse
a_4: $\frac{1}{2}$-bosse droite directe
b: puits inverse
c_1: $\frac{1}{2}$-bosse gauche directe
c_2: $\frac{1}{2}$-bosse gauche inverse
d: source directe

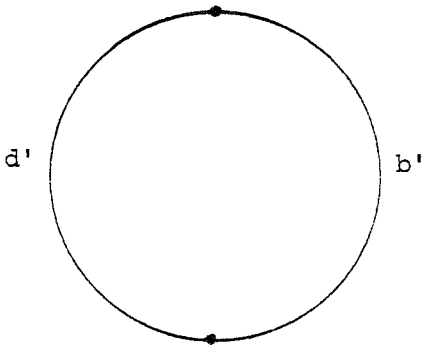
La suppression des chameaux a_1-a_2-a_3-a_4 et c_1-c_2 conduit à la forme fondamentale convexe b'-d'. Il suffit donc de tester chameau contre chameau, ce qui est rendu trivial par leur indifférence mutuelle; aucun test de type Bresenham n'est nécessaire pour aboutir à l'ordre total suivant :

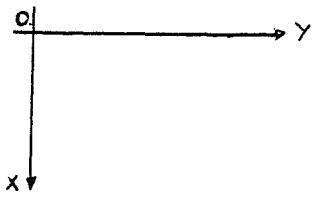
$d < a_1 < a_2 < a_3 < a_4 < c_2 < c_1 < b$.

E majuscule Baskerville Figure 4-22



Forme réduite





V.- ETUDE SOMMAIRE DES ROTATIONS.

1.- Représentation intrinsèque.

Nous avons choisi a priori un point de mort m^+ comme origine de l'abscisse curviligne, et l'introduction de la représentation intrinsèque $\theta(s)$ nous a permis de dégager les premières informations sur la structure d'un caractère :

$\theta'(s) = 0$	point d'inflexion (si $\theta''(s) \neq 0$)
$\theta'(s)$ 0	point d'une paroi tournant vers la gauche
$\theta'(s)$ 0	point d'une paroi tournant vers la droite
$\theta(s) = k\pi$	extrema locaux en x (morts et naissances)

De plus, le théorème de la tangente tournante introduit une pseudo-périodicité le long d'une courbe simple fermée, puisque nous avons :

$\forall\ s \in \mathbf{R}\ ,\ \theta(L+s) = \theta(s) + 2\pi$

A partir de ces données, nous avons déterminé la nature des parois d'un caractère, et démontré quelques unes de leurs propriétés élémentaires.

Or, une simple translation de l'origine des abscisses curvilignes modifie l'ensemble des parois, puisqu'elle correspond à un changement d'orientation du caractère par rapport aux axes de balayage : si le graphe $\theta(s)$ est connu à partir d'une origine s_0, une translation de longueur s_1 correspond à une rotation du caractère (ou des axes), d'un angle ($\theta(s_1) - \theta(s_0)$).

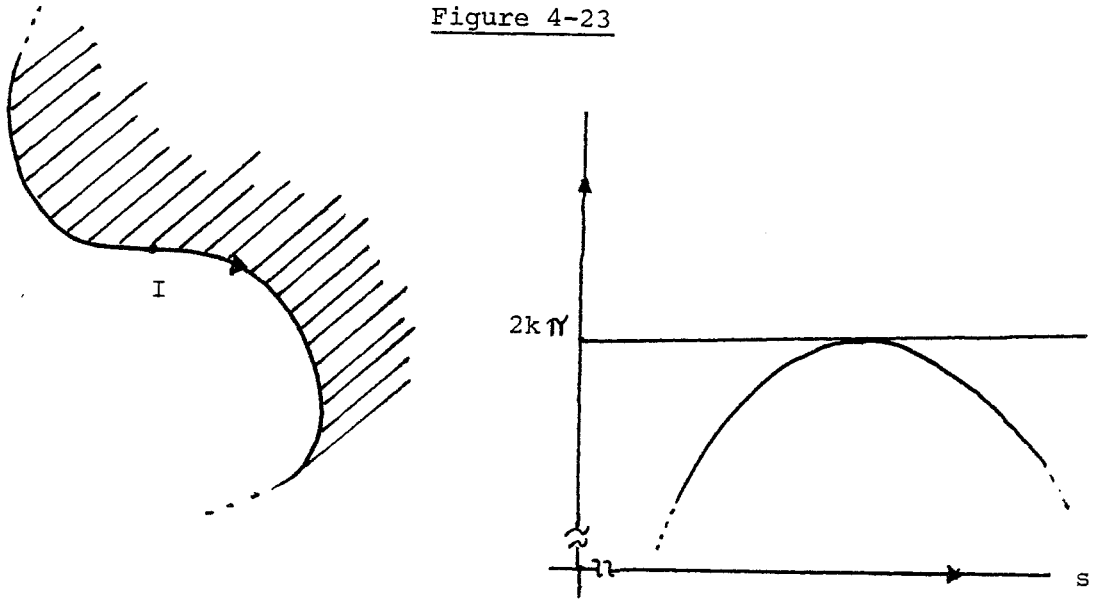
2.- Mort et naissance des extrema locaux.

Supposons qu'il existe un point d'inflexion sur ∂C (sinon ∂C est convexe). Prenons alors l'orientation qui rend horizontale la tangente d'inflexion en ce point. (fig. 4-23-1). Une petite rotation $\Delta\theta$ dans l'orientation de la figure correspond à une variation incrémentale de l'origine des s.

Si $\Delta\theta$ est négatif, rien n'est modifié (fig. 4-23-2); par contre, si $\Delta\theta$ est positif, il y a apparition d'une nouvelle paroi par création d'un point de naissance n^- et d'un point de mort m^+ (fig. 4-23-3).

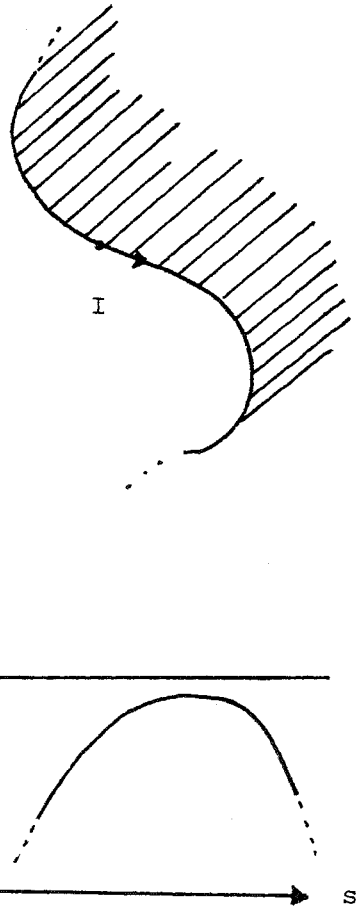
Nous pouvons imager ce phénomène en disant que les points d'inflexion sont des

Figure 4-23

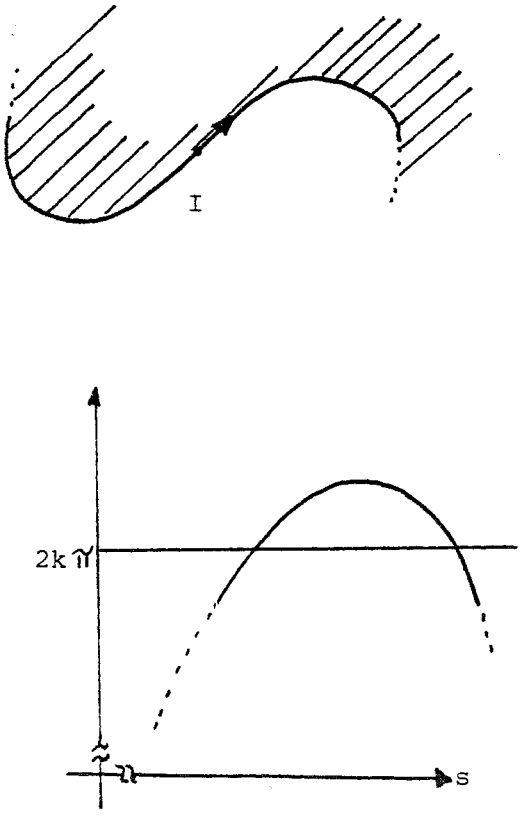


(1)

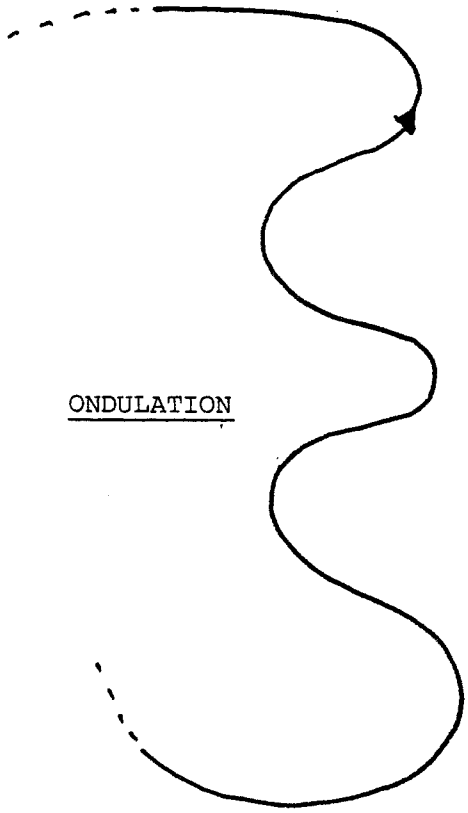
(2)



(3)

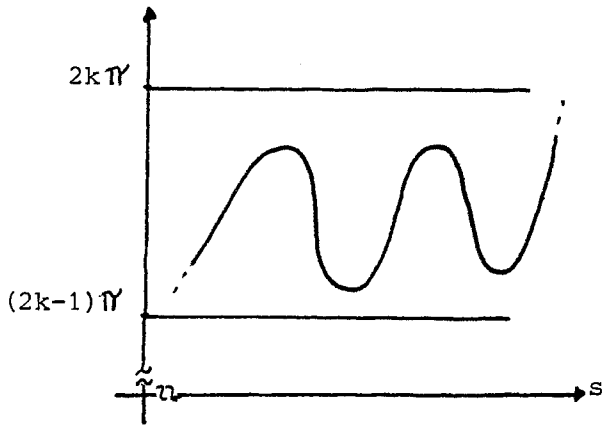


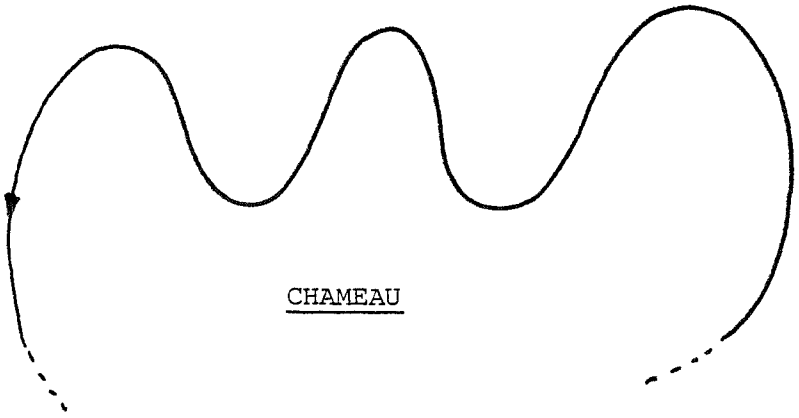
points de mort ou de naissance d'un couple de points de mort et de naissance adjacents. Nous expliquons ainsi que des ondulations sur un paroi se transforment en chameau par changement d'orientation (fig. 4-24).



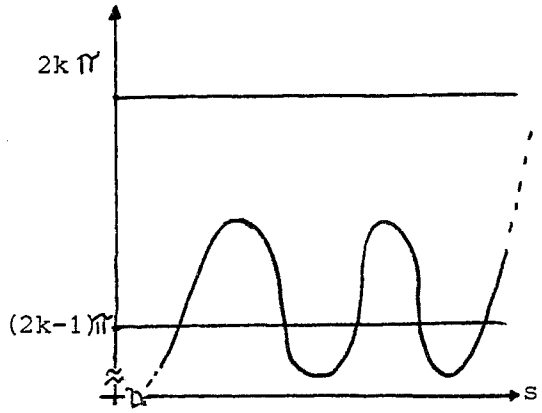
ONDULATION

Figure 4-24





CHAMEAU



Chapitre 5

CHAPITRE 5.

GESTION DES PROCESSUS EN CAS DE REPLIEMENT.

1.- ORGANISATION GLOBALE DU LOGICIEL DE SUIVI DE CONTOUR.

A - Contraintes.

B - Archictecture logicielle.

C - Contextes.

2. - GESTION DES PROCESSUS.

A - Problème.

B - Solutions.

1.- ORGANISATION GLOBALE DU LOGICIEL DE SUIVI DE CONTOUR.

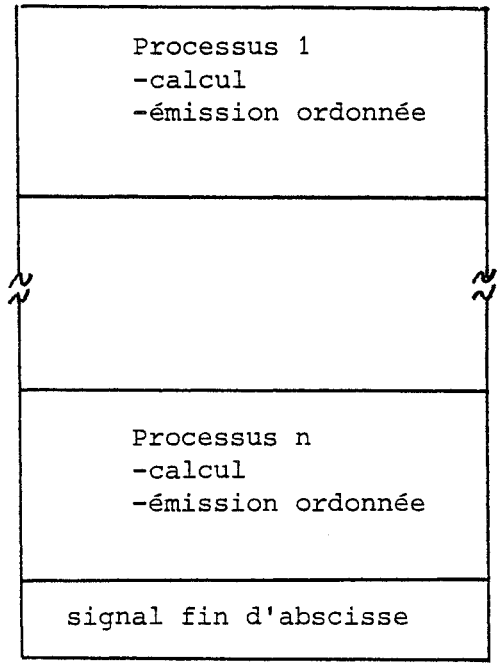
A - Contraintes.

Sur chaque processeur de suivi de contour, le logiciel de l'Etape 3 gère le calcul d'un nombre variable de parois, compris entre 0 et 10. La philosophie générale qui a guidé son écriture découle de la réalisation de la contrainte de vitesse - 1000 caractères de taille 100 X 100 par seconde - : il faut optimiser les programmes pour des configurations haut-de-gamme de la machine, lorqu'un processeur de suivi de contour exécute au plus une paroi. On peut encore dire que tant que le nombre des PSC est suffisant, et cela arrive aussi sur des machines bas de gamme pour des caractères relativement simples, la complexité du logiciel de repliement doit être transparente. Il est nécessaire d'autre part de privilégier les traitements les plus fréquents (calcul incrémental d'un point sur une paroi), au détriment peut-être de la prise en compte des autres événements (changement d'arc, naissance et mort des parois), et de déporter certaines fonctions au niveau du matériel (synchronisation avec les autres processeurs PSC et la carte de sortie CS).

B - Architecture logicielle.

Une paroi est exécutée sur un PSC par un processus : la naissance d'une paroi prise en charge par le processeur provoque l'activation d'un processus, et sa mort le libère. En fonctionnement normal, c'est à dire lorsqu'à l'abscisse courante il n'y a ni mort ni naissance, on exécute un module de gestion, module "zéroprocessus" lorqu'il n'y a pas de parois allouées, "monoprocessus" s'il y en a une et "multiprocessus" lorsque, par repliement, le processeur calcule n parois . ($1 \leq n \leq 10$). (figure 5-1)

Pour éviter de nombreux tests inutiles, la détection des abscisses de naissance a été déportée au niveau du matériel : lors d'une naissance, un décompteur est initialisé à la distance séparant l'abscisse courante de la prochaine abscisse de naissance. A la fin d'une itération du module de gestion actif, un signal est émis vers la carte de sortie pour indiquer le passage à l'abscisse suivante ; il est aussi



$0 \leq n \leq 10$

Figure 5-1 :
Module de gestion

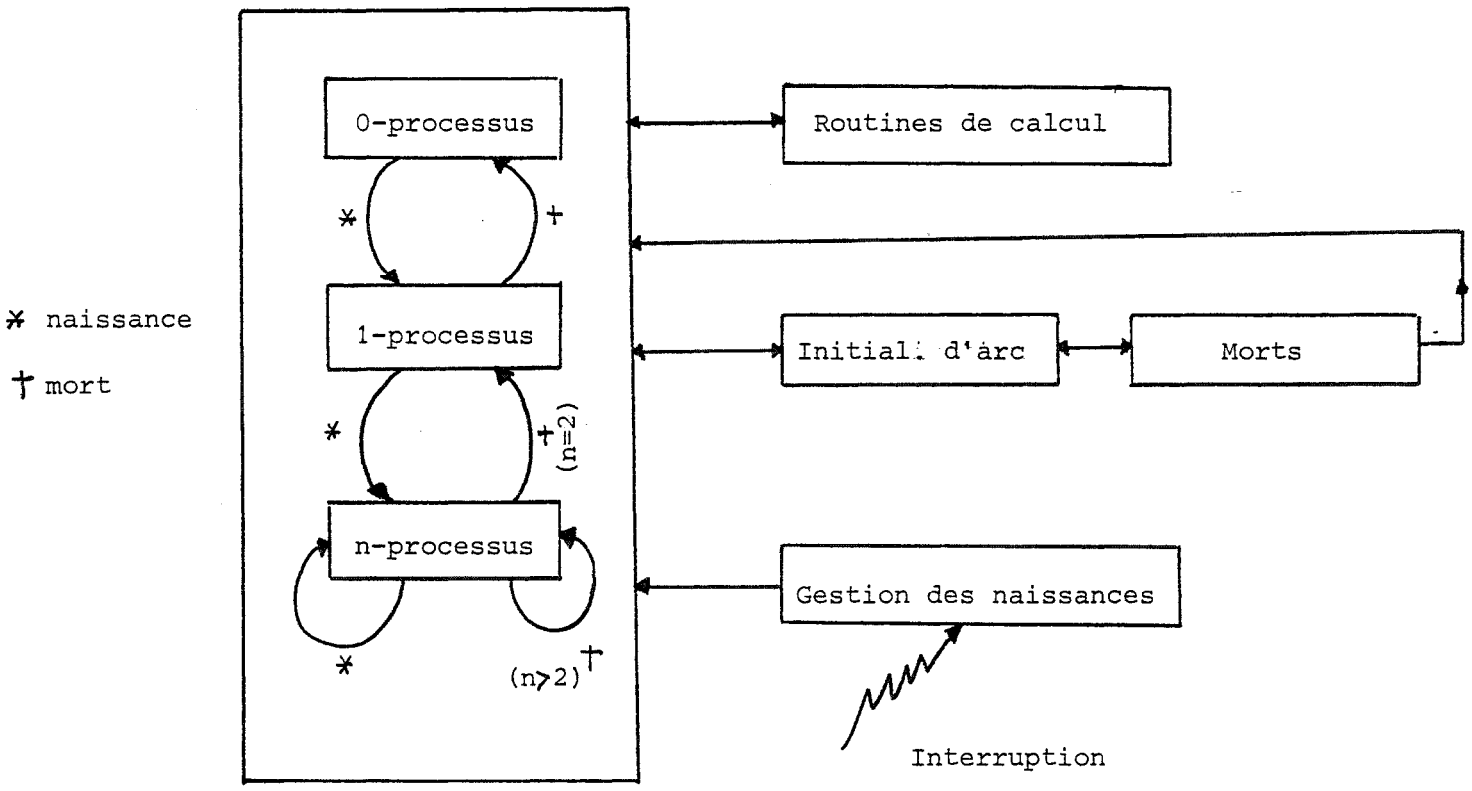


Figure 5-2
Organisation du logiciel.

utilisé comme horloge de décrémentation du compteur, dont le passage à zéro provoque une interruption et l'exécution du programme de gestion des naissances. Ce dernier examine la structure d'arbre et peut, en fonction des numéros des parois à activer, créer de nouveaux processus et provoquer une transition d'un module de gestion vers un autre. Lorsque cela est nécessaire (naissance et fin d'arc), on active un programme d'initialisation d'arc, qui est aussi capable de déterminer les abscisses de mort et de les répercuter au niveau des modules de gestion.
Pour terminer, signalons qu'un processus actif appelle à chaque abscisse une routine de calcul qui détermine l'ordonnée courante de la paroi correspondante. (figure 5-2).

C - Contextes.

Chaque processus définit un contexte que l'on peut diviser en deux sous-ensembles :

- le contexte de la paroi contient le numéro d'ordre, le pointeur dans l'anneau de description du contour, et le sens d'acquisition des paramètres.
- le contexte de l'arc courant est formé des variables nécessaires au calcul incrémental d'un point d'un arc de cercle ou d'un segment de droite (ordonnée courante, valeurs de la fonction implicite et des deux dérivées partielles, longueur restant à calculer). Les éléments du contexte de l'arc sont modifiés à chaque abscisse (une exception peut être faite pour les dérivées partielles qui restent constantes pour un segment de droite).

Dans un module monoprocessus, le contexte de la paroi active a pu être conservé dans les registres du processeur ; ce n'est d'ailleurs que dans ces conditions que nous observons un temps de 10 μs par itération. Pour éviter un test sur la nature de l'arc à tracer, le programme d'initialisation d'arc redéfinit dans le module l'argument de l'appel à la routine de calcul ; cela implique que l'exécution du module se fasse en mémoire vive.

Lorsque PSC gère plusieurs processus, il devient impossible de sauvegarder tous les contextes dans les registres ; il faut donc les stocker en mémoire. Afin de gagner du temps, nous avons stocké les variables du contexte à la place d'opérandes immédiats d'instruction de chargement de registre. L'exécution d'un processus d'un module n-processus ($n > 2$), se passe, du moins à une abscisse normale, en trois temps :

. chargement des registres par exécution des instructions à opérandes immédiats

. appel de la routine de calcul

. sauvegarde du contexte dans les instructions de changement.

Un tel processus se modifie à chaque itération ; son exécution doit donc se faire en RAM. C'est pourquoi, à l'initialisation de la machine, l'ensemble des modules de gestion est transféré de ROM à RAM, où ils seront exécutés.

2.-GESTION DES PROCESSUS.

A - Problème.

Il faut maintenant supposer l'existence d'un module de gestion unique, de manière à masquer les subtilités logicielles destinées au respect de la contrainte de vitesse ; il y a alors homogénéité de tous les processus, quelque soit leur nombre. A des moments non connus a priori, un processus doit être activé, pour une durée déterminée, qui correspond à l'exécution de la paroi prise en charge. On peut considérer les processus comme similaires et de même priorité.

Une itération du module de gestion, c'est à dire une itération de chacun des processus qui le constitue, doit être effectuée à chaque abscisse. Il faut donc résoudre un problème d'allocation d'un processeur réel à des processus pour leur exécution. Une stratégie de type tourniquet (round-robin CRO 75) doit être envisagée, de façon à respecter l'égalité de traitement des processus.

Dans les systèmes d'exploitation d'ordinateur temps réel, un processus spécial, réveillé par une interruption matérielle au bout d'un quantum de temps déterminé, s'occupe de gérer le mécanisme d'allocation du processeur aux autres

processus (utilisateurs ou système). Le critère de fin d'allocation d'un processeur à un processus échappe donc au processus lui-même. C'est là que réside la différence principale avec notre système : un processus rend la main lorsqu'il a terminé un quantum de traitement -c'est à dire le calcul d'une ordonnée- ; il dispose lui-même du critère de fin d'allocation et provoque lui-même son passage à l'état d'attente. Il faut encore signaler que tous les processus que nous activons sont de priorité identique.

B - Solutions.

Dans la première solution que nous avons étudiée, nous utilisions pour chaque contexte, une zone de mémoire vive associée à une table d'occupation ou d'activation des contextes. Un programme réentrant s'exécutait à chaque abscisse sur les contextes correspondant aux parois actives. Nous étions obligés de gérer de manière efficace la table des contextes et la table d'occupation lors des activations et des libérations de processus. En cas de naissance, où placer le contexte qui venait de s'initialiser ? En cas de mort, que faire du trou dans la table d'occupation et du contexte correspondant ? D'autre part, la notion même de programme réentrant impliquait une séparation du traitement et du contexte, que nous voulions fortement imbriqués pour gagner du temps. Une première remarque que l'on peut faire est qu'il n'est pas nécessaire d'exécuter les processus dans un ordre précis, pourvu que chacun d'eux soit exécuté au cours de chaque cycle : de toute façon, la carte de sortie triera les résultats en fonction des numéros d'ordre des parois. Dans ces conditions, l'élimination des trous dans la table d'occupation est immédiate, lorsqu'une paroi meurt, qui n'est pas la dernière du module de gestion, on remplace son contexte par le dernier contexte, et on l'active (si c'est la dernière, il suffit d'émettre le signal de fin d'abscisse et de libérer le contexte). On a alors un procédé qui utilise une représentation continue des processus actifs.
Comme la gestion d'une table d'occupation, quelle qu'en soit la forme, était contradictoire avec les contraintes de vitesse, nous avons dû rechercher une solution plus simple qui tienne compte de l'ensemble de ces remarques ; elle est caractérisée par les points ci-dessous :

* Plutôt que d'exécuter un programme réentrant sur n contextes consécutifs, nous exécutons n programmes qui incluent le contexte dans leurs instructions.
* Il y a continuité en mémoire des programmes des processus actifs ; à la fin de chaque programme, une zone d'instruction dynamique permet soit d'entrer dans l'exécution du processus suivant, soit de revenir à l'exécution du premier processus, pour l'itération suivante après avoir émis un signal de fin d'abscisse (figure 5-3).
* A l'initialisation, 10 programmes-processus consécutifs sont chargés de ROM en RAM, et aucun contexte activé. 10 est le nombre maximum de parois allouées à un même PSC ; il ne peut se rencontrer que sur une machine bas de gamme qui comprend un seul processeur de suivi de contour.
* A l'activation d'un processus, on affecte son contexte au premier programme libre, et on met à jour les instructions de branchement ; ce traitement est effectué par le programme de gestion des naissances (figure 5-4). En cas de mort, il y a retassage immédiat si nécessaire par simple transfert du contexte et mise à jour des instructions de branchement (figure 5-5 et 5-6).

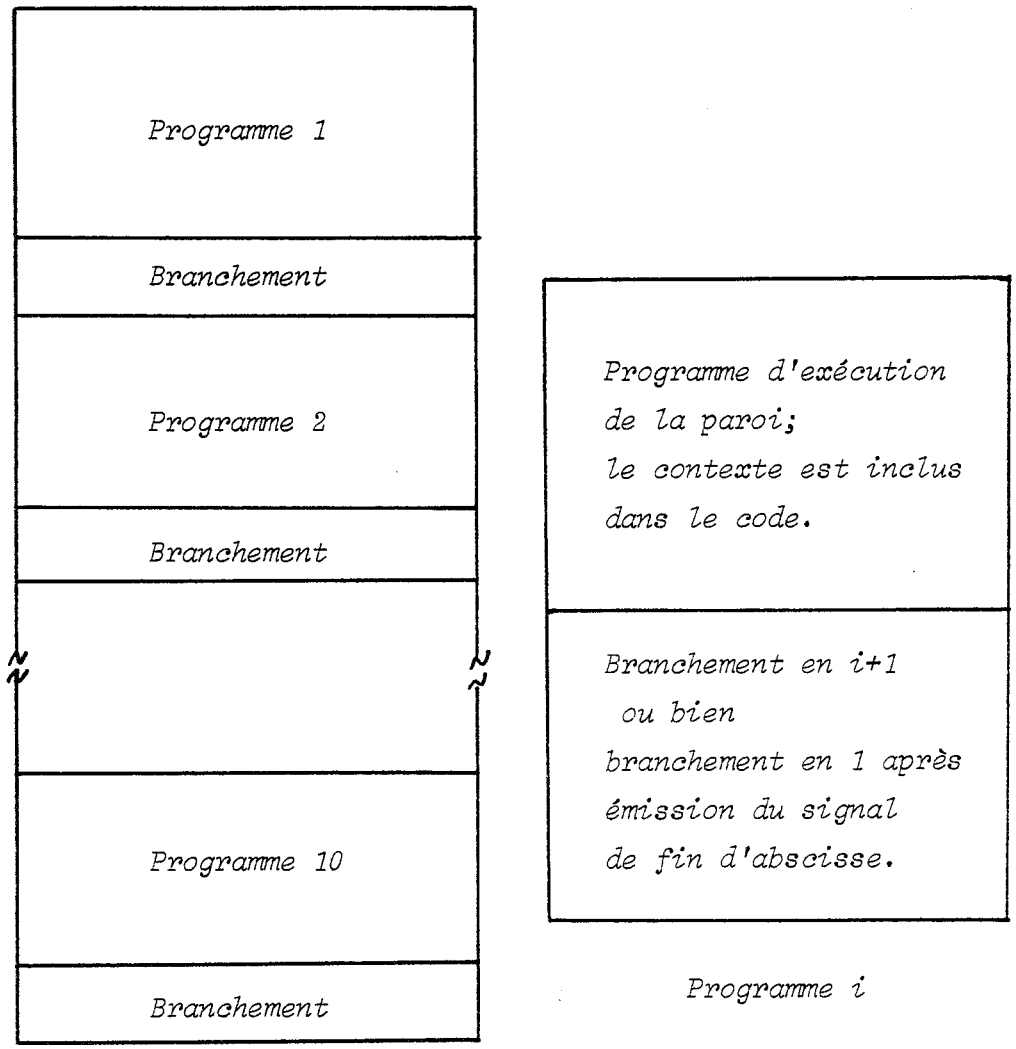


Figure 5-3 : *Gestion de l'exécution des processus actifs.*

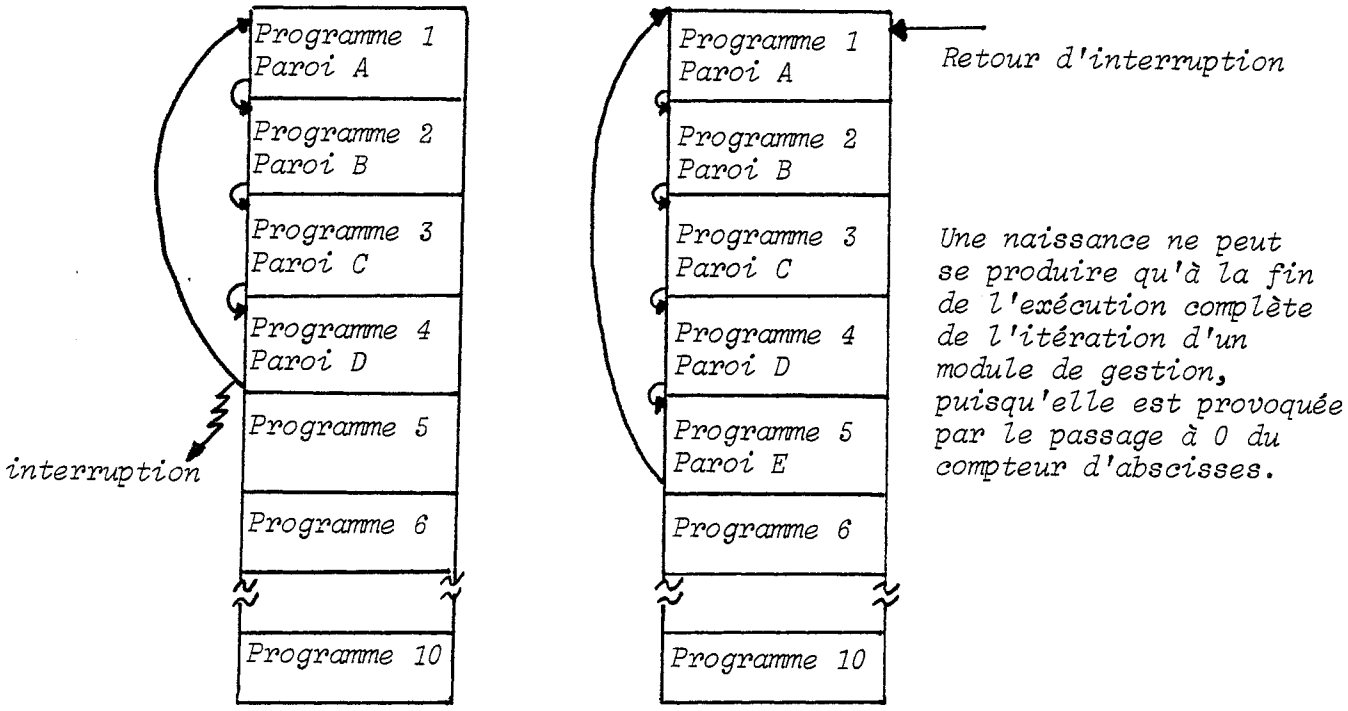


Figure 5-4 : *Naissance d'une paroi.*

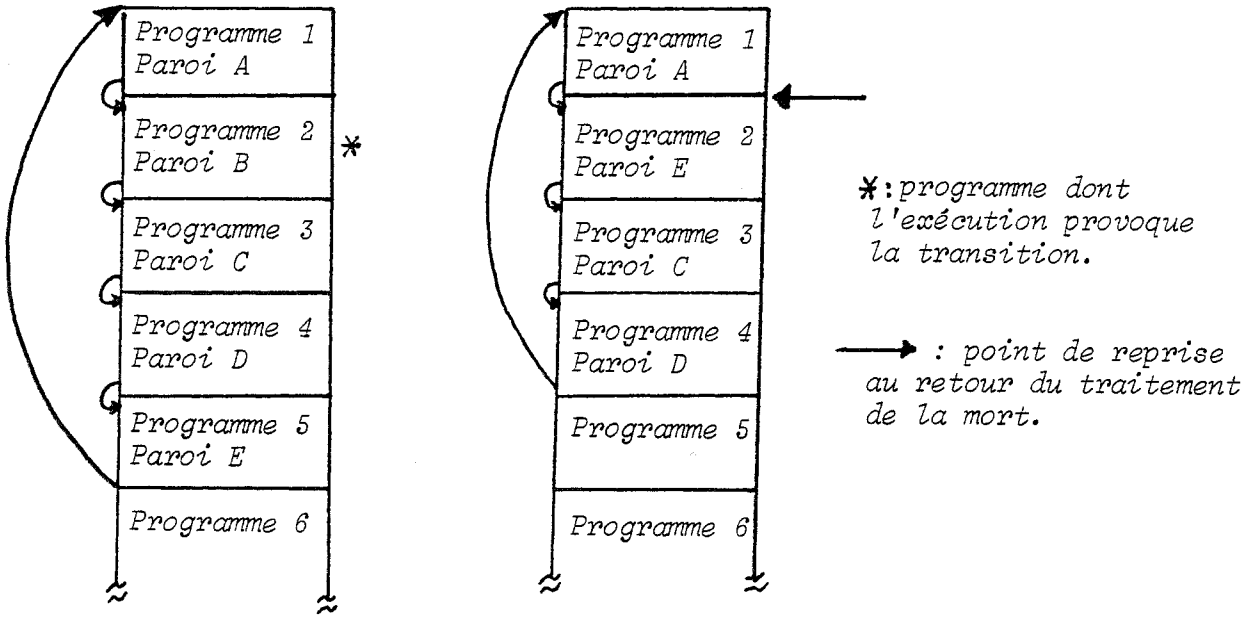


Figure 5-5 : *Mort d'une paroi. (cas 1)*

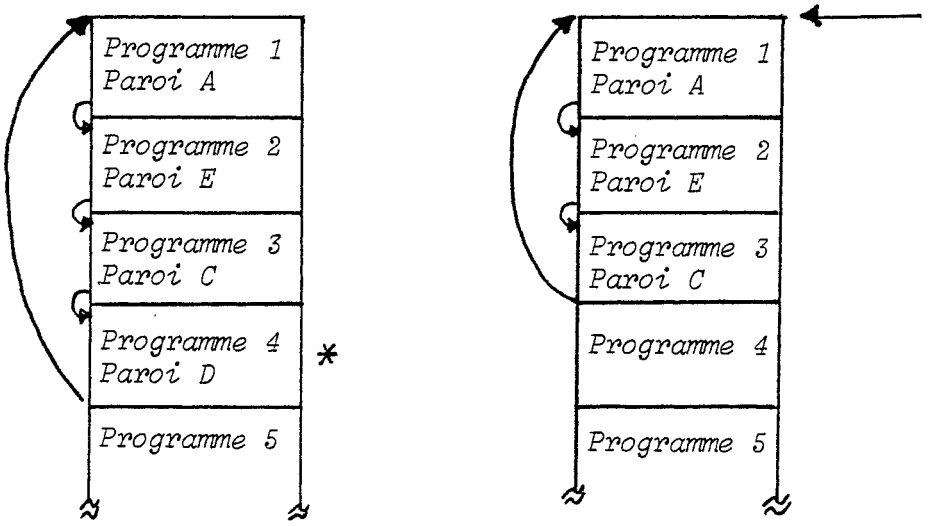


Figure 5-6 : Mort d'une paroi. (cas 2)

Chapitre 6

CHAPITRE 6.

ANALYSE DES PERFORMANCES

A - Présentation.

B - Paramètres du modèle.

C - Caractères tests.

D - Modèles linéaires.

E - Influence des facteurs d'échelle.

F - Application du modèle.

G - Conclusion.

A - Présentation.

La 3° étape constitue un ensemble matériel et logiciel complexe sur lequel il est paradoxalement assez facile de faire des mesures, puisqu'un seul paramètre en mesure l'efficacité : le temps de génération d'un caractère. Comme il est impensable de tester exhaustivement l'ensemble des caractères engendrables (les fabricants de photocomposeuses en proposent plusieurs dizaines de milliers), nous avons dû déterminer un groupe restreint de caractères spéciaux dont la simplicité structurelle nous a permis d'élaborer un modèle décrivant leur temps de génération à partir de leurs caractéristiques élémentaires.

Pour élargir ces résultats partiels, nous avons été amenés à formuler quelques hypothèses statistiques sur des ensembles de caractères réels (police romaine de corps de texte) ; cela nous a permis de calculer une estimation de la vitesse moyenne de fonctionnement qui, comme le prévoyait le cahier des charges, dépasse 1000 caractères simples de taille 100x100 par seconde.

C'est autour de cette double démarche -modélisation, puis application du modèle - que va être structuré ce chapitre ; nous y trouverons successivement :

- l'énumération des paramètres et la description des caractères de test ;
- l'approximation du comportement par un modèle linéaire ;
- l'étude de l'influence des facteurs d'échelle ;
- l'application du modèle aux polices d'imprimerie
- une discussion sur les modifications à apporter au codage des caractères en fonction des différentes utilisations du générateur.

B - Les paramètres du modèle.

On peut décomposer l'ensemble des paramètres en deux classes : ceux qui dépendent du caractère, les paramètres structurels, et les facteurs d'échelle en x (X) et en y (Y).

Paramètres structurels

D : nombre de segments de droite dans la description séquentielle du contour.

C : nombre d'arcs de cercle dans la description du contour ; si E est le nombre de courbes évoluées C=4E (du moins à l'échelle 1).

B : nombre de points de naissance. Bien qu'en toute rigueur il eût fallu distinguer les abscisses de naissance, et pour chacune d'elles, les points de naissance correspondants, nous n'avons gardé que ce dernier paramètre.

M : nombre moyens de parois actives.

Facteurs d'échelle

X : comme dans les machines qui utilisent directement du code par plage, le nombre d'incréments X dans la direction de balayage va jouer un rôle prépondérant. On relie facilement X à l'échelle E_X et à la chasse du caractère X_{Max} (2000 points pour le m ou le w ; environ 400 pour le i ou le l) : $X = E_X.X_{Max}$

Y : la définition du facteur d'échelle Y est plus délicate ; on ne peut plus trouver d'analogie directe avec les codes par plage. C'est l'algorithme de calcul incrémental des points du contour qui est en cause : la détermination de l'ordonnée d'un point d'un arc où la tangente à une pente forte demande plus d'itérations de la boucle de calcul intérieure que celle d'un point où la pente de

la tangente est faible. A X=x donné, c'est donc l'accroissement maximal en y sur l'ensemble des parois activées qui influe sur le temps de calcul (du moins dans l'hypothèse d'une machine non repliée).

C - Les caractères de test

Nous avons supposé que les paramètres que nous venons de définir étaient indépendants, ce qui nous a permis de décrire des caractères spéciaux où il était facile de mettre en évidence l'influence de l'un deux seulement. C'est ainsi que nous avons utilisés :

le disque (M=2, C=8, D=0, B=1, X_{Max}=2000, Y_{Max}=1000)

le O (M=4, C=16, D=0, B=2, X_{MAx}=2000, Y_{Max}=2000)

le caractère TN1 (M=2, C=0, D=2, B=1, X_{Max}=2000, Y_{Max}=0)

le caractère TN2 (M=2, C=0, D=4, B=2, X_{Max}=2000, Y_{Max}=0)

les caractères TV_i (M=2i, C=0, D=2i, B=i, X_{Max}=2000, Y_{Max}=0) i=1,2,3,4,5

TN1 et TN2 ont permis de déterminer la durée de traitement d'un point de naissance et d'une initialisation de droite.

Les caractères TV_i ont été utilisés pour étudier l'influence du repliement.

Le disque et le O ont permis de déterminer la durée de l'initialisation d'un arc de cercle.

En modifiant E_X et E_Y , nous avons pu calculer la durée moyenne de calcul du point courant sur un segment de droite et un arc de cercle.

D - Modèles linéaires

A une échelle telle que E_x=E_y, nous avons :

$$T=f(B,C,D,M,X).$$

La fonction f est affine en X avec une corrélation proche de 1 sur l'ensemble des caractères testés, et quelque soit la configuration de la machine, c'est à dire le nombre N de processeurs de suivi de contour :

$$T=aX + g(B,C,D,M)$$

g peut être considéré comme le "coût fixe" du caractère, aX représentant un "coût variable" directement proportionnel à la taille de la grille de génération.

Nous nous sommes rendus compte que Y intervenait très peu, dans la mesure où le facteur Ey/Ex devait rester compris entre $\frac{1}{2}$ et 2 : des tests sur le disque ont indiqué une dérive inférieure à 5% du temps ; de toutes façons le modèle que nous élaborons ne donnera pas une estimation fiable à 95%.C'est pourquoi nous considérons l'influence de Y comme négligeable. Dans ces conditions, les mesures effectuées nous ont conduit aux deux modèles linéaires suivants :

. $T_{\mu s}$=105B + 1/M(32D + 82C) + 18X

sur une machine à cycle de base de 250 ns, si l'on est assuré de ne jamais provoquer de repliement (c'est à dire lorsque N=10 pour des polices romaines de corps de texte.)

. $T_{\mu s}$= 250B + 1/N (32D + 82C) + M/N 38X

sur une machine à cycle de base de 250 ns où N<10.

Nous allons maintenant comparer les deux estimations et expliquer leurs différences :

temps de gestion d'un point de naissance :

Lorsqu'on est sûr que tout au long du caractère, on aura N≥M (et cela n'est vrai à priori que si N=10), on peut utiliser un programme spécial de gestion de naissance, puisqu'en une abscisse de naissance, un PSC est dans un des deux états suivants :

. il a déjà une paroi en charge, il ne peut donc être concerné par une paroi activée,

. il est libre, et peut être concerné par une paroi au plus.

Par contre, dans une machine susceptible de repliement, il faut examiner l'ensemble des points de naissance.

temps d'initialisation d'un arc

La grosse différence entre le temps d'initialisation d'un segment de droite (32 μs) et d'un arc de cercle (82 μs)vient essentiellement du temps très réduit de l'initialisation des droites horizontales (18 μs) intégrées au cas général (50 μs) ; d'autre part, l'initialisation des dérivées partielles est plus complexe pour un arc de cercle.

Dans le cas d'une machine non repliée (N=10), nous avons supposé que les arcs à initialiser étaient répartis uniformément entre les différentes parois (d'où le facteur 1/M) ; pour une machine repliée, nous pouvons supposer que

les arcs à initialiser seront répartis uniformément entre les différents processeurs (d'où le facteur 1/N).

Estimation du paramètre a (temps du calcul d'une abscisse):

Dans le cas d'une machine non repliée, à chaque abscisse le temps de calcul est équivalent au temps de calcul d'un point par un algorithme incrémental : $18\mu s$, qui correspond à une moyenne entre $10\mu s$ (droite horizontale) et $24\mu s$ (moyenne sur un arc représentant le quart d'un cercle). Le contexte de la paroi est mémorisé dans les registres internes d'un processeur de suivi de contour, qui se trouve soit en monoprocessus, soit en zéroprocessus (cf chapitre V).

Dans le cas d'une machine repliée, le temps de calcul incrémental est augmenté du temps de chargement puis de sauvegarde des registres, que l'on peut estimer à $20\mu s$. Il faut bien sûr pondérer ce nombre ($38\mu s$) par le nombre moyen de parois repliées sur chaque processeur de suivi de contour (M/N).

E - Influence de la mise à l'échelle

Nous avons supposé dans un premier temps que les paramètres structurels B, C et D ne dépendaient pas de X ; cette hypothèse grossière nous a permis de dégager le modèle que nous venons d'exposer. En fait C et D dépendent fortement des facteurs de mise à l'échelle Ex et Ey (donc de X et de Y) ; cela est dû à la superposition des deux phénomènes suivants :

- Tous les arcs de cercle d'accroissement inférieur à trois unités de grille après réduction (aussi bien en x qu'en y) sont transformés en segments de droite.
- Tout arc d'accroissement en x nul ou égal à 1 n'est pas initialisé (c'est ainsi que l'on considère que les droites verticales ont un temps de calcul nul).

On pourrait penser que ces transformations sont très rares ; il n'en est rien. Ainsi tout arc de cercle d'accroissement inférieur à 60 unités de grille sur le maître est transformé en segment de droite à l'échelle 100X100. Or sur une courbe évoluée d'accroissement L, on peut considérer que deux cercles générés ont un accrois-

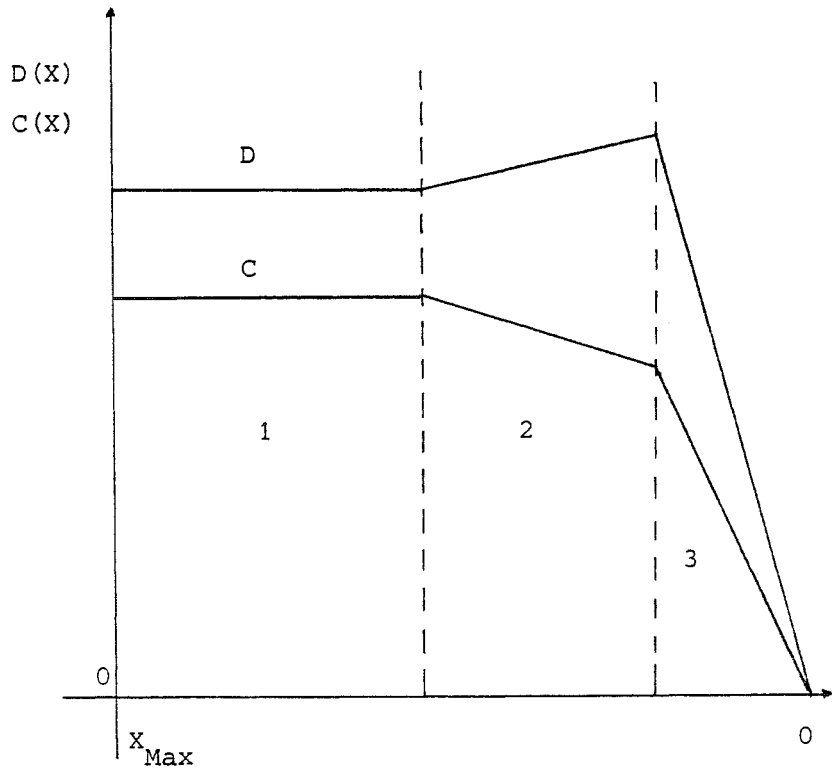
sement supérieur à L/4, et deux inférieurs à L/4. Donc sur une courbe évoluée d'accroissement 240 unités de grille sur le maître, il est probable qu'au moins deux cercles induits seront remplacés par des segments de droite. Pour la plupart des caractères, on a $X_{Max} \leq 1000$: ainsi sur toute courbe évaluée dont la longueur est inférieure au quart de la longueur totale du caractère, certains arcs de cercle seront transformés en segments de droite.

B ne dépend de X que pour de très petites échelles (quelques unités de grille).

Restreignons nous maintenant à l'étude du modèle représentant une machine non repliée, et supposons l'influence du paramètre Y mineure. Dans ces conditions, nous pouvons écrire :

$$T_{\mu s} = 105B + 1/M(32D(X) + 82\ C(X)) + 18X$$

Il est bien sûr impossible d'obtenir une représentation analytique des fonctions D(X) et C(X) ; nous indiquons une approximation graphique résumant la discussion ci-dessus : Dans la zone 1, D et C sont approximativement constants ; lorqu'on réduit l'échelle (zone 2), un certain nombre de cercles se transforment en droite ou disparaissent (la pente de D(X) est un peu plus faible que la pente de C(X)) ; et pour terminer, droites et cercles se fondent en un seul point. Le point de mesure correspondant à une grille de taille 100 X 100 se trouve dans la zone 3 en général.



F - Application du modèle

Pour passer de la description d'un modèle paramétré par des caractéristiques éléméntaires à son application à des ensembles d'objets plus complexes, il est nécessaire de formuler des hypothèses statistiques autorisant sa généralisation : cette démarche est habituelle dès que l'on désire relier le local au global (cf extensions des modèles microéconomiques au domaine macroéconomique) ; il faut grouper les objets qui ont à peu près le même comportement vis à vis du modèle en classes homogènes que l'on représentera par des paramètres moyens.
Pour les caractères réels, cette classification est immédiate parce que sous-jacente à la notion même de police : une police d'imprimerie est un jeu de caractères homogènes par rapport aux paramètres du modèle que nous avons déterminé.
Nous pouvons définir deux classes limites :

police simple : pas d'embellissements ; caractères bâtons. En moyenne, C=12, B=2, D=10, M=2,5 (type Univers).

police complexe : caractères ronds, embellissements nombreux : C=80, B=4, D=20, M=3,5 (type Baskerville).

Les calculs montrent alors que pour X_0=50, chasse moyenne correspondant à une taille de grille de l'ordre de 100 X 100, et en prenant $C(X_0)$= 6 (resp 40) ; $D(X_0)$ = 8 (resp 16), on obtient les temps de génération moyens suivants (sans repliement).

1,4 ms pour une police simple

2,4 ms pour une police complexe

cela avec une horloge de base de 4 MHz.

G - CONCLUSION

L'application du modèle nous montre que l'étape 3 respecte le cahier des charges imposé à l'ensemble du générateur de caractère. Pour une police de complexité moyenne, nous obtenons une vitesse de fonctionnement de l'ordre de 500 caractères définis dans une grille 100 X 100 par seconde, en utilisant une horloge à 4 Mhz, ce qui permet de prévoir un débit de 1000 caractères par seconde

si l'on utilise la version rapide du microprocessuer 8086. Cela ne doit pas nous empêcher de formuler quelques remarques :

Pour des échelles petites, l'influence du "coût fixe" g(B,M,C(X),D(X)) est prépondérante ($10 \leq X \leq 50$). Or la génération de caractères de faible définition correspond typiquement à des applications du générateur de caractères au traitement de texte et à la bureautique. On pourrait alors concevoir des police spéciales qui minimiseraient et le nombre de points de naissance, et la complexité du contour de manière à assurer un débit moyen suffisant, même en cas de repliement total (N=1 ou bien N=2). Toutefois l'intérêt d'une régénération des caractères en temps réel est faible, eu égard à la possibilité de décoder à faible vitesse la police entière dans une mémoire tampon intermédiaire. Pour des échelles moyennes, coût fixe et coût variable s'équilibrent sensiblement, on se rend encore compte de l'influence du coût fixe, puisque si l'on double l'échelle en X, passant de X=50 à X=100, ou même de X=100 à X=200, le débit est divisé par un facteur inférieur à 2 (de l'ordre de 1,7).

Ainsi, il aurait été plus intéressant de calculer le débit moyen sur des caractères de taille 200 X 200, puisque l'on passe de 1000 car/s (100X100) à 600 car/s (200X200).

Pour de grandes échelles, l'influence du coût variable masque complètement le coût fixe, et le modèle est parfaitement linéaire. Il peut alors être intéressant de minimiser le temps de calcul d'un point, par exemple en microprogrammant ou en câblant l'algorithme incrémental utilisé : ce phénomène serait encore plus sensible sur des machines fortement repliées.

Bibliographie

COU 73 P. COUEIGNOUX
"Compression of type faces by contour coding", MS thesis, Dep. Elec. Eng., Massachussetts Institute of Technology, USA, Jan 1973, 60 pages.

COU 75 P. COUEIGNOUX
"Generation of roman printed fonts", Ph.D, Dep. Elec. Eng, Massachusetts Institute of Technology, Juin 1975, 139 pages.

HOU 78 M. HOURDEQUIN
"Generation de polices d'imprimerie pour photocomposeuse digitale",doct. ing. ENS des Mines de Saint-Etienne, France, Nov. 1978, 150 pages.

HOC 79 M. HOURDEQUIN et P. COUEIGNOUX
"Specifying arbitrary planar smooth curves far fast drawing", Eurographics, 1979.

COG 80 P. COUEIGNOUX et R. GUEDJ
"Computer generation of colored planar patterns on TV-like rasters", Proceedings of the IEEE, vol 68, N° 7, Juillet 1980,pages 909 à 922.

COU - P. COUEIGNOUX
"Character generation by computer", CGIP, à paraître

BLO 79 M. BLOCH
"Architecture parallèle de génération du contour d'une tache", DEA, ENS des Mines de Saint-Etienne, Octobre 1979, 54 pages.

SIC 80 C. SICO
"Approximation par arcs de cercle du contour d'une tache", DEA ENS des Mines de Saint-Etienne, Novembre 1980, 50 pages.

KNU 78 D.E. KNUTH
"Mathematical typography", Stanford University Comp. Science Dpt, Stan-CS-78-648, Février 1978, 68 pages.

HOR 77 B.K. HORN
"Circle Generator for display devices", CGIP, vol 6, 1977, pages 196-198

BRE 77 J.E. BRESENHAM
"A linear algorithm for incremental digital display of circular arcs", Comm. of the ACM,vol 20 N° 2, pages 100-106, Février 1977.

BEG 72 M. BERGER et B. GOSTIAUX
"Géométrie différentielle", Armand Colin, pages 300-372, 1972

CRO 75 CROCUS
"Systèmes d'exploitation des ordinateurs", DUNOD, pages 158-162, 1975

dernière page de la thèse

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974,

VU les rapports de présentation de Messieurs

- P. COUEIGNOUX
- M. LUCAS

Monsieur B L O C H Marc

est autorisé à présenter une thèse en soutenance pour l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité " Systèmes et réseaux d'Informatiques".

Fait à Saint-Etienne, le
Fait à Grenoble, le 19 juin 1981

Le Président de l'I.N.P.G.

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

Le Directeur de l'EMSE,

M. MERMET
INGÉNIEUR EN CHEF DES MINES
DIRECTEUR
ECOLE NATIONALE SUPÉRIEURE DES
MINES DE SAINT-ÉTIENNE