Étude algébrique et programmation de la discrétisation de figures planes

Jean-Marie Miermont

N° d'ordre :

THESE

présentée à

L'UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

pour obtenir

LE GRADE DE DOCTEUR DE TROISIEME CYCLE

INFORMATIQUE

par

J.M. MIERMONT

ETUDE ALGEBRIQUE ET PROGRAMMATION DE LA DISCRETISATION DE FIGURES PLANES

Thèse soutenue le 16 Juin 1971 devant la Commission d'Examen :

Monsieur	J. KUNTZMANN	Président
Messieurs	N. GASTINEL	Examinateurs
	C. BENZAKEN	

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

LISTE DES PROFESSEURS

Président : Monsieur Michel SOUTIF
Vice-Président : Monsieur Gabriel CAU

PROFESSEURS TITULAIRES

MM.	ANGLES D'AURIAC Paul	Mécanique des fluides
	ARNAUD Georges	Clinique des maladies infectieuses
	ARNAUD Paul	Chimie
	AYANT Yves	Physique approfondie
	BARBIER Jean-Claude	Physique expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRIE Joseph	Clinique chirurgicale
	BENOIT Jean	Radioélectricité
	BESSON Jean	Electrochimie
	BEZES Henri	Chirurgie générale
	BLAMBERT Maurice	Mathématiques pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Pathologie médicale
	BONNIER Etienne	Electrochimie Electrométallurgie
	BOUCHERLE André	Chimie et Toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BRAVARD Yves	Géographie
	BRISSONNEAU Pierre	Physique du Solide
	BUYLE-BODIN Maurice	Electronique
	CABANAC Jean	Pathologie chirurgicale
	CABANEL Guy	Clinique rhumatologique et hydrologique
	CALAS François	Anatomie
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et Toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques pures
	CHATEAU Robert	Thérapeutique
	CHENE Marcel	Chimie papetière
	COEUR André	Pharmacie chimique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie Pathologique
	CRAYA Antoine	Mécanique
Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de Dermatologie et Syphiligraphie

	FAU René	Clinique neuro-psychiatrique
	FELICI Noël	Electrostatique
	GAGNAIRE Didier	Chimie physique
	GALLISSOT François	Mathématiques pures
	GALVANI Octave	Mathématiques pures
	GASTINEL Noël	Analyse numérique
	GERBER Robert	Mathématiques pures
	GIRAUD Pierre	Géologie
	KLEIN Joseph	Mathématiques pures
Mme	KOFLER Lucie	Botanique et physiologie végétale
MM.	KOSZUL Jean-Louis	Mathématiques pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
	LATURAZE Jean	Biochimie pharmaceutique
	LEDRU Jean	Clinique médicale B
	LLIBOUTRY Louis	Géophysique
	LOUP Jean	Géographie
Mle	LUTZ Elisabeth	Mathématiques pures
MM.	MALGRANGE Bernard	Mathématiques pures
	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Séméiologie médicale
	MASSEPORT Jean	Géographie
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	NEEL Louis	Physique du Solide
	OZENDA Paul	Botanique
	PAUTHENET René	Electrotechnique
	PAYAN Jean-Jacques	Mathématiques pures
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET René	Servomécanismes
	PILLET Emile	Physique industrielle
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REULOS René	Physique industrielle
	RINALDI Renaud	Physique
	ROGET Jean	Clinique de pédiatrie et de puériculture
	SANTON Lucien	Mécanique
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SILBERT Robert	Mécanique des fluides
	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale
	VAILLAND François	Zoologie
	VAUQUOIS Bernard	Calcul électronique
Mme	VERAIN Alice	Pharmacie galénique
	VERAIN André	Physique
Mme	VEYRET Germaine	Géographie
MM.	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale
	YOCCOZ Jean	Physique nucléaire théorique

PROFESSEURS ASSOCIES

MM.	BULLEMER Bernhard	Physique
	RADHAKRISHNA Pidatala	Thermodynamique

PROFESSEURS SANS CHAIRE

MM.	AUBERT Guy	Physique
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARRA Jean	Mathématiques appliquées
	BEAUDOING André	Pédiatrie
	BERTRANDIAS Jean-Paul	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BONNETAIN Lucien	Chimie minérale
Mme	BONNIER Jane	Chimie générale
MM.	CARLIER Georges	Biologie végétale
	COHEN Joseph	Electrotechnique
	COUMES André	Radioélectricité
	DEPASSEL Roger	Mécanique des Fluides
	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Métallurgie
	DOLIQUE Jean-Michel	Physique des plasmas
	GAUTHIER Yves	Sciences biologiques
	GEINDRE Michel	Electroradiologie
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	HACQUES Gérard	Calcul numérique
	JANIN Bernard	Géographie
Mme	KAHANE Josette	Physique
MM.	LATREILLE René	Chirurgie générale
	LAURENT Pierre	Mathématiques appliquées
	MULLER Jean-Michel	Thérapeutique
	PERRIAUX Jean-Jacques	Géologie et minéralogie
	POULOUJADOFF Michel	Electrotechnique
	REBECQ Jacques	Biologie (CUS)
	REVOL Michel	Urologie
	REYMOND Jean-Charles	Chirurgie générale
	ROBERT André	Chimie papetière
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIBILLE Robert	Construction Mécanique
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale
M.	VALENTIN Jacques	Physique nucléaire

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

Mle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBLARD Pierre	Dermatologie
	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Yves	Chimie

BEGUIN Claude	Chimie organique
BELORIZKY Elie	Physique
BENZAKEN Claude	Mathématiques appliquées
Mme BERTRANDIAS Françoise	Mathématiques pures
MM. BLIMAN Samuel	Electronique (EIE)
BLOCH Daniel	Electrotechnique
Mme BOUCHE Liane	Mathématiques (CUS)
MM. BOUCHET Yves	Anatomie
BOUSSARD Jean-Claude	Mathématiques appliquées
BOUVARD Maurice	Mécanique des Fluides
BRIERE Georges	Physique expérimentale
BRODEAU François	Mathématiques (IUT B)
BRUGEL Lucien	Energétique
BUISSON Roger	Physique
BUTEL Jean	Orthopédie
CHAMBAZ Edmond	Biochimie médicale
CHAMPETIER Jean	Anatomie et organogénèse
CHARACHON Robert	Oto-Rhino-Laryngologie
CHIAVERINA Jean	Biologie appliquée (EFP)
CHIBON Pierre	Biologie animale
COHEN-ADDAD Jean-Pierre	Spectrométrie physique
COLOMB Maurice	Biochimie médicale
CONTE René	Physique
CROUZET Guy	Radiologie
DURAND Francis	Métallurgie
DUSSAUD René	Mathématiques (CUS)
Mme ETERRADOSSI Jacqueline	Physiologie
MM. FAURE Jacques	Médecine légale
GAVEND Michel	Pharmacologie
GENSAC Pierre	Botanique
GERMAIN Jean-Pierre	Mécanique
GIDON Maurice	Géologie
GRIFFITHS Michael	Mathématiques appliquées
GROULADE Joseph	Biochimie médicale
HOLLARD Daniel	Hématologie
HUGONOT Robert	Hygiène et médecine préventive
IDELMAN Simon	Physiologie animale
IVANES Marcel	Electricité
JALBERT Pierre	Histologie
JOLY Jean-René	Mathématiques pures
JOUBERT Jean-Claude	Physique du Solide
JULLIEN Pierre	Mathématiques pures
KAHANE André	Physique générale
KUHN Gérard	Physique
Mme LAJZEROWICZ Jeannine	Physique
MM. LAJZEROWICZ Joseph	Physique
LANCIA Roland	Physique atomique
LE JUNTER Noël	Electronique
LEROY Philippe	Mathématiques
LOISEAUX Jean-Marie	Physique nucléaire
LONGEQUEUE Jean-Pierre	Physique nucléaire
LUU DUC Cuong	Chimie organique
MACHE Régis	Physiologie végétale
MAGNIN Robert	Hygiène et Médecine préventive
MARECHAL Jean	Mécanique
MARTIN-BOUYER Michel	Chimie (CUS)
MAYNARD Roger	Physique du Solide
MICOUD Max	Maladies infectieuses
MOREAU René	Hydraulique (INP)

	NEGRE Robert	Mécanique
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B)
	PEFFEN René	Métallurgie
	PELMONT Jean	Physiologie animale
	PERRET Jean	Neurologie
	PERRIN Louis	Pathologie expérimentale
	PFISTER Jean-Claude	Physique du Solide
	PHELIP Xavier	Rhumatologie
Mle	PIERY Yvette	Biologie animale
	RACHAIL Michel	Médecine interne
	RACINET Claude	Gynécologie et obstétrique
	RICHARD Lucien	Botanique
Mme	RINAUDO Marguerite	Chimie macromoléculaire
MM.	ROMIER Guy	Mathématiques (IUT B)
	ROUGEMONT (DE) Jacques	Neuro-chirurgie
	STIEGLITZ Paul	Anesthésiologie
	STOEBNER Pierre	Anatomie pathologique
	VAN CUTSEM Bernard	Mathématiques appliquées
	VEILLON Gérard	Mathématiques appliquées (INP)
	VIALON Pierre	Géologie
	VOOG Robert	Médecine interne
	VROUSSOS Constantin	Radiologie
	ZADWORNY François	Electronique

MAITRES DE CONFERENCES ASSOCIES

MM.	BOUDOURIS Georges	Radioélectricité
	CHEEKE John	Thermodynamique
	GOLDSCHMIDT Hubert	Mathématiques
	YACOUD Mahmoud	Médecine légale

CHARGES DE FONCTIONS DE MAITRES DE CONFERENCES

Mme	BERIEL Hélène	Physiologie
Mme	RENAUDET Jacqueline	Microbiologie

A ma femme

Je voudrais dire toute ma reconnaissance à Monsieur le Professeur KUNTZMANN qui dirige le service de Mathématiques Appliquées de l'Université de Grenoble. Ses encouragements et ses précieux conseils m'ont permis d'entreprendre et de poursuivre cette étude.

Je suis très sensible à l'honneur qu'il me fait en acceptant de présider le Jury.

Je remercie Monsieur le Professeur GASTINEL d'avoir bien voulu rendre cette étude plus intéressante en apportant ses idées de réalisation et en m'accordant des facilités de travail.

Je remercie Monsieur BENZAKEN, Maître de Conférences, qui a accepté de diriger ce travail.

Je tiens à remercier ceux qui m'ont aidé par leur expérience en programmation, en particulier, MM. BELLOT, DU MASLE, LECARME et SIRET.

Je ne voudrais pas non plus oublier dans mes remerciements, Mme DOREL et M. MOUNET pour le soin qu'ils ont apporté à la réalisation matérielle de cette thèse.

TABLE DES MATIERES

IIème PARTIE - ETUDE DE DOMAINES

§ - I - POSITION DU PROBLEME

I-1 Exemples de problèmes aux limites

I-2 La méthode des différences finies.

§ - II - DEFINITIONS

II-1 Domaines admissibles

II-2 Grille. Résolution. Bloc

II-3 Carte d'un bloc

§ - III - DISCRETISATION DE DOMAINES

III-1 Description de l'intérieur d'un domaine

III-1-a - Liste d'intérieur

III-1-b - Liste auxiliaire d'intérieur.
Algorithme ALG.LO.

III-1-c - Recherche des composantes connexes. Algorithme ALG.L1

III-1-d - Algorithme ALG.L2.

III-2 Changement d'alphabet

III-3 Changement de résolution.

§ - IV - PROBLEMES D'IMPLEMENTATION

IIIème PARTIE - PROGRAMMATION

§ - I - TERMINAL GRAPHIQUE

I-a Introduction

I-b Le terminal graphique TG.

I-b-1 - Description physique et logique

I-b-2 - Le jeu d'instructions

I-b-3 - Le point de vue du programmeur

§ - II - PROGRAMMES

II-a Mémorisation d'un tracé linéaire par morceaux

II-a-1 - Génération de la grille

II-a-2 - Génération du tracé linéaire par morceaux.

II-b Mémorisation d'un tracé discrétisé

II-b-1 - Génération de la figure test

II-b-2 - Génération et affichage du tracé

II-b-3 - Modification de la figure test.

II-c Commandes de modification des figures

II-d Exemples.

INTRODUCTION

Dans l'étude des représentations des configurations géométriques [Fre], Freeman fait les constatations simples suivantes :

- si l'on met bout à bout deux lignes polygonales orientées, on obtient une loi de composition interne sur l'ensemble de ces lignes qui est associative et possède un élément neutre (les mathématiciens appellent une telle structure un monoïde).

- si de plus, on ne considère que les lignes polygonales orientées construites avec un nombre fini de vecteurs auxquels on affecte des lettres, on peut associer à chaque ligne un mot formé par les lettres des vecteurs qui la constituent. Il est alors possible de travailler dans l'ensemble des mots composés avec ces lettres, ensemble qui constitue un monoïde libre.

Cette thèse est consacrée à l'étude et à la programmation de problèmes de nature géométrique pour lesquels l'utilisation des monoïdes définis ci-dessus est un outil commode.

1ère PARTIE

Pour un utilisateur de calculatrice digitale, l'ensemble des nombres réels est un ensemble discret. Ceci est dû à la représentation (en virgule fixe ou en virgule flottante) des nombres réels. Il s'ensuit que dans le plan, les points que le programmeur peut distinguer par programme sont situés sur une grille. Ceci nous conduit à l'étude d'une géométrie discrète.

Dans cette partie, nous étudions un certain nombre de propriétés géométriques relatives aux courbes planes, en utilisant la structure de monoïde indiquée plus haut. Le choix d'une structure moins riche que celle d'espace vectoriel est motivé par le fait que dans un monoïde libre nous disposons d'une classification des algorithmes. Celle-ci nous permet de préciser le 'degré de difficulté' du problème considéré.

D'autre part, pour un choix convenable des alphabets, nous sommes dans la situation évoquée auparavant. Les points du plan que l'on considère sont situés sur une grille.

2ème PARTIE

Certains types de discrétisation de courbes ou de domaines du plan que l'on rencontre en analyse numérique consistent à utiliser des grilles formées de mailles régulières (carré, rectangle, triangle, hexagone). Il en va ainsi dans les problèmes aux limites lorsqu'on discrétise un domaine du plan où l'on cherche à déterminer une fonction inconnue. Dans la réalisation d'un système automatique de résolution de problèmes aux limites dans le plan, il est nécessaire de coder le domaine d'étude considéré avant de commencer le traitement des équations aux dérivées partielles. Nous présentons dans cette deuxième partie des algorithmes qui permettent d'associer à un domaine du plan une représentation en mémoire suffisamment souple pour être utilisable en cas de modification d'une partie du domaine ou de changement dans le choix de la méthode de résolution du problème.

3ème PARTIE

Nous examinons dans cette troisième partie la programmation proprement dite. Nous avons bénéficié en cela de la qualité du matériel et de la puissance de la configuration des machines qui se trouvent au Laboratoire de Mathématiques Appliquées de Grenoble. En particulier, la console de visualisation (display 2250) qui est reliée au système 360 possède les caractéristiques d'un petit calculateur. Si cette console est couramment utilisée pour générer à l'écran un programme graphique déjà défini par le programmeur, elle est beaucoup moins employée pour enregistrer le programme graphique correspondant à une figure définie manuellement par un opérateur. C'est cet aspect de l'activité de ce périphérique qui nous a intéressé. On voit en particulier que pour les lignes polygonales orientées on peut définir une correspondance très simple entre l'abscisse curviligne d'un point sur cette ligne et l'adresse de l'instruction qui va générer ce point. Ceci est à la base des programmes de poursuite de crayon optique que nous présentons.

Nous décrivons ensuite les commandes de modification de figure qui peuvent servir dans la réalisation effective du système dont il est question à la deuxième partie. Dans une telle réalisation, les facilités d'emploi des consoles et le gain très appréciable de temps qu'elles procurent justifient leur utilisation malgré leur coût d'achat élevé.

Pour la rédaction, nous avons eu le souci de montrer comment on pouvait passer des situations géométriques réelles à des situations algébriques. Cette présentation, si elle se fait parfois au détriment d'une plus grande rigueur, a pour nous l'avantage d'être conforme à la façon dont les problèmes nous ont été posés.

Ière PARTIE

CHAPITRE I

ALPHABETS ET LANGAGES

I.1 RAPPELS DE DEFINITION.

Etant donné un ensemble fini V que nous appellerons commodément alphabet, on notera par V^* l'ensemble des suites finies d'éléments de V. De telles suites seront appelées mots (ou chaînes).

La concaténation de deux mots x et y avec

$$x = x_1 \dots x_n \qquad x_i \in V$$

$$y = y_1 \dots y_p \qquad y_j \in V$$

est le mot $z = xy = x_1 \dots x_n y_1 \dots y_p$.

Cette loi est interne, associative et possède un élément neutre, noté Λ, qui est le mot vide. V^* est alors un monoïde (pour cette loi) dit monoïde libre engendré par V.

On appelle langage sur V tout sous-ensemble de V^*.

Il est important de pouvoir définir ou décrire un langage déterminé même s'il est infini.

On dispose de deux modes de définition.

- définition par un procédé génératif. Ce procédé permet de construire de façon systématique les mots qui appartiennent au langage. Le modèle mathématique de grammaire développé par Noam Chomsky en 1956 fournit de telles définitions.
- définition par un procédé de reconnaissance. On se donne un algorithme qui, après examen du mot considéré, délivre une réponse positive si le mot est élément du langage, négative dans le cas contraire. De tels algorithmes sont en particulier réalisés par les automates.

I.2 GRAMMAIRES.

I.2.a Définition des grammaires de Chomsky.

Une grammaire G est un quadruplet (V_N, V_T, P, S) où

- V_N est le vocabulaire non terminal (ou auxiliaire)
- V_T est le vocabulaire terminal disjoint du précédent
- P une relation binaire finie sur $V^+ \times V^*$

avec $V = V_N \cup V_T$.

$V^+ = V^* - \{\wedge\}$.

Les couples qui satisfont à la relation P sont appelés productions.

Une production est notée $\alpha \rightarrow \beta$.

- S est un symbole distingué de V_N appelé axiome.

A partir de la relation P, on définit la relation binaire $\overline{P}$ notée $\longmapsto$ par :

$x \longmapsto y$ si et seulement si il existe deux chaînes u et v de V^* et une production $\alpha \rightarrow \beta$ telles que

$x = u\,\alpha\,v$

$y = u\,\beta\,v$.

(y dérive simplement de x).

On considère alors la fermeture transitive $\overline{P}^{\infty}$ de $\overline{P}$, notée $\longmapsto\!\!\!\!\!=$.

$x \longmapsto\!\!\!\!\!= y$ si et seulement si il existe un nombre fini de chaînes de V^* $x_0, x_1, \ldots, x_n$ telles que

$x_0 = x \quad x_n = y \quad$ et $\quad x_i \longmapsto x_{i+1}$ pour $i = 1, 2, \ldots, n-1$.

(y dérive de x dans G).

Par définition, le langage généré par G noté L(G) est l'ensemble des chaînes terminales qui dérivent de l'axiome.

$$L(G) = \{ z \in V_T^* \ ; \ S \Longmapsto z \}.$$

La classification des langages est obtenue en particularisant les productions des grammaires associées.

. s'il n'y a aucune restriction sur P autre que celles de la définition, la grammaire est de type 0.

. si pour toute production de P $\alpha \rightarrow \beta$ on a :

$$\alpha = \alpha_1 \, A \, \alpha_2 \text{ et } \beta = \alpha_1 \, \beta' \, \alpha_2$$

avec α_1, α_2, β' dans V^* $\beta' \neq \Lambda$ et A dans V_N la grammaire est de type 1 ou à contexte lié.

. si pour toute production de P $\alpha \rightarrow \beta$ on a :

$$\alpha = A \quad \text{avec A dans } V_N$$
$$\beta \neq \Lambda \quad \text{avec } \beta \text{ dans } V^*$$

la grammaire est de type 2 ou à contexte libre. (De telles grammaires servent à la construction des langages de programmation usuels).

. si toutes les productions de P sont de la forme

$$\text{ou} \quad \begin{cases} A \rightarrow a \, B \\ A \rightarrow a \end{cases} \quad \text{avec A, B dans } V_N \text{ et a dans } V_T$$

la grammaire est de type 3 ou régulière.

Un langage généré par une grammaire de type 0, 1, 2, 3 est dit de type 0 (ou récursif), de type 1 (ou à contexte lié), de type 2 (ou à contexte libre), de type 3 (ou régulier, de Kleene, d'états finis). Les inclusions des différentes classes de langages résultent immédiatement des définitions.

Kleene et Brzozowski ont montré que la classe des langages réguliers coïncide avec celle des expressions régulières. Celles-ci sont formées à partir des symboles de l'alphabet V pris comme mots de longueur 1 au moyen de règles qui utilisent la concaténation, l'union, l'opération étoile (si U est un sous ensemble de V^*, U^* est l'ensemble formé par les produits d'un nombre quelconque d'éléments de U). Une sous-classe particulière de langages réguliers est constituée par les K-langages locaux. Ces langages simples jouent un rôle important dans notre étude.

Un K-langage local L est la donnée de

- $A \subseteq V$, A est l'ensemble des lettres initiales
- $B \subseteq V$, B est l'ensemble des lettres terminales
- $I \subseteq V \times V$, I est l'ensemble des doublets interdits.

Une chaîne x appartient à L si et seulement si :

- elle commence par une lettre de A
- elle se termine par une lettre de B
- et deux lettres consécutives de x ne forment jamais un doublet interdit.

On constate que L peut s'écrire en notation ensembliste

$$L = AV^*B \cap (V^* - V^*IV^*).$$

La classe des langages réguliers étant stable pour les opérations de produit, passage au complémentaire, intersection, réunion, substitution, il est clair que tout K-langage local est régulier.

I.3 AUTOMATES.

I.3.a Définition.

Nous ne donnons pas ici les définitions mathématiques des différents types d'automates. Le lecteur les trouvera par exemple en [Hop] où sont aussi exposés les résultats récents qui concernent la théorie des langages formels et des automates.

Un automate est constitué

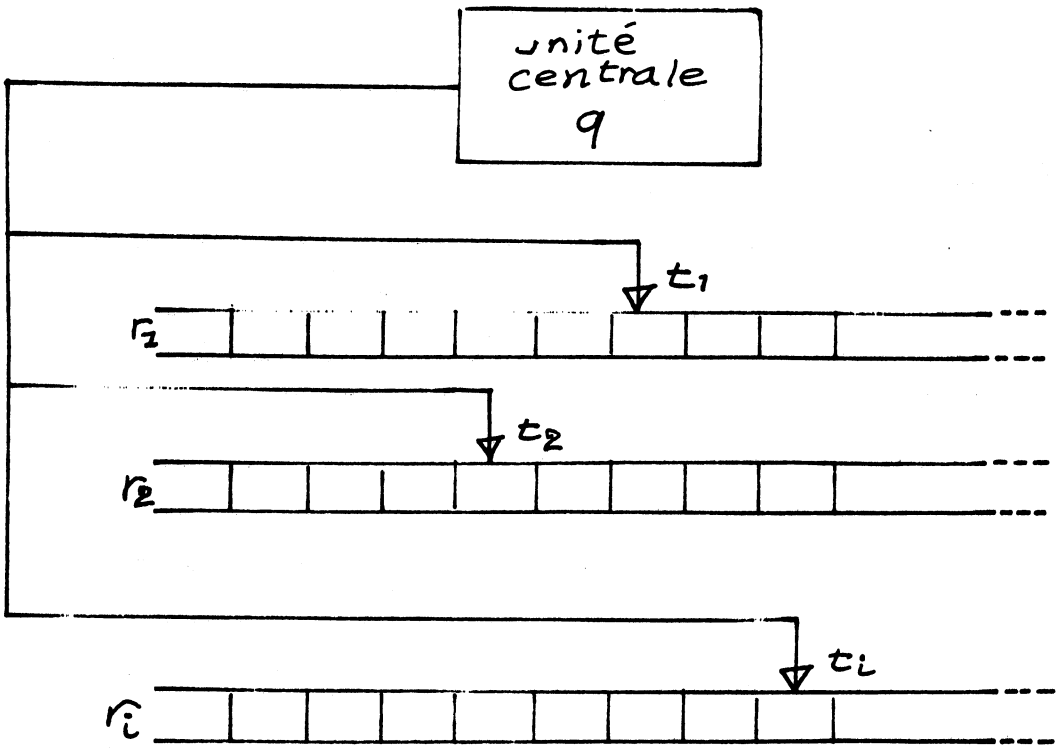
- d'une unité centrale caractérisée par son état q. L'ensemble Q des états est fini et possède un sous ensemble particulier formé des états finaux.
- d'un ou plusieurs rubans r_i (supports d'information organisés en cellules).
- de têtes t_i qui permettent à l'unité centrale de lire ou d'écrire sur les rubans.

Pour décrire le fonctionnement de l'automate, nous supposons que le temps est découpé en instants

$$\tau_0 \ \tau_1 \ \cdots\cdots \ \tau_n \ \cdots\cdots\cdots$$

A un instant τ_n on définit la <u>situation</u> de l'automate. Cette situation est caractérisée par

- son état q, le positionnement de ses têtes t_i.
- les informations existant sur les rubans, en particulier celles qui se trouvent sous les têtes.



Entre les instants τ_n et τ_{n+1} on définit <u>l'action</u> de l'automate. Cette action est déterminée (de façon unique si l'automate est déterministe) par la situation à l'instant τ_n.

Cette action peut

- être nulle
- consister à déplacer des têtes
- consister à déplacer des têtes après modification du contenu de la cellule en regard de ces têtes
- consister à modifier les cellules sans déplacement des têtes
- maintenir ou changer l'état de l'unité centrale.

On dit que l'automate s'arrête à l'instant τ_m

. lorsque l'action de τ_m a τ_{m+1} est nulle

. lorsqu'il y a conservation de l'état aux instants ultérieurs.

Le fonctionnement de l'automate est alors caractérisé par une relation entre l'ensemble des situations possibles (ensemble fini) et l'ensemble des actions permises pour cet automate.

Le langage L(A) accepté par un automate A est tel que, après examen d'une chaîne de ce langage (et de ce langage seulement) transcrite sur un des rubans de A, l'automate se trouve dans un état final lorsqu'il s'arrête.

I.3.b Classification des Automates.

La classification des automates se fait d'après le nombre de rubans et les possibilités des têtes de lecture et écriture.

(1) La classe la plus générale est constituée par les machines de Turing. Une machine de Turing dispose d'un ruban et d'une tête qui peut se déplacer à gauche comme à droite.

(2) Les machines de Turing telles que la tête ne peut écrire que sur une cellule vierge constituent les automates linéaires bornés.

(3) Les automates à pile possèdent deux rubans. Sur l'un une tête (de lecture) peut se déplacer toujours dans le même sens. Sur l'autre, le ruban de manoeuvre, on peut lire et écrire et la tête peut se mouvoir dans les deux sens. L'action que va entreprendre un tel automate à un instant donné est fonction du symbole lu sur la bande d'entrée, du symbole situé à une extrémité fixée du ruban de manoeuvre, et de l'état de l'unité centrale.

(4) Les automates d'états finis sont des automates à piles qui ne possèdent pas de ruban de manoeuvre. Leur seule possibilité d'action est déterminée par l'état et le symbole lu et consiste à changer d'état. Il s'ensuit que les règles de fonctionnement d'un tel automate peuvent se représenter par un graphe dont les sommets sont les états. Une flèche joint les sommets q et q' s'il existe une situation (q,a), où a est le symbole placé dans la cellule sous la tête qui conduit au nouvel état q'.

Chomsky en 1958 et 1962, Lanuweber en 1963 et Kuroda en 1964 établissent la correspondance entre les quatre classes d'automates (non déterministes) décrits ci-dessus avec les quatre types de grammaires.

I.4 LANGAGES VECTORIELS.

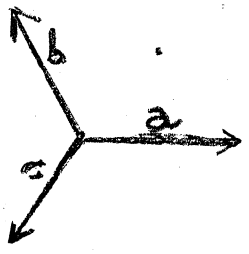
Considérons un système libre ou non de vecteurs dans un espace de dimension finie. Construisons alors l'ensemble des lignes polygonales orientées formées de vecteurs juxtaposés (l'origine de l'un des vecteurs coïncide avec l'extrémité de son prédécesseur), chacun de ces vecteurs étant équipollent à un vecteur du système considéré. On peut définir la juxtaposition de deux lignes polygonales orientées de façon similaire.

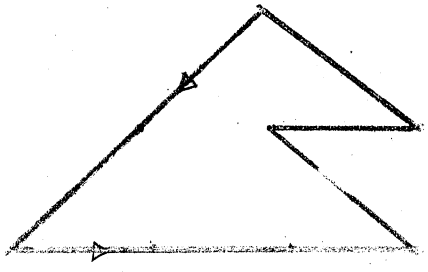
Donnons-nous un vocabulaire et faisons correspondre à chaque lettre du vocabulaire un des vecteurs du système.

L'opération de concaténation des lettres se traduit par la juxtaposition des vecteurs correspondants.

Exemple

V, vocabulaire $V = \{a, b, c\}$
Système de vecteurs du plan :
à la chaîne a^3babc^2 on fait correspondre la ligne polygonale suivante définie à une translation près.





I.4.a Définition.

Un langage vectoriel L (en notation simplifiée) est donné par

- un vocabulaire fini non vide V
- une représentation R application de V dans un espace vectoriel de dimension finie E
- un sous ensemble L du monoïde engendré par V.

Soit M_0 un point de l'espace affine associé à E et $M_0(E)$ l'ensemble des chemins polygonaux finis de E. On construit alors

$\tilde{R} : V^* \to M_0(E)$ par

$\tilde{R}(\wedge) = M_0$

$\tilde{R}(a) = M_0M_1$ $\quad \overrightarrow{M_0M_1}$ équipollent à R(a).

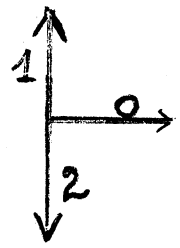
$\tilde{R}(a_1a_2 \ldots a_n) = (M_0M_1 \ldots M_n)$ ligne polygonale de sommets M_i telle que $\overrightarrow{M_{i-1}M_i}$ équipollent à $R(a_i)$.

$\tilde{R}(L) = \{R(x) \; ; \; x \in L\}$ est la représentation du langage L.

I.4.b Exemples.

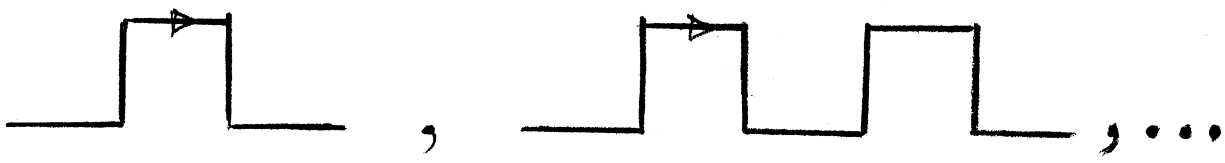
Soit $V = \{0, 1, 2\}$

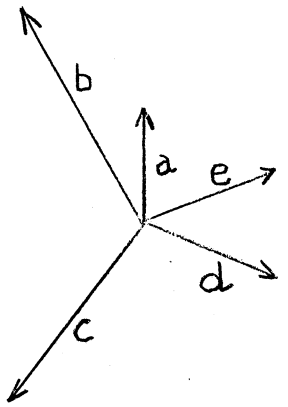
R(V) est donné par la figure

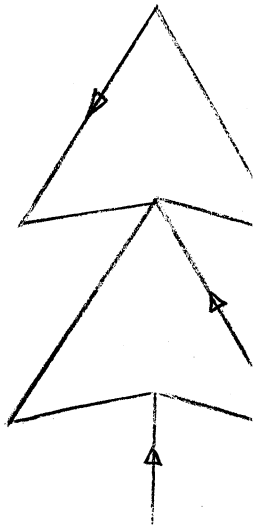


$L = 0102.(0102)^*.0$

L est un langage régulier dont la représentation est l'ensemble des créneaux orientés suivants :







Soit V = {a,b,c,d}

Avec le système de vecteurs ci-contre, le langage à contexte libre

$M = \{aa^n (db)^p (ce)^p \; ; \; n \in N, \; p \in N\}$

est tel que toute chaîne de M admet une représentation en forme de sapin.

I.5 DISCRETISATION DE LIGNES ORIENTEES PLANES.

On obtient les langages vectoriels de façon naturelle par codage des lignes orientées par les procédés qui suivent.

Nous nous plaçons dans le cas de représentations planes. La généralisation à des espaces de dimension quelconque ne présente pas de difficulté particulière.

I.5.a Codage par intersection avec une figure test.

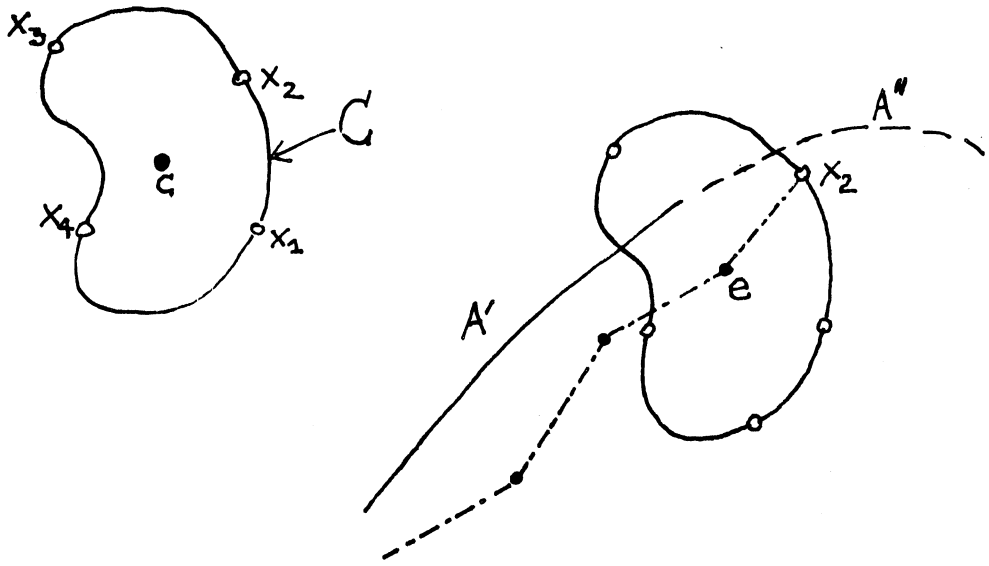
Considérons le cas d'un utilisateur de console de visualisation (display) qui "trace" un arc avec le crayon optique. On sait que ce crayon ne "trace" rien de visible mais peut, grâce à la cellule photoélectrique qui lui est incorporée, repérer l'occurrence d'un rayon lumineux émis par le tube cathodique.

Pour "discrétiser" le tracé d'un arc, on peut procéder de la façon suivante :

On se donne une figure test, définie à une translation près, qui est constituée

- d'un arc fermé C
- d'un nombre fini n de points sur cet arc
- d'un point intérieur, le centre c.

On dispose de plus d'un algorithme permettant d'associer à un point quelconque du pourtour C le point x_i le plus "proche".



Traçons un arc A orienté

A = A' u A"

où A' est déjà tracé. Nous lui avons fait correspondre une certaine ligne polygonale d'extrémité e.

On dessine alors la figure test centrée en e.

En traçant A" on recoupe le pourtour C en x et à x on associe le point x_i. Il suffit alors de rajouter l'arête ex_i à la ligne polygonale et de continuer le procédé.

L'itération n'est possible que si la figure test satisfait à certaines conditions, en particulier il faut être sûr que, lors du tracé, le crayon optique se trouve à l'intérieur de la figure test lorsque celle-ci sera affichée sur l'écran. Indépendamment des contraintes techniques, il nous faut :

$$\inf_i d(c,x_i) > \sup_{x \in C} d(x,x_i(x))$$

$x_i(x)$ étant le point x_i associé à x.

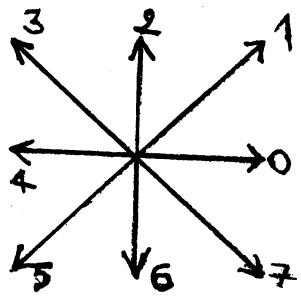
Suivant le choix de la figure test, on obtient différents vocabulaires et diverses représentations. Le système de vecteurs est constitué des

$$\{\overrightarrow{cx_i}\}$$
$$i \leqslant i \leqslant n$$

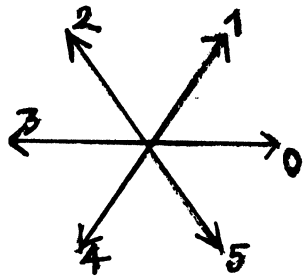
En pratique la figure test est choisie de façon à pouvoir résoudre simplement les questions de convergence entre l'arc orienté et sa ligne polygonale associée.

Exemples

Pour la figure test ci-contre (fig. 1) on désignera par V_8 le vocabulaire $\{0,1,2,3,4,5,6,7\}$



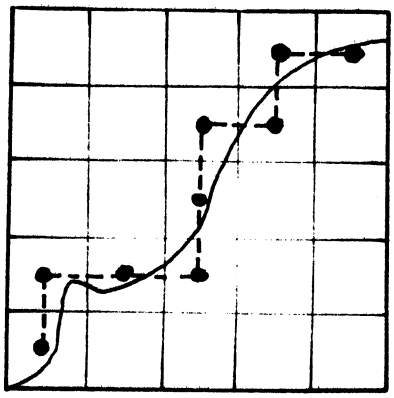
Pour la figure test ci-contre (fig. 2) on notera V_6 le vocabulaire $\{0,1,2,3,4,5\}$



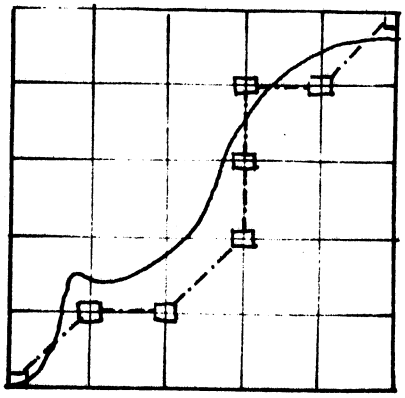
La longueur du vecteur 0 constituera la résolution adoptée.

I.5.b Autres procédés.

On trouve dans [fre], [Roz] et [Mon] d'autres méthodes de discrétisation de lignes qui conduisent à l'utilisation d'alphabets vectoriels. Ces méthodes consistent à définir une partition du plan ou à utiliser une grille du plan au moyen des voisinages de certains points, et à étudier l'intersection de la ligne avec ces voisinages. On reviendra dans la IIIème partie sur le choix de la méthode de discrétisation.



Les voisinages des points sont des carrés.



Les voisinages des points sont des segments de la grille.

I.5.c Code d'un arc orienté.

On associe à l'arc orienté A sa discrétisation qui est la ligne polygonale $(M_0 M_1 \ldots M_n)$ (voir la figure de la page suivante).

Si l'on choisit un langage vectoriel de vocabulaire V, de représentation R telle que

$$R(V) = \{cx_i\}_i \quad \text{vecteurs de la figure test utilisée.}$$

On peut faire correspondre à la ligne polygonale $M_0 M_1 \ldots M_n$ la chaîne x de V^* telle que

$$\tilde{R}(x) = (M_0, M_1, \ldots, M_n)$$

(Ceci sera possible si l'on prend R injective).

Sous cette hypothèse, posant

$$Y = \tilde{R}^{-1}$$

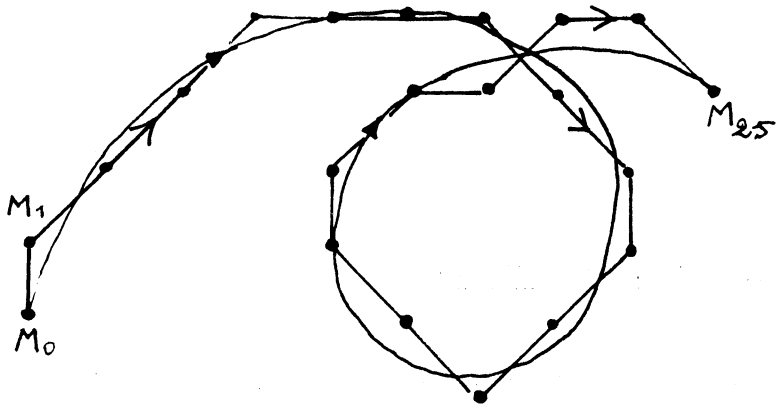
on a

$$Y(M_0, M_1, \ldots, M_n) = x$$

x est la chaîne associée à la ligne polygonale considérée. On dira que x est le code de A.

Dans la suite, nous allons étudier quelques langages vectoriels particuliers. Pour plus de clarté, nous travaillerons avec le vocabulaire V_8 et la représentation R définis à la figure 1 du paragraphe précédent.

Exemple de code d'un arc.



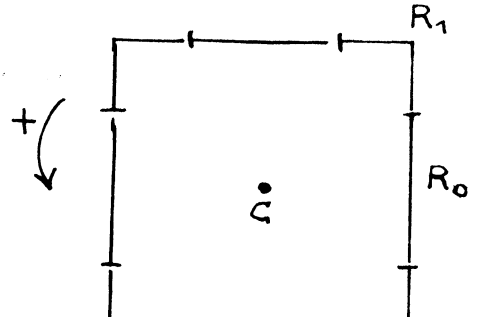
$$\gamma(M_0, M_1, \ldots, M_{25}) = 21^3 0^3 7^2 65^2 3^2 210107$$

L'algorithme qui a un point x de C associe le point x_i le plus proche utilise la distance enclidienne de 2 points du plan. On partitionne le pourtour de C en 8 zones orientées contiguës ouvertes à l'origine, fermées à l'extrémité

$R_0, R_1, \ldots, R_7$

$x_i(x) = j$ si $x \in R_j$

Cet algorithme est décrit et programmé dans la IIIème partie.



CHAPITRE II

ETUDE ALGEBRIQUE DE LA DISCRETISATION DE LIGNES

II.1 INTRODUCTION.

Freeman et Feder étudient les propriétés algébriques de la discrétisation de certaines classes de lignes avec le système V_8. Leur souci est de caractériser les propriétés géométriques de ces lignes par des propriétés algébriques dans le monoïde V_8^*.

Dans les paragraphes qui suivent, nous nous intéressons à deux classes de lignes particulières.

- la classe des lignes fermées. Des considérations géométriques simples permettent de préciser le degré de complexité du langage correspondant à cette classe. Ce langage a été étudié par des moyens combinatoires dans [Fis] pour le rôle important qu'il joue dans la comparaison entre les temps de calcul respectifs des machines de Turing à plusieurs rubans et des machines à plusieurs compteurs.

- la classe des langages monotones et celle des langages convexes sont étudiées pour leurs propriétés algébriques intéressantes et aussi pour leurs applications en reconnaissance de forme (cas des caractères manuscrits).

II.2 LANGAGES FERMES.

II.2.a Définition des langages fermés.

On considère le langage formé de toutes les chaînes x de V^* telles que $\tilde{R}(x)$ soit une ligne polygonale fermée (l'origine coïncide avec l'extrémité).

Notons L.F. le langage fermé construit sur V_8 au moyen de la représentation R utilisée au Chapitre I.

Si l'on trace un arc fermé quelconque, codé par le procédé décrit en I.5.a, il est manifeste que la ligne polygonale obtenue est fermée et que la chaîne résultante est dans L.F. La réciproque est vraie si l'on admet la tolérance $p\sqrt{2}$ (p étant la résolution) dans la distance euclidienne entre l'origine et l'extrémité de l'arc.

Il est intéressant d'exhiber des algorithmes adaptés au problème de la recherche des boucles avec le souci de leurs performances (simplicité d'adaptation aux calculatrices, temps d'exécution).

II.2.b Algorithme de réduction de Freeman.

Freeman procède par réductions successives sur deux éléments non obligatoirement consécutifs d'une chaîne.

$$x = a_1\ a_2\ \ldots\ a_n \qquad (a_i \in V_8).$$

en utilisant le tableau suivant qui exprime la dépendance linéaire des vecteurs de R(V).

	0	1	2	3	4	5	6	7
0			1	2	∧	6	7	
1				22	2	∧	0	00
2	1				3	4	∧	0
3	2	22				44	4	∧
4	∧	2	3				5	6
5	6	∧	4	44				66
6	7	0	∧	4	5			
7		00	0	∧	6	66		

Exemple :

$x = 0075423$

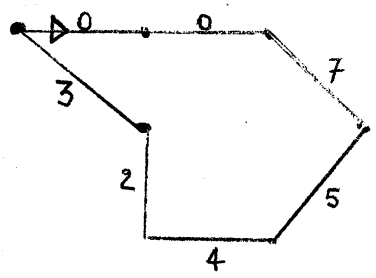
607423

77423

6723

$\wedge 73$

$\wedge\wedge = \wedge$ et $x \in L.F$



Propriété - L.F. est un langage à contexte lié (de type 1).

Il suffit de lire le tableau de réduction à l'envers pour construire une grammaire générant L.F.

Soit $G = (V_T, S, V, P)$

V_T vocabulaire terminal est V_8

V vocabulaire des non-terminaux est :

$V = \{S, A_1, A_2, \ldots, A_7\}$ S axiome

P ensemble des productions ou règles est donné par

$S \rightarrow \wedge$

$S \rightarrow A_0A_4S$

$S \rightarrow A_1A_5S$

$S \rightarrow A_2A_6S$

$S \rightarrow A_3A_7S$

} Génération des courbes fermées élémentaires.

$$
\left.\begin{array}{ll}
A_0 \to A_1A_6 & A_4 \to A_2A_5 \\
A_0 \to A_2A_7 & A_4 \to A_3A_6 \\
A_1 \to A_0A_2 & A_5 \to A_4A_6 \\
A_2 \to A_0A_3 & A_6 \to A_0A_5 \\
A_2 \to A_1A_4 & A_6 \to A_4A_7 \\
A_3 \to A_2A_4 & A_7 \to A_0A_6
\end{array}\right\}
$$

Règles de génération duales des règles de réduction.

$$
\begin{array}{l}
A_0A_0 \to A_1A_7 \\
A_2A_2 \to A_1A_3 \\
A_4A_4 \to A_3A_5 \\
A_6A_6 \to A_5A_7
\end{array}
$$

$A_iA_j \to A_jA_i$	$i,j = 0, 1, \ldots 7$
$A_iS \to S\,A_i$	règles de permutation
$S\,A_i \to A_i\,S$	
$A_i \to i$	règles lexicales

Les productions exhibées sont du type

$\Phi \to \Psi$ avec $l(\Psi) \geq l(\varphi)$ (où $l(x)$ désigne la longueur de la chaîne x).

G est une grammaire de type 1/2. A toute grammaire de type 1/2, on peut associer une grammaire de type 1 générant le même langage (voir [Hop]).

II.2.c Algorithme de décomposition.

La dépendance linéaire des vecteurs de R(V) peut être exploitée de façon plus directe. Pour cela, on extrait une base de R(V), par exemple R(0) et R(1).

Considérons alors l'homomorphisme h :

$h : V_8^* \rightarrow \{0, \overline{0}, 1, \overline{1}\}^*$ défini sur V_8 par

$h(0) = 0$

$h(1) = 1$

$h(2) = \overline{0}1$

$h(3) = \overline{0}\,\overline{0}\,1$

$h(4) = \overline{0}$

$h(5) = \overline{1}$

$h(6) = \overline{0}\,\overline{1}$

$h(7) = 00\overline{1}$

Cet homomorphisme, manifestement surjectif, permet d'étudier les projections sur les vecteurs de base des lignes polygonales.

Soit $z \in W^*$ où l'on pose $W = \{0, \overline{0}, 1, \overline{1}\}$ et notons $\$_0(z)$ le nombre de zéros contenus dans la chaîne z. (De même pour $\$_{\overline{0}}(z)$, $\$_1(z)$, $\$_{\overline{1}}(z)$). Il est alors évident que l'on a

<u>Propriété</u> : $h(L.F) = \{z \in W^* \; ; \; \$_0(z) = \$_{\overline{0}}(z) \text{ et } \$_1(z) = \$_{\overline{1}}(z)\}$

Notons $\overline{L.F} = h(L.F)$

$C_0 = \{z \in W^* \; ; \; \$_0(z) = \$_{\overline{0}}(z)\}$

$C_1 = \{z \in W^* \; ; \; \$_1(z) = \$_{\overline{1}}(z)\}$

C_0(resp C_1) est un langage à contexte libre déterministe (reconnu par un automate à pile déterministe). Cette propriété est citée dans [Gin]. On peut trouver une grammaire à contexte libre générant C_0 dans [Hop].

$\overline{L.F} = C_0 \cap C_1$

<u>Propriété</u> : $\overline{L.F}$ est l'intersection de deux langages à contexte libre déterministes.

En utilisant l'image réciproque de l'homomorphisme h (l'image réciproque par un homomorphisme d'un langage à contexte libre est un langage de même nature). On déduit

$$L.F = h^{-1}(C_0) \cap h^{-1}(C_1)$$

Corollaire : L.F est l'intersection de deux langages à contexte libre.

Si l'on note L.O l'ensemble des chaînes ouvertes

$$L.O = V_8^* - L.F$$

on a $L.O = h^{-1}(W^* - \overline{L.F})$

et $W^* - \overline{L.F} = (W^* - C_0) \cup (W^* - C_1)$

Le complémentaire d'un langage à contexte libre déterministe est un langage de même nature [Gin]. L'union de deux langages à contexte libre est aussi un langage de même nature d'où

Corollaire : L.O est un langage à contexte libre.

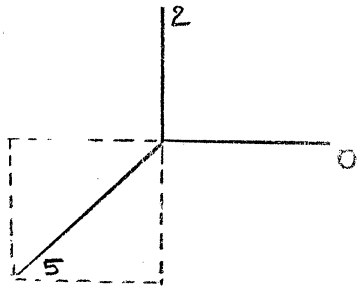
Nous terminerons ce paragraphe en montrant qu'il n'est pas possible de trouver un algorithme de reconnaissance de L.F plus simple dans la classification des algorithmes de Chomsky.

Propriété : L.F n'est pas un langage à contexte libre.

Supposons que L.F soit à contexte libre.

Sachant que l'intersection d'un langage à contexte libre et d'un langage d'états finis est un langage à contexte libre, les langages

$L.F \cap \{0,2,5\}^*$ et $L.F \cap 0^*2^*5^*$ seraient alors à contexte libre.



Or

$$L.F \cap \{0,2,5\}^* = \{x \in V_8^* \ ; \ \$_0(x) = \$_2(x) = \$_5(x)\}$$

Ceci résulte du fait que R(5) se décompose en deux vecteurs opposés à R(0) et R(2) respectivement.

On en déduit :

$$L.F \cap 0^*.\ 2^*.\ 5^* = \{x \in V_8^* \ ; \ x = 0^n\ 2^n\ 5^n \ n \in \mathbb{N}\}.$$

Ce dernier langage n'étant pas à contexte libre (voir [Gin]), il en est de même pour L.F.

Corollaire : $\overline{L.F} = h\ (L.F)$ n'est pas un langage à contexte libre.

Corollaire : Tout langage vectoriel fermé dont la représentation R est telle que les vecteurs de R(V) sont des combinaisons entières de deux d'entre eux est un langage à contexte lié sans être un langage à contexte libre.

On construit de manière analogue l'homomorphisme h à valeurs dans W^* de façon que le langage considéré soit l'image réciproque de $\overline{L.F}$.

Il en est ainsi pour le vocabulaire V_6 et la représentation R définis au chapitre I. Cette propriété répond à une question posée dans [Fed].

II.3 LANGAGES MONOTONES ET CONVEXES.

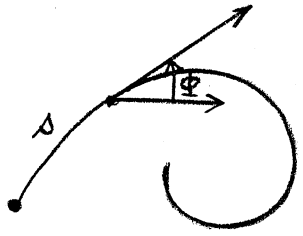
Exploitons la représentation R de l'alphabet dans un ensemble de vecteurs finis pour traduire certaines propriétés géométriques telles que la monotonie et la variation de pente le long d'une courbe.

II.3.a Les langages monotones M^+ et M^-.

Supposons que l'on trace un arc suffisamment régulier pour qu'en tout point l'on puisse parler de l'abscisse curviligne s et de l'angle Φ de la tangente avec une direction fixe.

On suppose, de plus, que Φ (s) est une fonction monotone.

Exemple : Ces hypothèses sont réalisées en première approximation lorsqu'on trace les caractères alphanumériques suivants : B, C, O, α, γ, 3, 9, 6.



De même, nous dirons qu'une ligne polygonale formée avec les vecteurs correspondant à V_8 est monotone si localement elle vérifie cette même propriété.

Soit $x \in V_8^*$ $\quad x = a_1\, a_2 \ldots a_n$

la chaîne x est dite monotone croissante si elle vérifie

$$m^+ \quad \begin{cases} \forall i = 1, 2\ldots, n-1 \\ a_{i+1} = a_i \text{ ou } a_i^+ \text{ ou } a_i^{++} \text{ ou } a_i^{+++}. \end{cases}$$

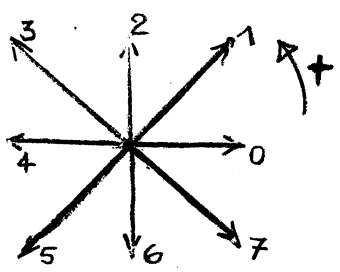
La chaîne x est dite monotone décroissante si elle vérifie

$$m^- \quad \begin{cases} \forall i = 1, 2 \ldots, n-1 \\ a_{i+1} = a_i \text{ ou } a_i^- \text{ ou } a_i^{--} \text{ ou } a_i^{---}. \end{cases}$$

Les indices supérieurs + et - indiquent que l'on prend le successeur ou le prédécesseur de l'élément considéré dans l'ensemble

$$\{0, 1, 2, \ldots, 7\}$$

selon l'orientation définie par la flèche de la figure



exemple $7^+ = 0$

$0^{--} = 6$

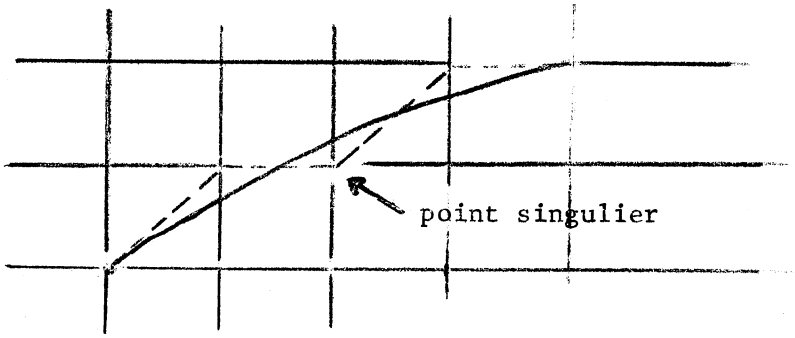
$3^{++} = 5$

Définition - $M^+ = \{x \in V_8^* \; ; \; x \text{ vérifie } m^+\}$

$M^- = \{x \in V_8^* \; ; \; x \text{ vérifie } m^-\}$

Il aurait été particulièrement simple que l'hypothèse faite sur l'arc ($\Phi(s)$ monotone) se traduise par la monotonie de la chaîne qui représente son code par le procédé de Freeman.

Ceci n'est pas vrai en général comme le montre l'exemple suivant :



La chaîne 1 0 1 0 n'est pas dans M^- .

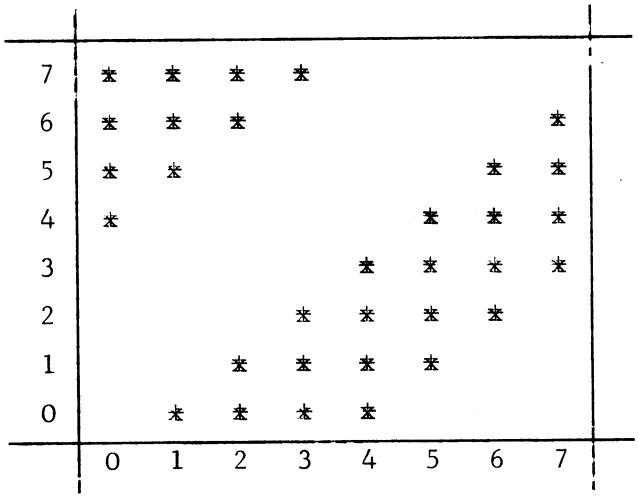
On verra au paragraphe II.3.c qu'il est possible de trouver un langage plus général que M^+ ou M^- et dont les chaînes traduisent encore la propriété de monotonie en tenant compte des points singuliers.

Propriété : M^+ et M^- sont des K-langages locaux.

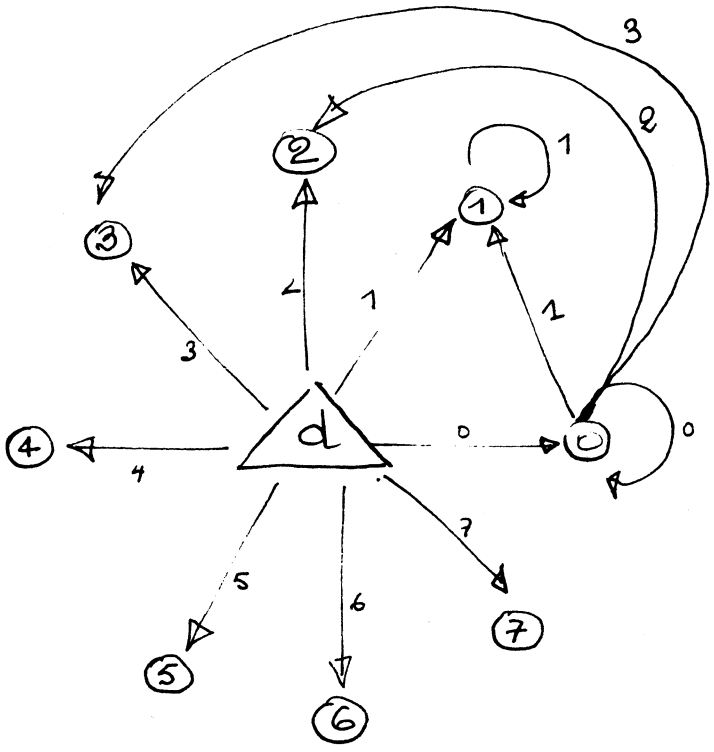
Il est clair que M^+ est le K-langage local défini par les paramètres:

$A = B = V_8$

$I \subseteq V \times V$ donné par le tableau :



L'automate d'états finis qui reconnaît M^+ est donné par le graphe suivant : (on complètera les flèches par permutation circulaire autour de d)



D'autre part, considérons l'application m :

$$m : V_8^* \rightarrow V_8^*$$

$$x = a_1 a_2 \ldots a_n \qquad m(x) = a_n\, a_{n-1} \ldots\, a_1$$

m(x) est l'image miroir de x. Si K est un langage régulier (ou un K-langage local). m(K) est un langage de même nature.

On montre facilement que $m(M^+) = M^-$.

II.3.b Variation de pente.

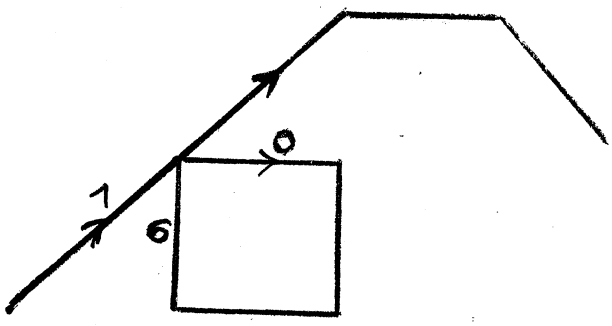
Soit $x = a_1\, a_2 \ldots a_n \quad a_i \in V_8 \quad i = 1, 2 \ldots, n. \quad x \in M^+ \cup M^-$

Définition - La variation de pente de la chaîne x est l'entier relatif défini par

$$v(x) = \sum_{i=1}^{n-1} (a_{i+1} - a_i)_8$$

L'indice inférieur 8 indique que les différences $a_{i+1} - a_i$ sont calculées modulo 8 et prennent leurs valeurs dans le système de représentants de $\mathbb{Z}$ modulo 8 : $\{-3, -2, -1, 0, 1, 2, 3, 4\}$

Exemple :



$x = 10642107$

$v(x) = -1-2-2-2-1-1-1 = -10$

Si l'on se donne un entier relatif $p \in \mathbb{Z}$, la détermination de toutes les chaînes de V_8^* de variation, de pente égale à p peut être faite par un automate à une pile. Dans le cas où l'on travaille avec les chaînes monotones, ce problème a une solution plus simple. Un automate d'états finis suffit comme le montre ce qui suit.

<u>Propriété</u> - $K_p = \{x \in M^+ \ ; \ v(x) = p\}$ où $p \in \mathbb{N}$ est un langage régulier pour tout p.

La démonstration consiste à construire l'automate d'états finis qui reconnaît K_p.

L'alphabet d'entrée de l'automate est V_8.

Son ensemble d'états est

$$S = \{d\} \cup \{\bar{p}\} \cup \{(k, a_i) \ ; \ 0 \leqslant k \leqslant p \qquad a_i \in V_8\}$$

d état de départ

$\bar{p}$ état "puits"

(k, a_i) est l'état obtenu après lecture par l'automate d'une chaîne $x = a_1 \ldots a_i$ telle que $v(x) = k$

La fonction de transition de l'automate est :

$f : S \times V_8 \to S$ définie par

$$(d, a_i) \mapsto (0, a_i)$$

$$((k, a_i) \ a_i) \rightsquigarrow (k, a_i)$$

$$((k, a_i) \ a_i^+) \rightsquigarrow \begin{cases} (k+1, a_i^+) \text{ si } k+1 \leqslant p \\ \bar{p} \text{ sinon.} \end{cases}$$

$$((k, a_i) \ a_i^{++}) \rightsquigarrow \begin{cases} (k+2, a_i^{++}) \text{ si } k+2 \leqslant p \\ \bar{p} \text{ sinon.} \end{cases}$$

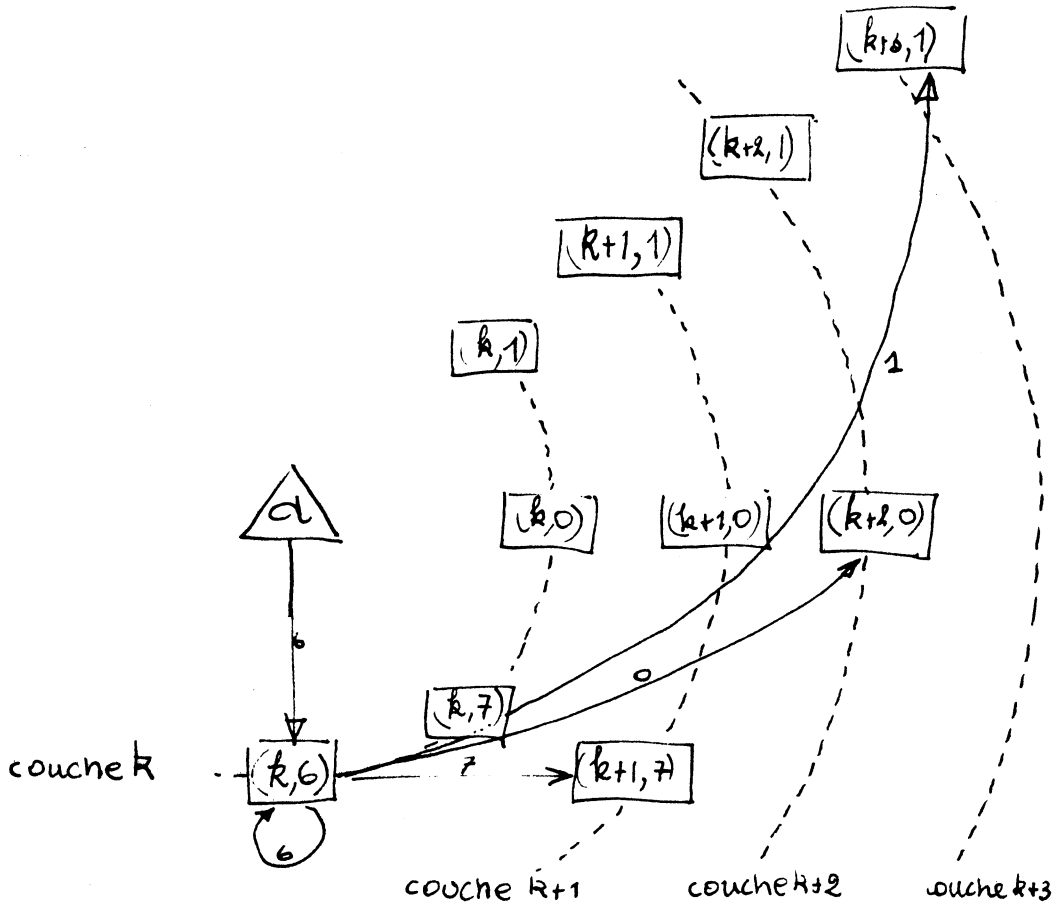
$$((k, a_i) \ a_i^{+++}) \rightsquigarrow \begin{cases} (k+3, a_i^{+++}) \text{ si } k+3 \leqslant p \\ \bar{p} \text{ sinon.} \end{cases}$$

Pour tout couple $(s, a_i) \in S \times V_8$ où f n'est pas encore définie, on posera

$$f((s, a_i)) = \bar{p}$$

L'ensemble des états finaux est $F = \bigsqcup_{a_i \in V_8} (p, a_i)$

Le graphe de l'automate peut être représenté par la figure ci-après où les états (k, a_i) pour k fixé sont répartis régulièrement sur la couche d'indice k (la représentation totale du graphe est obtenue en faisant varier k de 0 à p et en faisant les permutations circulaires convenables).



Corollaire : Les langages

$\overline{K_p} = \{x \in M^- \ ; \ v(x) = -p\}$

et

$K_B = \{x \in M^+ \cup M^- \ ; \ v(x) \in B\}$ où B est une partie bornée de $\mathbb{Z}$ sont réguliers.

$\overline{K_p}$ est régulier en vertu des propriétés de l'application miroir et de la relation $\overline{K_p} = m(K_p)$.

D'autre part, posant $B \cap \mathbb{N} = B'$ et $B \cap (-\mathbb{N}) = B''$

$$K_B = \bigcup_{p \in B'} K_p \cup \bigcup_{p \in B''} \overline{K_p} .$$

K_B est l'union de langages réguliers.

Ce dernier résultat indique que les conditions de variation de pente le long d'une ligne monotone se traduiront par des algorithmes simples.

II.3.c Langages convexes.

La définition de convexité utilisée ici diffère notablement de celle de [Fed] et conduit à des résultats plus simples. On peut démontrer que le code de certaines courbes (courbes telles que $\varphi(s)$ soit monotone et que la variation de φ pour deux points proches par rapport à la résolution adoptée ne soit pas supérieure à une constante) fournit une chaîne convexe.

Soit $x \in V_8^*$ $\quad x = a_1 a_2 \ldots a_n$.

Définitions - x est strictement monotone croissante si elle satisfait à

$$sm^+ \begin{cases} \forall i = 1, 2, \ldots n-1 \\ a_{i+1} = a_i^+ \text{ ou } a_i^{++} \text{ ou } a_i^{+++} \end{cases}$$

(de même pour strictement monotone décroissante par dualité).

x est dite convexe (croissante) si elle satisfait aux deux conditions

i) il existe une chaîne $y = a_{i_1} a_{i_2} \ldots a_{i_p}$ extraite de x strictement croissante avec $i_1 = 1$ et $i_1 < i_2 < \ldots < i_p \leqslant n$.

ii) pour tout $j = 1, 2, \ldots, p-1$, les éléments de la chaîne x d'indice k avec $i_j \leqslant k < i_{j+1}$ sont dans $\{a_{i_j}, a_{i_j}^-\}$ et ceux d'indice k avec $i_p \leqslant k \leqslant n$ sont dans $\{a_{i_p}, a_{i_p}^-\}$

<u>Propriété</u> : Pour une chaîne x convexe, il existe une chaîne y et une seule vérifiant i) et ii).

Soit une autre chaîne z extraite de x vérifiant i) et ii)

$$z = a_{j_1} \dots a_{j_k} \qquad j_1 = i_1 = 1$$

Soit r le plus petit entier tel que $j_r \neq i_r$

On a par exemple $j_r < i_r$ et $j_{r-1} = i_{r-1}$

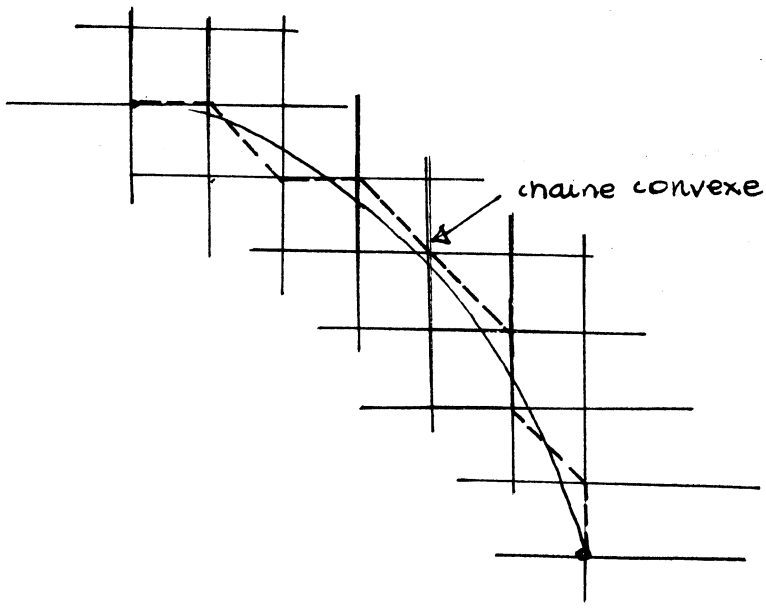
On déduit $i_{r-1} < j_r < i_r$ donc $a_{j_r} \in \{a_{i_{r-1}}, a^-_{i_{r-1}}\}$

D'autre part $a_{j_r} \in \{a_{j_{r-1}}, a^-_{j_{r-1}}\}$

L'hypothèse de stricte monotonie de z fournit :

$$a_{j_r} \in \{a^+_{j_{r-1}}, a^{++}_{j_{r-1}}, a^{+++}_{j_{r-1}}\}$$

ce qui est incompatible. D'où $i_r = j_r$ $\forall r$ et $y = z$



Notons C^+ l'ensemble des chaînes convexes (on obtient C^- par une définition duale).

Propriété : C^+ (resp C^-) est un langage régulier.

L'ensemble des chaînes strictement croissantes est un K-langage local. On génère une chaîne convexe en substituant à une lettre a_i de la chaîne strictement croissante une chaîne de l'expression régulière :

$$a_i \cdot (\{a_i , a_i^- \})^*$$

Nous réalisons ainsi une opération de substitution portant sur des langages réguliers. Le résultat obtenu est lui-même régulier (Propriété de l'opération de substitution).

On déduit de $C^- = m(C^+)$ un résultat analogue.

De façon constructive, donnons le graphe de l'automate qui reconnaît C^+. L'ensemble de ses états est :

$$S = \{d\} \cup \{p\} \cup \{0, 1, 2, \ldots, 7\} \cup \{\bar{0}, \bar{1}, \ldots \bar{7}\}.$$

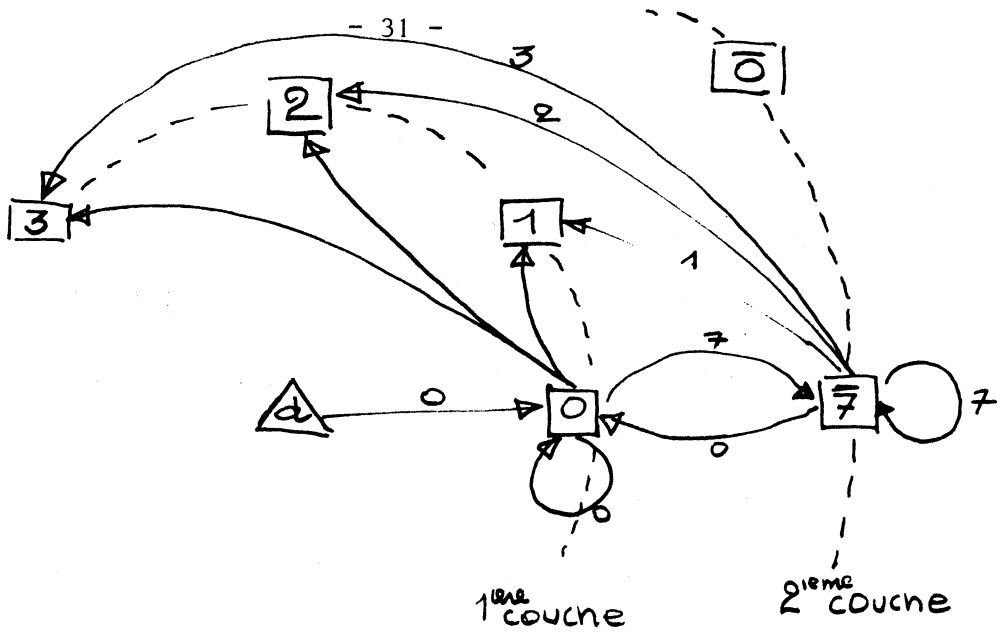
d et p sont l'état initial et l'état puits.

Les autres états sont répartis sur deux couches et constituent les états finaux.

On complète la figure par les permutations circulaires convenables.

Si pour un doublet (s, a_i) de $S \times V_8$ il n'y a pas de flèche partant de l'état s et portant le renseigenement a_i , on pose :

$$f(s, a_i) = p$$



II.3.d Enveloppe monotone.

La définition d'une chaîne convexe donnée en 3-C permet de lui associer une chaîne monotone.

Soit $x \in C^+$ par exemple

$x = a_1\ a_2 \ldots a_n$

$y = a_{i_1}\ a_{i_2} \ldots a_{i_p}$ vérifiant i) et ii).

Dans la sous-chaîne de x $a_{i_j}\ a_{i_j+1} \ldots a_{i_{j+1}-1}$

où chaque lettre est soit a_{i_j} ou $\bar{a}_{i_j}$, commutons les éléments a_{i_j} et $\bar{a}_{i_j}$ de façon à mettre les $\bar{a}_{i_j}$ en tête de cette chaîne.

Répétons cette opération pour tous les indices j tels que

$$1 \leqslant j \leqslant p-1$$

Il est facile de vérifier que l'on obtient une chaîne notée $e^+(x)$ qui est monotone croissante.

$e^+(x)$ est l'enveloppe monotone de la chaîne x.

Exemple : Soit la chaîne

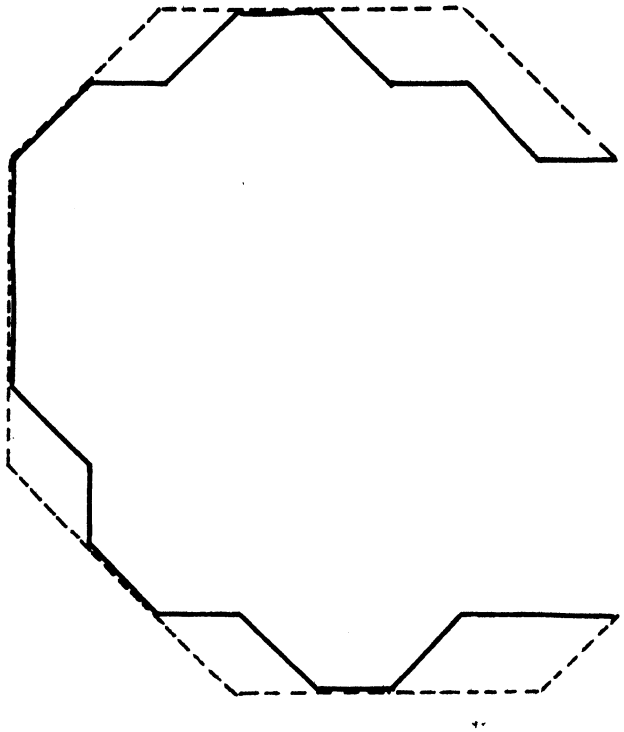
$$x = \underline{4}\ 3\ 4\ 3\ 4\ \underline{5}\ 4\ 5\ \underline{6}\ 6\ 6\ \underline{7}\ 6\ 7\ \underline{0}\ 7\ 0\ \underline{1}\ 0\ 0$$

la sous-chaîne extraite strictement croissante est

$$y = 4\ 5\ 6\ 7\ 0\ 1$$

la chaîne $e^{+}(x)$ est :

$$e^{+}(x) = 3\ 3\ 4\ 4\ 4\ 4\ 5\ 5\ 6\ 6\ 6\ 6\ 7\ 7\ 7\ 0\ 0\ 0\ 0\ 1$$



II.3.e Application.

Dans [Mie], on étudie certains sous-ensembles de chaînes convexes. Ces chaînes admettent des enveloppes monotones et la "distance" entre une telle chaîne et son enveloppe reste inférieure à une quantité déterminée. Ceci permet d'étudier algébriquement des courbes ou portion de courbes (telles que les lettres) en travaillant directement sur l'enveloppe monotone de leur code. En effet, sur cette enveloppe, on peut spécifier des conditions initiales ou finales de pente, des conditions de variation de pente au moyen des langages locaux ou réguliers.

IIème PARTIE

§.1 POSITION DU PROBLEME

L'étude qui suit a été menée dans le cadre de l'étude d'un système automatique de résolution de problèmes aux limites. Un tel système pose de nombreuses difficultés dans l'ordre du traitement par des méthodes mathématiques, comme dans l'ordre de l'acquisition de données graphiques par un calculateur et de la manipulation de ces données. Le lecteur trouvera ces problèmes évoqués dans [For], [Til] et [Mag].

1.1 EXEMPLES DE PROBLEMES AUX LIMITES

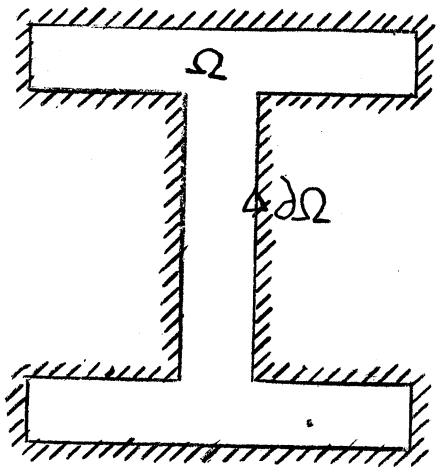
Dans les problèmes aux limites qui vont nous intéresser, on cherche à déterminer une fonction inconnue (ou plusieurs fonctions) de deux variables définie dans un domaine du plan de ces deux variables. Les divers aspects graphiques sont montrés dans les trois exemples qui suivent.

PB1 :

La torsion d'une poutre, dont la section est donnée par la figure ci-contre, est la solution U du système d'équations :

(α) $\Delta U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = -1$ dans Ω

(β) $U = 0$ sur $\partial\Omega$



On désigne par la lettre α l'équation valable à l'intérieur du domaine, et par la lettre β l'équation valable sur la frontière (condition aux limites).

PB2 :

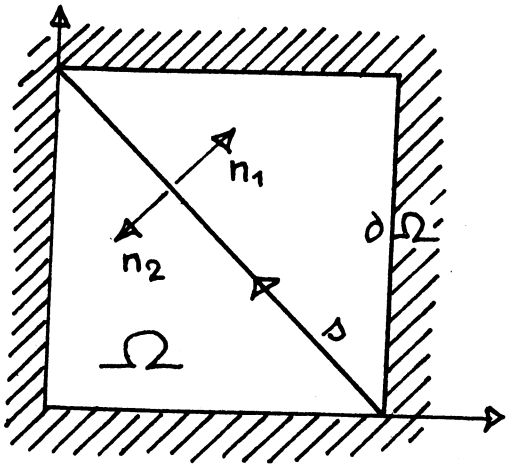
L'amplitude de vibration d'un point situé sur une membrane carrée traversée par un fil élastique suivant une diagonale est donnée par

(α) $\Delta U = \lambda U$ dans Ω

(β1) $U = 0$ sur $\partial\Omega$

(β2) $\frac{\partial U}{\partial n_1} + \frac{\partial U}{\partial n_2} + a \frac{\partial^2 U}{\partial s^2} + bU = 0$ sur D

(problème de valeurs et vecteurs propres).



a, b, λ sont des constantes
s est l'abscisse curviligne sur D.

PB3 :

Considérons le problème de mécanique des fluides qui consiste à déterminer la section d'un barrage

(α) $\Delta U = 0$

(β1) $U = Cste$ sur AB

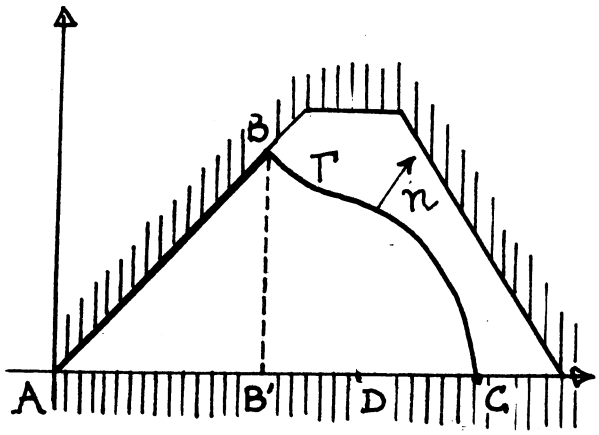
(β2) $U = 0$ sur DC

(β3) $\frac{\partial U}{\partial y} = 0$ sur AD

(β4) $U = y$ sur la frontière Γ qui est à déterminer

(β5) $\frac{\partial U}{\partial n} = 0$ sur la frontière

Une particularité de cet exemple est de posséder une frontière libre. Pour résoudre ce problème, on peut faire choix d'une frontière Γ sur laquelle on impose (β4) ou (β5) . On détermine alors U et l'on vérifie, à postériori, la condition aux limites inemployée. Si celle-ci n'est pas satisfaite, on ajuste Γ et on recommence.



On voit sur cet exemple qu'il peut être intéressant de modifier une frontière en gardant le bénéfice des portions de domaine invariantes (le triangle ABB').

1.2. LA METHODE DES DIFFERENCES FINIES.

On transforme le problème original en continu en un problème discret associé en vue d'une résolution numérique.

Ω est un domaine plan.

U est la fonction inconnue que l'on cherche à déterminer sur Ω au moyen du système

(α) EDP(U) = 0 dans Ω

(β1) CL1(U) = 0 sur $\partial\Omega_1$

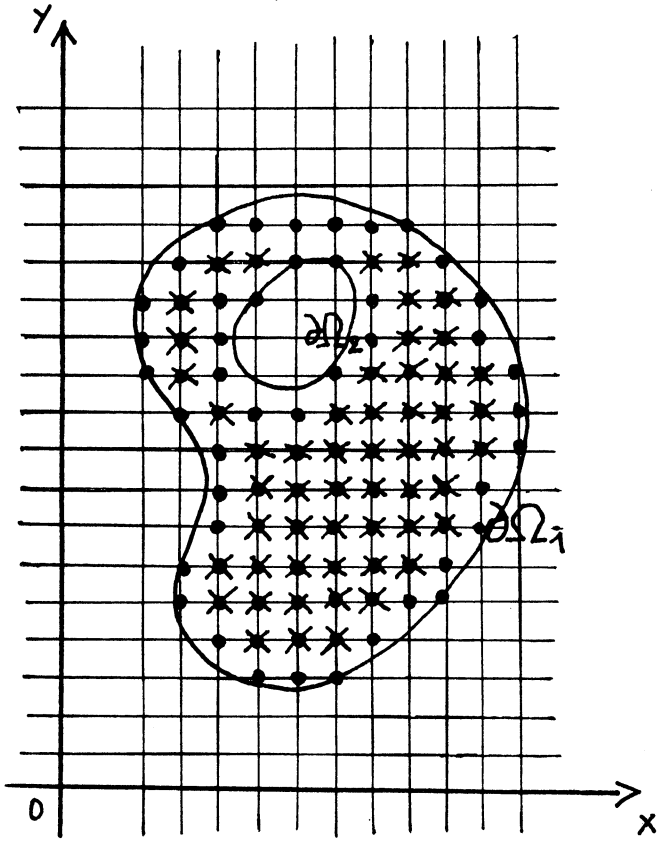
(β2) CL2(U) = 0 sur $\partial\Omega_2$

⋮

(βN) CLN(U) = 0 sur $\partial\Omega_N$

EDP pour équation aux dérivées partielles

CL pour condition aux limites.



On considère un nombre fini de points intérieurs à Ω déterminés par intersection de deux familles de droites. Soit Ω_H cet ensemble.

On distingue parmi les points de Ω_H les points qui sont centre d'une croix située à l'intérieur de Ω_H, ensemble $\mathring{\Omega}_H$, et les points de la frontière, ensemble $\Omega_H - \mathring{\Omega}_H$.

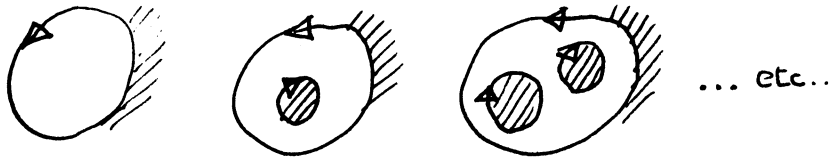
Le principe de la méthode des différences finies consiste à associer à chaque point de Ω_H une équation linéaire, ayant au plus comme inconnues les valeurs de la fonction, cherchée au point considéré, et en ses quatre voisins immédiats. Si ce point appartient à Ω_H on utilise l'équation EDP sinon on utilise l'équation CL correspondante.

L'importance des informations graphiques et numériques qu'il est nécessaire de stocker pour les problèmes que l'on rencontre dans la pratique (10.000 points dans Ω_H) nous oblige à définir des structures pour ces domaines.

§.2 DEFINITIONS

2.1. DOMAINES ADMISSIBLES.

Ce sont les domaines bornés qui, par une déformation continue du plan, sont superposables avec un des domaines suivants.



La frontière de ces domaines possède les propriétés d'orientation et de régularité suivantes :

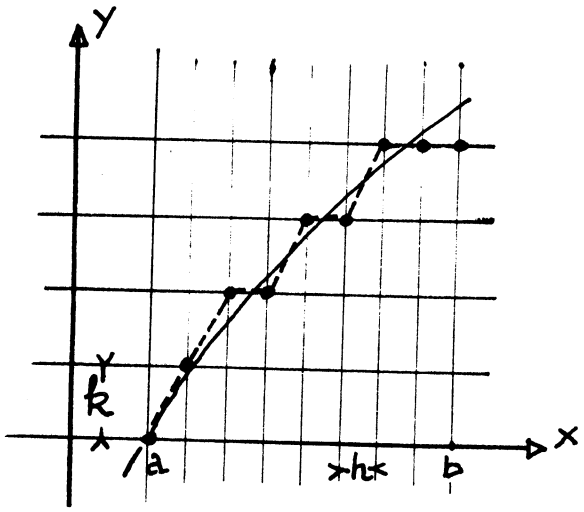
(O) Orientation : l'intérieur du domaine se trouve sur la gauche d'un observateur parcourant la frontière.

(R) Régularité : la frontière est décomposable en un nombre fini d'arcs (arcs suffisamment réguliers pour que les conditions aux limites correspondantes aient un sens).
On suppose que pour chacun de ces arcs, on dispose d'un algorithme qui permet de calculer les intersections de l'arc avec une grille régulière.

Ceci est le cas si, par exemple, on a une représentation paramétrique de l'arc, du type

$$\{(x, f(x)) \; ; \; a \leqslant x \leqslant b\}$$

ou $\{(f(y), y) \; ; \; c \leqslant y \leqslant d\}$



On peut prendre alors pour ensemble des points de la portion de frontière correspondant à l'arc, l'ensemble :

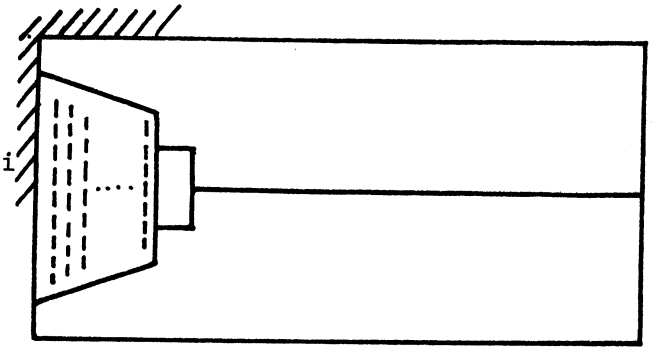
$$\{(nh, E(\frac{f(nh)}{k} + 1/2).k) ; a \leqslant nh \leqslant b\}$$

(E(u) désigne le plus grand entier relatif inférieur ou égal à u).

On admet la possibilité d'avoir, à l'intérieur des domaines, des lignes orientées, qui, dans la pratique, correspondent à la séparation de deux milieux d'un même domaine où les conditions physiques diffèrent.

On les appellera, par la suite, les séparatrices,pour les distinguer des arcs définissant la frontière du domaine. Ces séparatrices vérifient les conditions de régularité ci-dessus.

Exemple : isolant 1 [Des]
On cherche dans ce problème à trouver une position des séparatrices (en **pointillé**) qui sont dans le trapèze de gauche. Cette position devant **vérifier** certains critères.



2.2. GRILLE RESOLUTION - BLOC.

Pour un domaine Ω inclus dans le rectangle

$$[m_x, M_x] \times [m_y, M_y]$$

on se donne deux subdivisions

$$x_0 = m_x, x_1, \ldots, x_n = M_x.$$

$$y_0 = m_y, y_1, \ldots, y_m = M_y.$$

Sur l'intervalle $[x_i, x_{i+1}]$ on fait choix d'un pas h_i.
Sur l'intervalle $[y_j, y_{j+1}]$ on fait choix d'un pas k_j.

On obtient donc une grille comme indiquée sur la figure 2 avec des pas constants en x ou en y suivant des bandes horizontales ou verticales.

On peut généraliser cette notion en définissant une partition du rectangle toujours au moyen des deux subdivisions ci-dessus. Un <u>bloc</u> est alors le rectangle

$$B_{i,j} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$$

Dans le bloc $B_{i,j}$ on définit la <u>résolution</u>

$\rho_{i,j} = (h_i, k_j)$ qui donne les pas de discrétisation horizontaux et verticaux.

On a ainsi une grille, avec des pas constants par blocs, comme indiqué sur la figure 1.

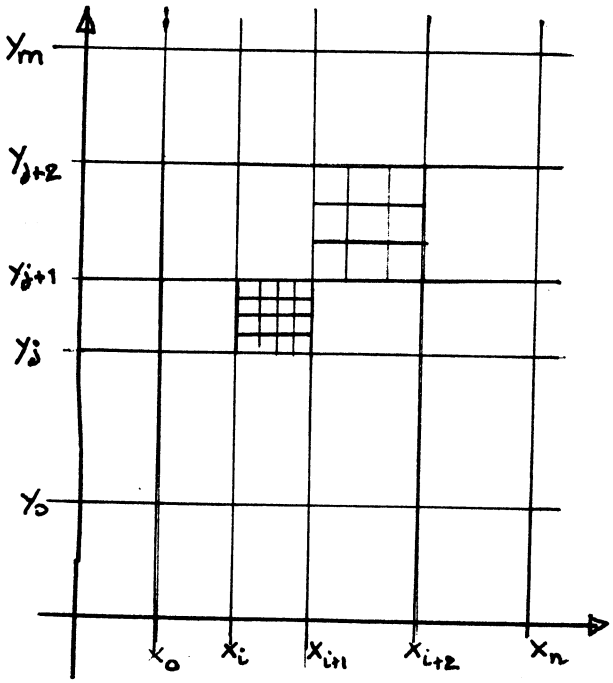


Figure 1

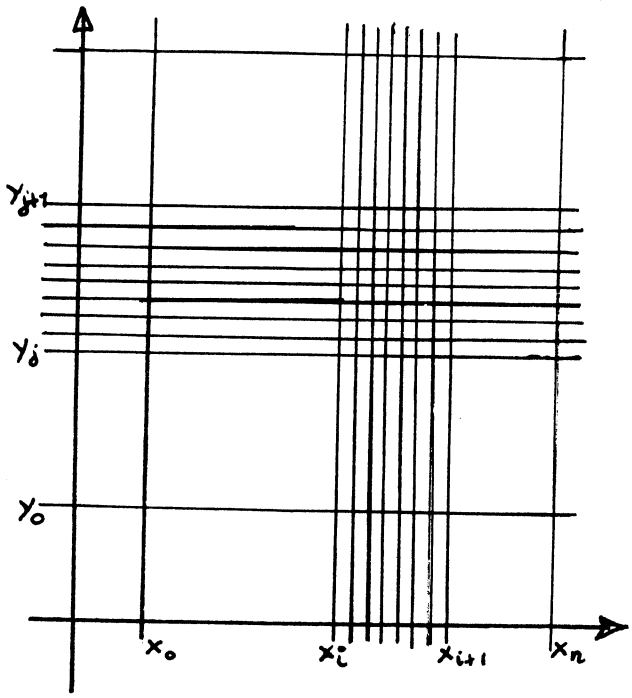


Figure 2

Dans un problème aux limites, le choix de la grille et de la résolution peut être un élément décisif parce qu'il conditionne l'encombrement de la mémoire, la précision des calculs, et le temps de traitement proprement dit. Le choix de grilles formées de droites orthogonales est souvent justifié par une plus grande simplicité de la méthode mathématique. L'organisation de la grille par bandes ou par blocs, permet d'adapter les pas de discrétisation, de façon que la discrétisation d'un arc soit composée de points de la grille. Dans l'exemple PB3, pour le bloc contenant ABB', on peut prendre :

$$h \text{ quelconque et } k = \frac{h}{\alpha\pi} .$$

La possibilité de faire varier les pas d'un bloc à un autre permet aussi de mieux étudier une région critique, sans pour autant grandir exagérément le nombre de points intérieurs au domaine, et par là, l'encombrement de la mémoire. Ces régions critiques peuvent être la proximité de deux arcs frontières, d'un arc et d'une séparatrice, etc...

2.3. CARTE D'UN BLOC.

Soit Ω un domaine admissible.

Considérons une grille et B un des blocs de cette grille (les indices sont omis pour raison de commodité).

Par la suite, nous appellerons carte, désignée par C, une liste d'informations qui concernent les arcs frontières et les séparatrices du domaine Ω contenu dans B.

Cette liste comprend notamment :

- les coordonnées absolues d'un point de la carte choisi comme origine relative
- la valeur des pas constants horizontaux et verticaux (*)
- à chaque arc frontière Γ_i (resp.^{t} séparatrice S_j), on associe, par l'algorithme que l'on se donne (**) , la discrétisation de cette ligne, que l'on représente par une chaîne z_i (resp.^{t} z_j) formée sur le vocabulaire $\overline{V}_8 = \{\overline{0}, \overline{1}, \ldots \overline{7}\}$. Cette chaîne est précédée des coordonnées relatives de l'origine de la ligne, de son type (f pour frontière, s pour séparatrice), et de l'indice de cette ligne. Cet indice permet de faire référence à une certaine équation exprimant une condition aux limites valable pour cette ligne.

Exemple : isolant 2.

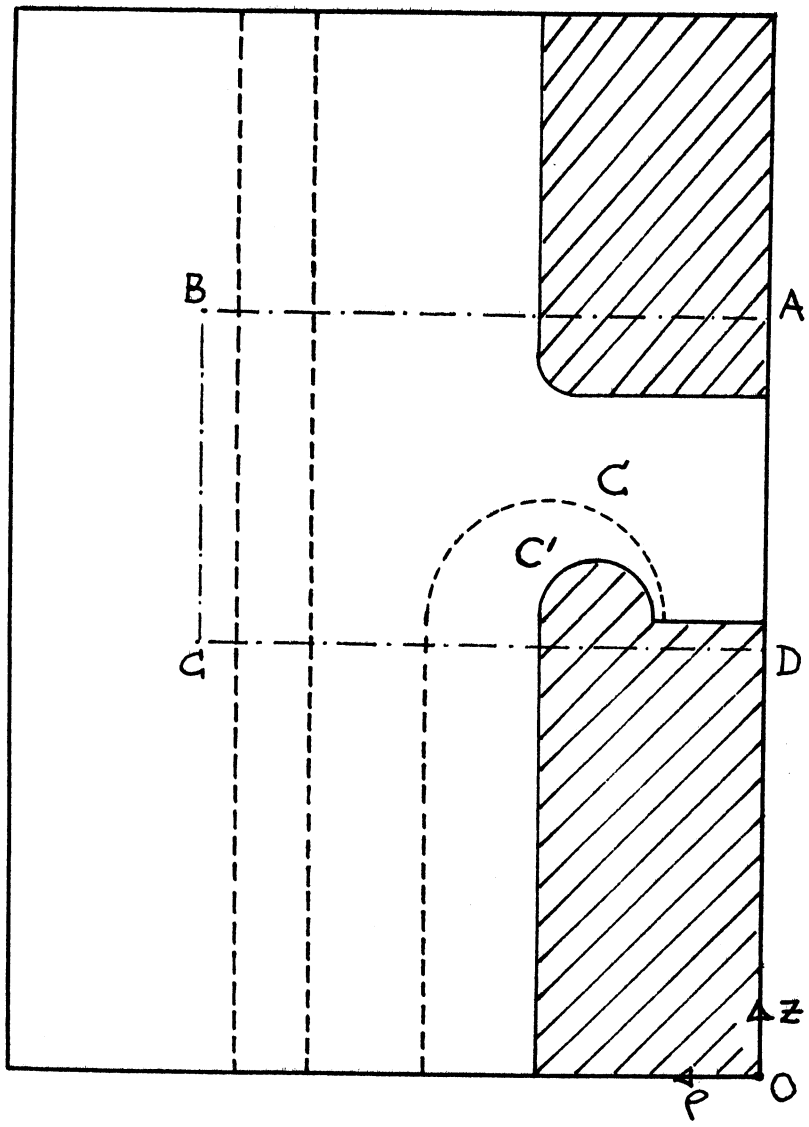
Pour l'étude de la répartition du potentiel dans l'isolant de révolution représenté en coordonnées cylindriques par la figure isolant 2, on distingue la zone ABCD qui comprend la région critique des deux demi-cercles C et C'.

---

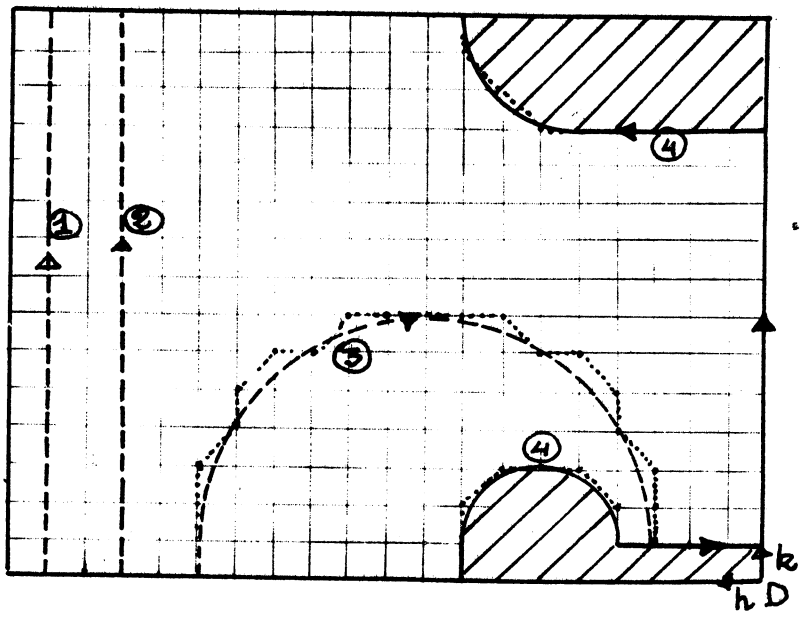
* Pour des repères non orthogonaux et des alphabets différents, voir le paragraphe changement d'alphabet.

** hypothèse (R) du paragraphe 2.1.

Figure isolant 2



partie ABCD



A supposer que la grille soit celle de la figure relative à ABCD (dans une étude réelle, le nombre de points de la grille intérieurs à cette zone est de l'ordre de 5000); l'organisation de la carte correspondante est par exemple :

. $x_D = 0$, $y_D = 600$

. $h = 2$, $k = 2$

. $(19, 0, s, 1)\ \bar{2}^{15}$

$(17, 0, s, 2)\ \bar{2}^{15}$

$(15, 0, s, 3)\ \bar{2}^{3}\ (1\bar{2})^{2}\ 0\bar{1}\bar{0}^{4}\ \bar{7}0\ (\bar{7}\bar{6})^{2}\ \bar{6}$

$(8, 0, f, 4)\ 2^{2}\ 10^{2}\ 760^{4}\ 2^{11}\ 4^{6}\ 3^{2}\ 2$

§.3 DISCRETISATION DE DOMAINES.

3.1. DESCRIPTION DE L'INTERIEUR $\overset{\circ}{\Omega}_H$.

Considérons Ω un domaine admissible et Ω_H l'ensemble des points intérieurs de Ω relativement à une grille donnée G.

Il s'agit de donner une description de $\overset{\circ}{\Omega}_H$ telle que l'on puisse résoudre les problèmes suivants :

- Si M appartient à G ; M appartient-il à Ω_H ?
- Si M appartient à Ω_H ; M appartient-il à $\overset{\circ}{\Omega}_H$?
- Indexation des points de Ω_H.
- Pour M appartenant à Ω_H quels sont ses voisins immédiats ?

Plaçons-nous au niveau d'un bloc B, et désignons encore, pour ne pas alourdir les notations, par Ω, Ω_H et $\overset{\circ}{\Omega}_H$ les ensembles $\Omega \cap B$, $\Omega_H \cap B$ et $\overset{\circ}{\Omega}_H \cap B$. (Laissons de côté le cas des points des interfaces, points situés sur la frontière de deux blocs contigus).

On constate qu'il est inutile d'obtenir une description explicite de tous les points de Ω_H, du moins au niveau de l'introduction des données graphiques en machine. Cette remarque, jointe à l'idée simple qui consiste à décrire un domaine en parcourant sa frontière, nous conduit aux algorithmes qui suivent et qui fournissent une description implicite de Ω_H et $\overset{\circ}{\Omega}_H$.

Cette description se fait en fonction d'une direction privilégiée de balayage (par ligne, par colonne, suivant telle diagonale, etc...) ; ce qui permet à l'utilisateur une plus grande liberté dans le choix de la méthode de traitement.(relaxation ou surrelaxation, par ligne, par colonne, etc...). Les algorithmes s'adaptent aux cas de grilles formées par des figures élémentaires autres que des rectangles ; par exemple : les triangles, les hexagones.

En vertu de l'hypothèse (R) faite dans la définition des domaines admissibles, on ne considèrera dans la suite que des frontières et séparatrices de domaines qui sont des lignes polygonales formées sur le vocabulaire $\bar{V}_8$. Dans ces conditions, il est clair qu'un point qui est dans Ω_H et qui n'est pas sur ces lignes polygonales, est nécessairement dans $\mathring{\Omega}_H$ (propriété particulière de cette discrétisation).

3.1.a. Liste d'intérieur

Considérons Ω la partie d'un domaine relative à un bloc B.

On désire obtenir la description du domaine Ω_H pour une direction de balayage δ.

Donnons-nous un repère R : (O, γ, δ) tel que tout point du bloc B ait ses coordonnées entières dans ce repère.

La description de B est donnée par la liste à deux niveaux L, dite liste d'intérieur.

$$L = (L0, L1, \ldots, Lk, \ldots, Ln)$$

Lk est une sous-liste qui décrit la colonne d'indice k (points de Ω_H d'abscisse k dans le repère R). Cette sous-liste comprend :

- les ordonnées dans R des points caractéristiques, c'est-à-dire les points situés sur la frontière ou sur une séparatrice.
- les doublets décrivant des intervalles de points de $\mathring{\Omega}_H$.

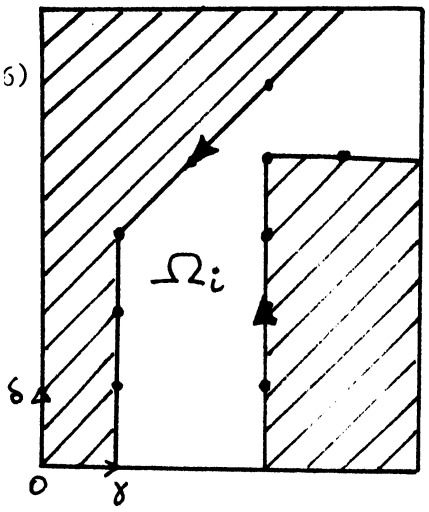
Exemple : Pour le domaine de la figure ci-dessous, la carte contient :

(4, 6, f, 3) $5^3 6^3$ et (3, 0, f, 2) $2^4 0^2$

La liste d'intérieur sera

(, $\bar{0}$ $\bar{1}$ $\bar{2}$ $\bar{3}$, 0-$\bar{4}$, $\bar{0}$ $\bar{1}$ $\bar{2}$ $\bar{3}$ $\bar{4}$ $\bar{5}$, $\bar{4}$-$\bar{6}$, $\bar{4}$-6)

Pour obtenir la liste correspondant à la description de l'intérieur Ω_H, on va considérer successivement chaque chaîne z qui se trouve dans la carte C considérée.



3.1.b. Liste auxiliaire d'intérieur. Algorithme ALG.LO.

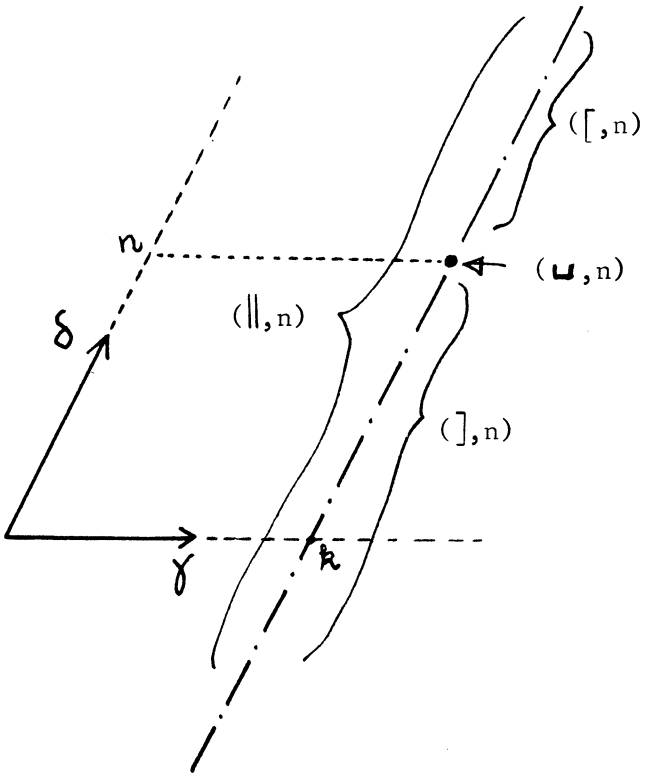
Soit C une carte. L'algorithme ALG.LO sert à décrire l'intérieur du domaine relatif à cette carte au moyen de droites, de demi-droites et de points. ALG.LO examine les chaînes figurant dans la carte considérée et construit une liste auxiliaire d'intérieur.

LA = (LA0, LA1, ..., LAk, ..., LAp)

La sous-liste LAk est une succession de doublets. Chaque doublet peut être :

- ([,n) symbolisant une demi-droite d'origine le point (k,n)
- (],n) symbolisant une demi-droite d'extrémité le point (k,n)
- (||,n) symbolisant une droite contenant le point (k,n)
- (ω,n) symbolisant le point (k,n).

(l'indice n peut être surligné. Il signifie alors que le point (k,n) est sur la frontière ou sur une séparatrice et non sur l'interface.



ALG.LO : Soit une chaîne

$z = x_1 x_2 \ldots x_{r-1} x_r \ldots x_n$

de la carte C.

M_r est l'extrémité de la chaîne $x_1 x_2 \ldots x_{r-1}$ et a pour coordonnées dans R γ_r et δ_r. Au point M_r, on fait correspondre un doublet.

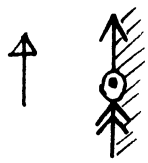
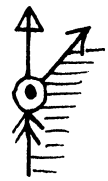
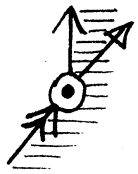
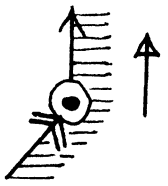
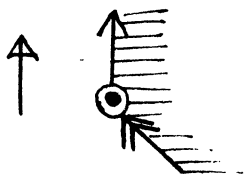
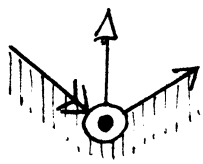
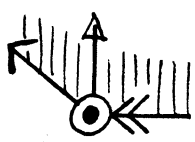
Si z est de type s

on enregistre la valeur $(\|, \delta_r)$ dans la $\gamma_r^{\text{ième}}$ sous-liste de LA.

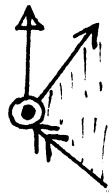
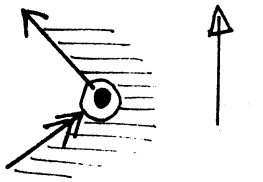
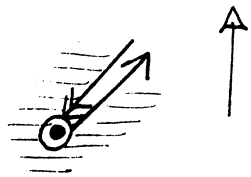
Si z est de type f

on compare les orientations O_r et O'_{r-1} (lorsqu'elles existent) définies par (x_r, δ) et (x_{r-1}, δ) avec l'orientation standard θ définie par (0,1) (vecteurs de base du système V_8).

Les différents cas sont traités dans le tableau suivant. (On se souviendra que l'intérieur du domaine se trouve sur la gauche d'un mobile parcourant la frontière orientée).

Eventualités	Insertion dans la $\gamma_r^{ième}$ sous-liste	Cas de figure
e.1 δ, x_r, x_{r-1} colinéaires	$(\omega, \bar{\delta}_r)$	
e.2 δ non colinéaire à x_r (et colinéaire à x_{r-1}) θ_r existe		
e.2.1 $\theta = \theta_r$	$([, \bar{\delta}_r)$	
e.2.2 $\theta \neq \theta_r$	$(], \bar{\delta}_r)$	
e.3 δ non colinéaire à x_{r-1} (et colinéaire à x_r) θ_{r-1} existe		
e.3.1 $\theta = \theta_{r-1}$	$([, \bar{\delta}_r)$	
e.3.2 $\theta \neq \theta_{r-1}$	$(], \bar{\delta}_r)$	
e.4 δ colinéaire ni à x_r ni à x_{r-1} θ_r et θ_{r-1} existent.		
e.4.1 $\theta = \theta_r$ et $\theta = \theta_{r-1}$	$([, \bar{\delta}_r)$	
e.4.2 $\theta \neq \theta_r$ et $\theta \neq \theta_{r-1}$	$(], \bar{\delta}_r)$	

e.4.3

(a) Si x_r et x_{r-1} sont non colinéaires, ils déterminent une orientation θ'_r si $\theta = \theta'_r$	$(\parallel, \delta_r)$	
si $\theta \neq \theta'_r$	$(\sqcup, \delta_r)$	
(b) x_r et x_{r-1} colinéaires	$(\sqcup, \bar{\delta}_r)$	

Dans ce tableau, les conventions sont les suivantes :

- ↟ direction de balayage δ.
- ↑ vecteur x_r
- ⇑ vecteur x_{r-1}
- ⊙ point M_r.

On incrémente alors r (ou l'on passe à la chaîne suivante) en mettant à jour les coordonnées du point courant de la frontière ou de la séparatrice.

On va appliquer (voir ce qui suit) cet algorithme à des chaînes fermées ou à des spératrices. Il n'y aura donc pas de cas particuliers à envisager en début ou fin de chaîne.

Lorsque l'on a épuisé les chaînes de la carte considérée, on a obtenu la liste auxiliaire. Il suffit alors d'étudier les sections qui y figurent, pour déterminer les intervalles.

(Important)

En général, pour une carte donnée, les chaînes z_j qui figurent dans cette carte ne permettent pas de donner une description de l'intérieur $\mathring{\Omega}_H$, comme le montrent les deux exemples suivants.

Exemple 1 :

C : (2, 3, f, 1) 6^3

L'algorithme précédent conduit à la liste

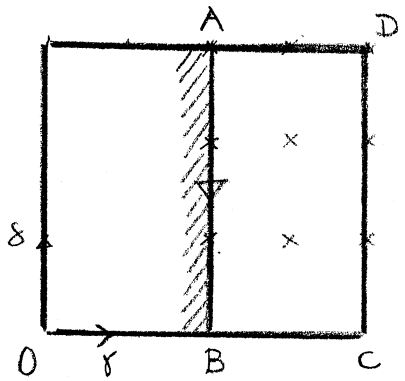
(, , $\bar{0}$ $\bar{1}$ $\bar{2}$ $\bar{3}$, ,)

Il est nécessaire de compléter la chaîne en

$6^3 0^2 2^3 4^2$ qui décrit alors le rectangle ABCD.

On aura cette fois

L = (, , $\bar{0}$ $\bar{1}$ $\bar{2}$ $\bar{3}$, 0-3, 0-3)



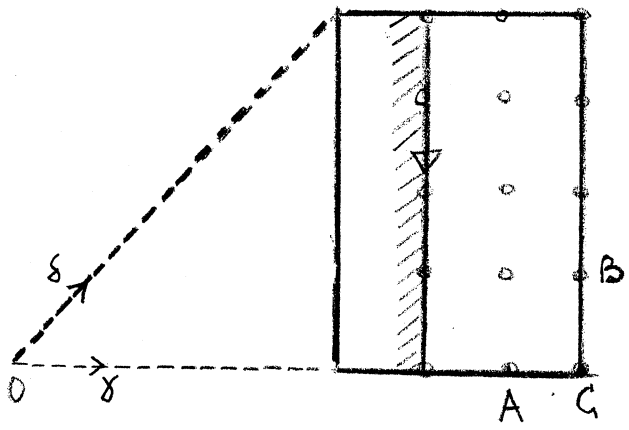
Exemple 2 :

C : (1, 4, f, 1) 6^4

La description suivant la diagonale δ est :

L = (, $\bar{4}$,$\bar{3}$ $\bar{4}$, $\bar{2}$-$\bar{4}$, $\bar{1}$-$\bar{3}$, $\bar{0}$-$\bar{2}$, ,)

On constate que les points A, B, C ne sont pas décrits.

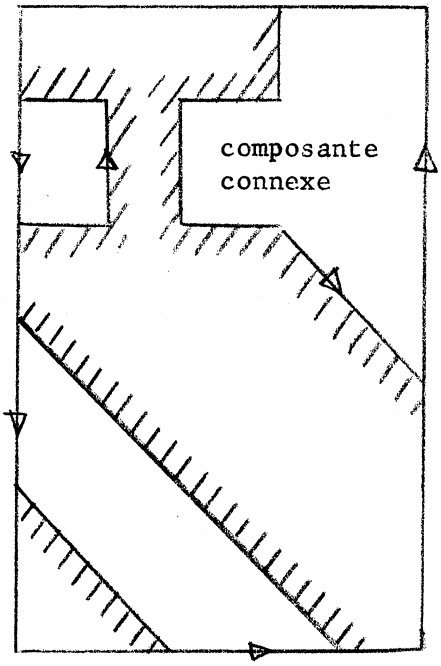


Il est nécessaire, avant de rechercher la liste auxiliaire associée d'une carte, de compléter les chaînes décrivant la frontière.

3.1.c. Recherche des composantes connexes. Algorithme ALG.L1

On fournit à cet algorithme les chaînes z_1, z_2 ..., z_i qui décrivent les frontières d'une carte (celles de type f). Cet algorithme délivre autant de chaînes $\hat{z}_1$ ... $\hat{z}_j$ qu'il y a de composantes connexes (*) dans Ω.

(*) En fait, les composantes connexes qui ont une frontière commune avec l'interface. Voir l'exemple qui suit.



$\hat{z}_k$ est le contour fermé convenablement orienté de la $k^{ième}$ composante connexe.

On est donc assuré que tout point de $\mathring{\Omega}_H$ sera explicitement ou implicitement décrit lorsque l'on transmettra à l'algorithme ALG.LO les chaînes $\hat{z}_1 \ldots \hat{z}_j$. Cette propriété résulte du fait que pour tout point M de $\mathring{\Omega}_H$ et pour toute direction de discrétisation δ, il existe deux points M' et M" situés sur une chaîne $\hat{z}_i$ avec

- M'M" parallèle à δ.
- M, M', M" alignés et M à l'intérieur du segment MM'.

Algorithme ALG.L1

Les données constituent la table suivante (i est l'indice).

1				
2				
i				
	ad(i)	ar(i)	b(i)	z(i) ième chaîne de la carte

où ad(i) est l'abscisse curviligne du départ de la chaîne z repérée sur l'interface.

ar(i) est l'abscisse curviligne de l'arrivée de la chaîne z repérée sur l'interface.

b(i) est le bit de traitement et vaut 0 si la chaîne z(i) ne fait pas déjà partie d'une composante connexe, 1 sinon.

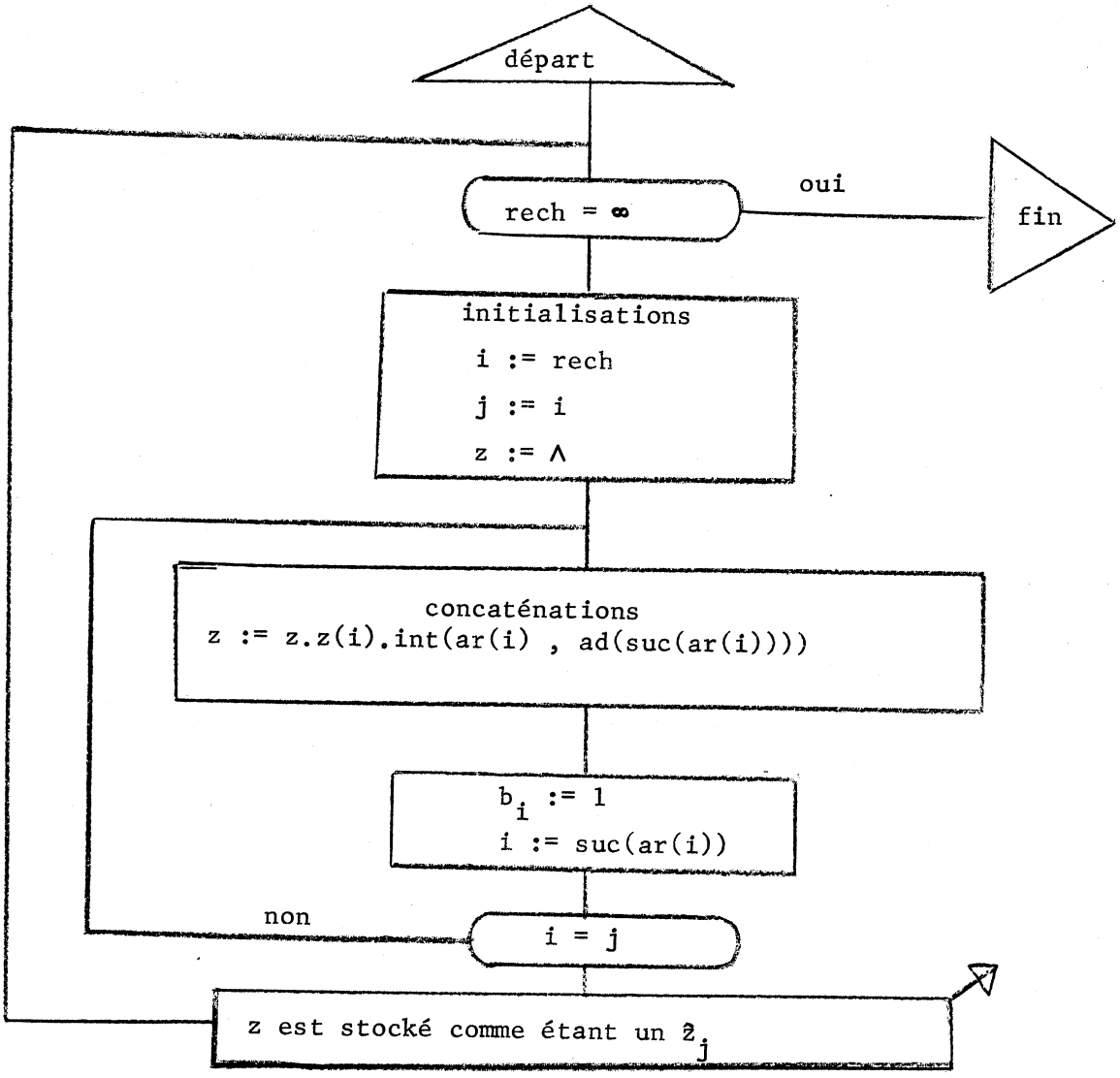
Nous supposons de plus que les fonctions auxiliaires suivantes sont construites.

. fonction suc(t) où t désigne une abscisse d'arrivée d'une chaîne de la carte. Cette fonction délivre l'indice dans la table de la première abscisse de départ d'une chaîne non traitée que l'on rencontre en parcourant l'interface en partant de labscisse t.

. fonction int(u,v) où u et v sont deux abscisses (curvilignes)de l'interface. Cette fonction délivre une chaîne qui est le code de la portion d'interface comprise entre les abscisses u et v.

. fonction rech sans argument. Elle délivre l'indice de la première chaîne non traitée en partant du haut de la table. Si toutes les chaînes sont traitées rech = ∞.

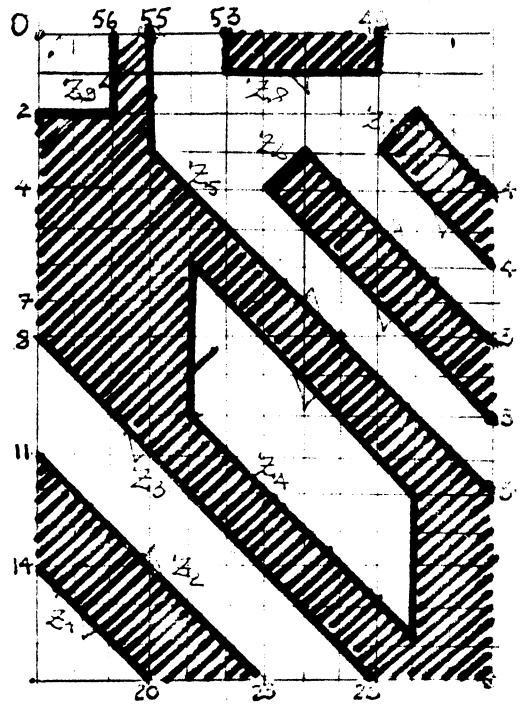
L'organigramme de ALG.LO est alors :



Exemple

La table de la carte correspondant à la figure ci-dessous est :

20	14	0	3^3
11	23	0	7^6
26	8	0	3^9
55	34	0	$6^3 7^9$
36	38	0	$3^6 1\ 7^5$
40	42	0	$3^3 1\ 7^2$
49	53	0	$6\ 4^4\ 2$
2	56	0	$0^2 2^2$



La constitution des composantes connexes se fait comme suit :

$z := \Lambda$

$z := z_1.\text{int}(14,20) = z_1 6^3 0^3 = \hat{z}_1$

$z := \Lambda$

$z := z_2.\text{int}(23,26) = z_2 0^3$

$z := z_2 0^3.z_3.\text{int}(8,11) = z_2 0^3 z_3 6^3 = \hat{z}_2$

$z := \Lambda$

$z := z_5.\text{int}(34,36) = z_5 2^2$

$z := z_5 2^2.z_6.\text{int}(38,40) = z_5 2^2 z_6 2^2$

$z := z_5 2^2 z_6 2^2.z_7.\text{int}(42,49) = z_5 2^2 z_6 2^2 z_7 2^4 4^3$

$z := z_5 2^2 z_6 2^2 z_7 2^4 4^3.z_8.\text{int}(53,55) = z_5 2^2 z_6 2^2 z_7 2^4 4^3 z_8 4^2 = \hat{z}_3$

$z := \Lambda$

$z := z_9.\text{int}(52,2) = z_9 4^2 6^2 = \hat{z}_4$

On prendra de plus $z_4 = \hat{z}_5$.

3.1.d. Algorithme ALG.L2.

Pour une carte C, l'algorithme ALG.L1 fournit les chaînes fermées qui sont les codes des frontières des composantes connexes du domaine relatif à C.

L'algorithme ALG.L0 appliqué à l'ensemble de ces chaînes fournit une liste auxiliaire

LA = (LA1, LA2, ..., LAk, ..., LAp)

avec

$LA_k = d_1\ d_2 \ldots d_i \ldots d_{n_k}$ suite ordonnée de doublets (Pour comparer les doublets, il suffit de comparer leurs deuxièmes composantes qui sont des entiers).

d_i = (␣, $\bar{n}_i$) ou ([, $\bar{n}_i$) ou (], $\bar{n}_i$) ou (‖, $\bar{n}_i$).

L'algorithme ALG.L2 transforme cette liste auxiliaire en la liste d'intérieur. Il travaille sur chaque sous-liste LAk en opérant sur deux doublets consécutifs. Il y a création d'un intervalle (représenté par un tiret) si deux doublets consécutifs sont compatibles.

La liste d'intérieur formée est alors :

L = (L0, L1, ..., Lk, ..., Lp) avec

$Lk = pr_2(d_1)s_1\ pr_2(d_2) \ldots pr_2(d_i)s_i\ pr_2(d_{i+1}) \ldots pr_2(d_{n_k})$

où $pr_2(d_i) = \bar{n}_i$

et s_i est donné par la table de correspondance donnant les seuls cas possibles ($pr_1(d_i)$ est la première composante du doublet d_i).

$pr_1(d_i)$	$pr_1(d_{i+1})$	résultat s_i
[	] ou ␣ ou ‖	-
]	[ou ␣ ou ‖	␣
‖	] ou ␣ ou ‖	-
␣	] ou ‖	-
	␣ ou [	␣

Exemple : Pour la figure du bas, la carte est :

$z_1 = (2, 4, 7)\ 2^3 0^3 13134^4 2^3 0^7 6^{10} 4^7$

$z_2 = (12, 17, f)\ 4^{12}$

$z_3 = (1, 9, s)\ 0^3$

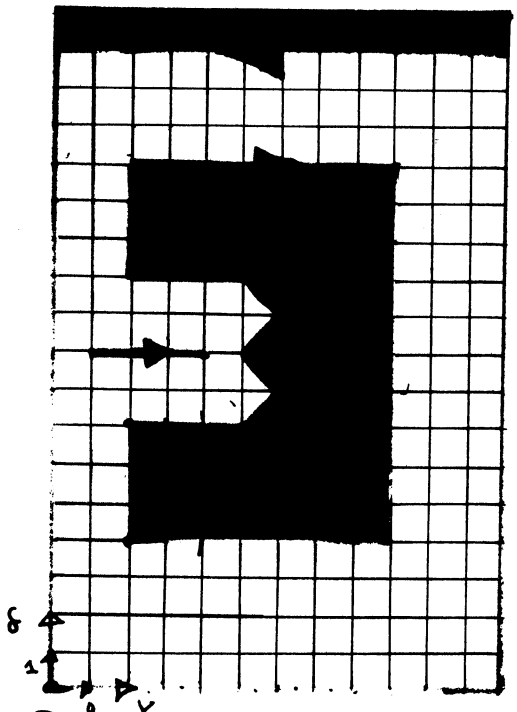
L'algorithme ALG.L1 transforme z_2 en

$\hat{z}_2 = (12, 17, f)\ 4^{12} 6^{17} 0^{12} 2^{17}$

Les sous-listes d'indice 2, 4 et 6 des listes LA et L sont données par les tableaux :

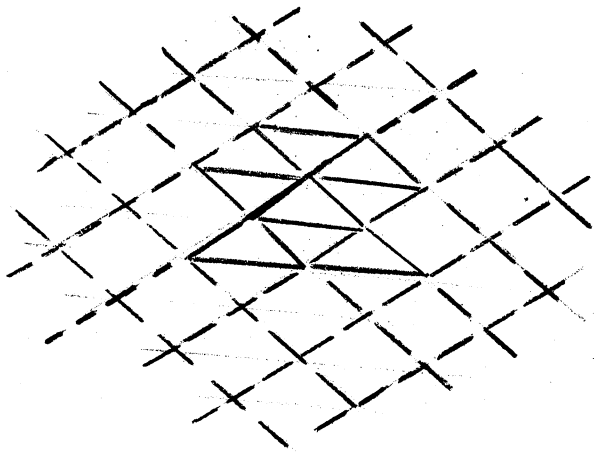
LA2	([,0) (],$\bar{4}$) (␣,$\bar{5}$) (␣,$\bar{6}$) ([,$\bar{7}$) (‖,$\bar{9}$) (],$\overline{11}$) (␣,$\overline{12}$) (␣,$\overline{13}$) ([,$\overline{14}$) (],$\overline{17}$)
LA4	([,0) (],$\bar{4}$) ([,$\bar{7}$) (‖,$\bar{9}$) (],$\overline{11}$) ([,$\overline{14}$) (],$\overline{17}$)
LA6	([,0) (],$\bar{4}$) (␣,$\bar{8}$) (␣,$\overline{10}$) ([,$\overline{14}$) (],$\overline{17}$)

L2	0-$\bar{4}$ $\bar{5}$ $\bar{6}$ $\bar{7}$-$\bar{9}$-$\overline{11}$ $\overline{12}$ $\overline{13}$ $\overline{14}$-$\overline{17}$
L4	0-$\bar{4}$ $\bar{7}$-$\bar{9}$-$\overline{11}$ $\overline{14}$-$\overline{17}$
L6	0-$\bar{4}$ $\bar{8}$ $\overline{10}$ $\overline{14}$-$\overline{17}$

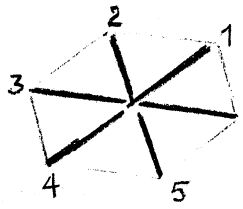


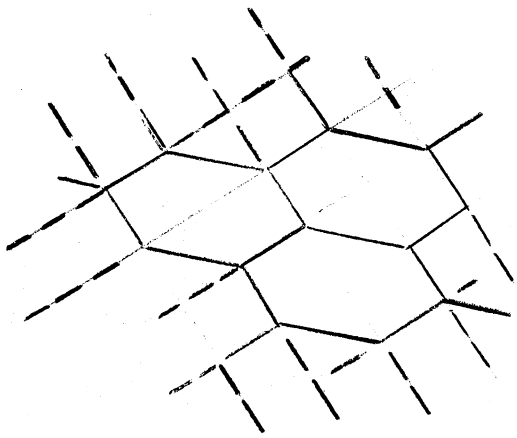
3.2. CHANGEMENT D'ALPHABET.

Dans la pratique, il arrive que l'on utilise d'autres grilles que des grilles à maille rectangulaire. Afin d'ajuster une frontière suivant une ligne de la grille ou de faire choix d'une direction de balayage plus favorable, on se sert de mailles formées de triangles ou d'hexagones.



Pour de telles grilles, on peut utiliser le vocabulaire V_6 .





Le cas des hexagones se ramène au cas précédent. Le lecteur trouvera de telles applications dans [Col]

3.3. CHANGEMENT DE RESOLUTION.

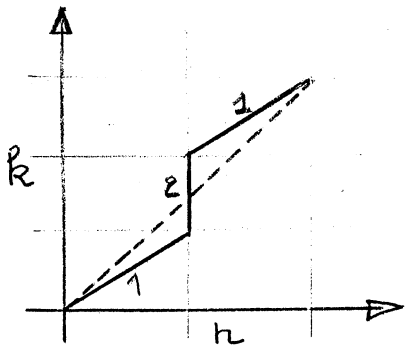
Pour une carte donnée C d'un domaine Ω, nous avons les pas de discrétisation h et k.

Considérons les nouveaux pas de discrétisation $h' = \frac{h}{n}$ et $k' = \frac{k}{p}$. En général, chaque vecteur x d'une chaîne z de C sera remplacé par une chaîne $x_1 x_2 \ldots x_{sup(n,p)}$

Exemple : n = 2 et p = 3

1 devient 1.2.1

Dans le cas particulier où p = n x sera remplacé par x^n.



Il résulte que, dans le procédé de description de l'intérieur $\overset{\circ}{\Omega}_H$, on obtient très facilement les chaînes des nouvelles composantes connexes. Il suffit alors de recomposer les sous-listes dans la liste d'intérieur correspondant aux nouveaux points ainsi introduits.

En réalité, ceci se produira dans le cas où les frontières du domaine sont simples. Dans les autres cas, le procédé de dédoublement ci-dessus pourra introduire des erreurs notables. Il sera nécessaire de reprendre à son début la construction de la liste d'intérieur, du moins dans chaque composante connexe où de telles frontières existent.

§.4 PROBLEMES D'IMPLEMENTATION.

Pour un domaine Ω, nous avons fait choix d'une direction de balayage d'une grille et nous avons obtenu les listes d'intérieur correspondant à chaque carte.

A tout point P de Ω_H, on associe en vue du traitement numérique, une équation linéaire aux inconnues (au plus) :
U(P), U(NP), U(EP), U(OP), U(SP). (N pour nord, S pour sud, etc...).

$$\alpha_0 U(P) = \alpha_1 U(NP) + \alpha_2 U(SP) + \alpha_3 U(EP) + \alpha_4 U(OP) + \beta$$

Par conséquent, on doit disposer en mémoire d'une table de valeurs où seront stockées les quantités U(P) pour P appartenant à Ω_H.

Une relaxation au point P qui consiste à effectuer comme nouvelle valeur U(P) la quantité

$\alpha_1 U(NP)_A + \ldots + \alpha_A U(OP)_A + \beta$ (l'indice A pour ancien) nécessite la connaissance des quatre adresses de mémoire où sont enregistrés U(NP), U(SP), U(EP) et U(OP) et des cinq coefficients de l'équation. En double précision et pour un ensemble $\overset{\circ}{\Omega}_H$ de 10000 points, la capacité de mémoire nécessaire à l'enregistrement de ces données numériques peut atteindre le demi-million d'octets.

Ceci rend nécessaire l'emploi d'un système de pagination de la mémoire secondaire qui chargera, à un instant donné, les pages (zones de mémoire secondaire) qui seront utiles au calcul, à cet instant. On peut tirer parti du fait que pour tous les points d'une carte, excepté pour ceux de la bordure, le calcul ne nécessite que des adresses de mémoire relatives à des points de cette carte. Il suffit de donner une définition de page qui soit en correspondance avec la définition d'une carte.

La figure qui suit illustre cette correspondance. Le déplacement d_p, qui donne l'adresse relative (repérée par rapport à l'adresse de la page) de la mémoire qui contient la valeur U(P) est obtenu à partir de la liste d'intérieur. d_p est la valeur d'un compteur que l'on incrémente chaque fois que l'on trouve un point décrit dans la liste lorsque ce point précède le point P.

Cette fonction d_p réalise l'indexation des points de Ω_H. Il est alors aisé de résoudre les problèmes graphiques liés à la méthode des différences finies par un programme (qui joue le rôle d'une boussole).

Les adresses des voisins de P sont :

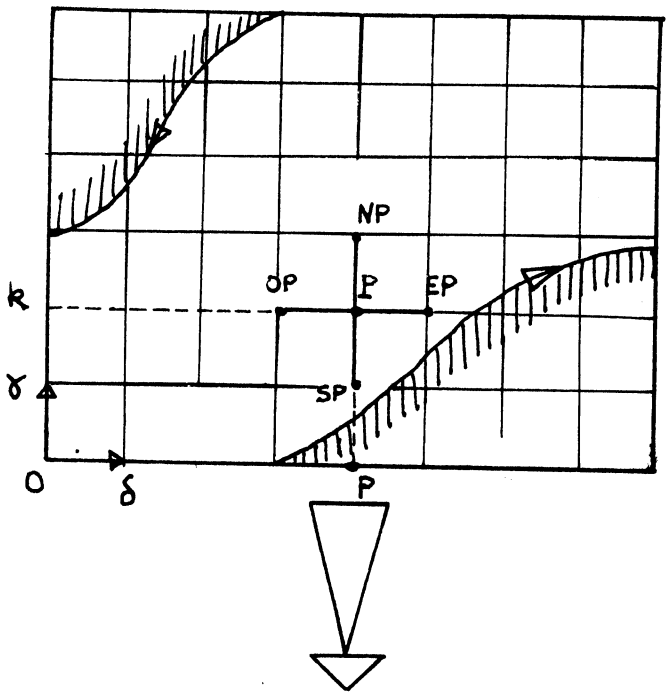
d_p - d pour OP

d_p + d pour EP

Celles de U(NP) et U(SP) sont calculées par le programme boussole, en inspectant la sous-liste qui suit et celle qui précède la sous-liste où est décrit P. (voir le schéma page suivante).

. SCHEMA D'ORGANISATION DE LA MEMOIRE .

Carte C (discrétisation par ligne)

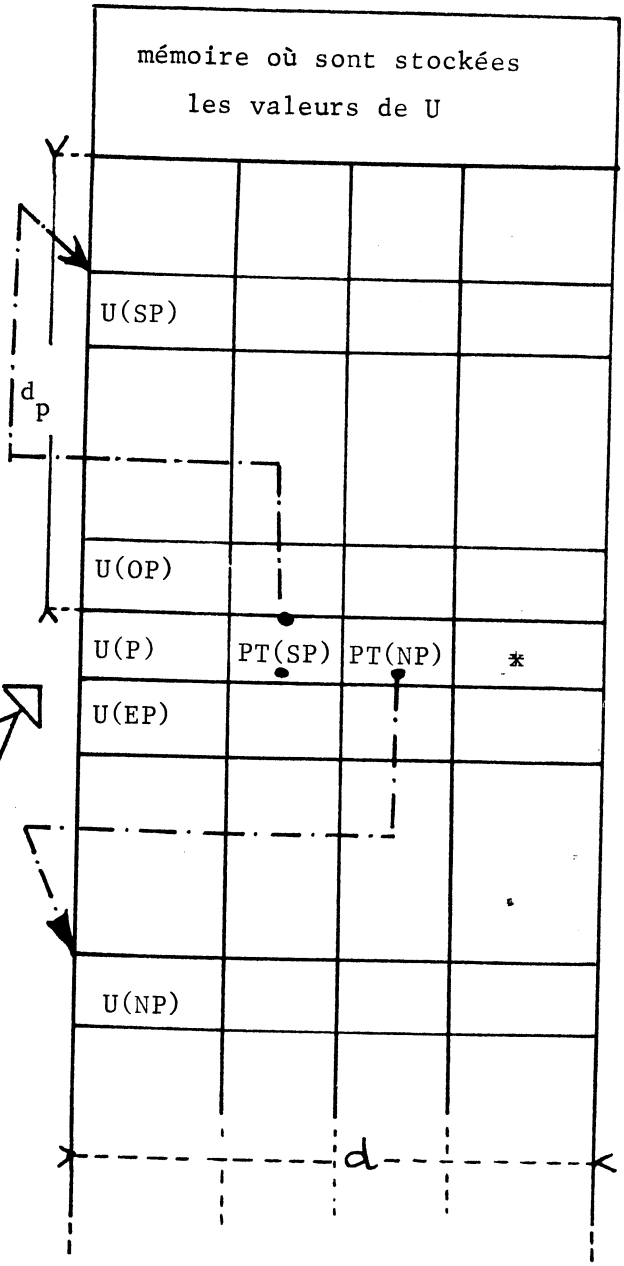


Liste d'intérieur

L = (, , ..., Lk, ...)

$L_k = 0 - \overline{P + 1}$

page



Programme boussole
calcul de d_p et des pointeurs

PT(NP)
PT(SP)

* Ces octets servent à l'enregistrement des coefficients α_i et β et permettent de spécifier des caractéristiques particulières du point considéré (appartenance à une frontière, indice de cette frontière, etc...).

IIIème PARTIE

§ 1. TERMINAL GRAPHIQUE.

1.a. INTRODUCTION.

Quand on étudie les mécanismes de discrétisation des courbes et des domaines à partir de leurs frontières au moyen d'un terminal graphique, on s'aperçoit qu'il est aisé d'associer à une suite de points P_1, P_2, ..., P_n déterminés en machine par les valeurs de leurs coordonnées respectives un programme graphique qui va générer à l'écran un arc passant par ces points P_1, P_2, ..., P_n. Cette utilisation du terminal graphique en traceur de courbes est décrite dans [Cag].

Supposons maintenant que l'on ne dispose plus en calculatrice de l'information numérique relative aux points P_1, P_2, ..., P_n et que l'on désire programmer l'algorithme décrit en I.5.a . Il s'agit cette fois d'associer à une représentation externe d'une courbe (par exemple celle décrite par l'extrémité du crayon optique que manipule un opérateur) une information binaire en mémoire qui est le code de cette courbe. Cette information binaire est constituée par le programme graphique qui génère à l'écran du terminal la représentation de la discrétisation de la courbe. Les programmes de poursuite du crayon optique usuels fournissent comme discrétisation d'un arc une ligne brisée aléatoire. Les programmes qui suivent satisfont à l'hypothèse de régularité sur les frontières des domaines faite en II.2.1 .

Les principes des algorithmes, dont il est question dans les deux premières parties, sont simples. Par contre, la multiplicité des spécifications et restrictions imposées par les constructeurs de périphériques graphiques font que, quelque soit le langage utilisé (langage assembleur, FØRTRAN, PL1, etc...), les programmes deviennent très rapidement conséquents et compliqués. (CLEEMAN dans [Cle] en donne un exemple). Cette constatation a été à l'origine de la conception de langages et de systèmes plus spécifiquement orientés vers l'utilisation de tels terminaux. (Voir [Lec] et [Cle]). Pour exposer à moindre frais la programmation des algorithmes étudiés, nous allons imaginer un terminal graphique TG qui a les possibilités d'un terminal graphique évolué mais dont le jeu d'instructions de base est volontairement réduit et qui a des caractéristiques uniformisées.

(Nous déclinons toute responsabilité en cas d'un vice de forme intervenant dans sa réalisation !). Le lecteur trouvera la description détaillée d'un terminal graphique réel tel que l'I.B.M. 2250 modèle 1 dans les notices du constructeur [Ibm 1], [Ibm 2]. A la fin de cette partie, il trouvera la version actuelle des programmes correspondant à cette machine reliée à un IBM 360-67 fonctionnant avec le système CP-CMS.

1.b. LE TERMINAL GRAPHIQUE TG.

1.b.1. Description physique et logique.

Nous supposons que le terminal graphique TG est relié à l'unité centrale d'une calculatrice par un canal C. Ce canal permet l'échange d'information binaire dans les deux sens. (Schéma b1). Le schéma b2 montre les différents constituants de TG. (Ces constituants donnent à TG une structure de calculatrice que l'on pourrait faire fonctionner de façon autonome.

MTG est la mémoire propre à TG. Dans cette mémoire se trouve un programme graphique lorsque TG est en état de marche. MTG est un support de fichier sur lequel l'unité centrale peut lire et écrire par l'intermédiaire du canal C. On désigne par um l'unité élémentaire de mémoire formée de bits. Cette unité correspond à la longueur standard des instructions du programme graphique.

BTG est une boîte qui contient en particulier un compteur ordinal symbolisé par la flèche ①, des dispositifs logiques qui permettent de décoder l'instruction sur laquelle pointe la flèche ① et qui génèrent les ordres de déflection du spot lumineux, ce que nous représentons par la flèche ② . D'autre part, cette boîte réalise l'interface entre l'unité centrale et le terminal graphique.

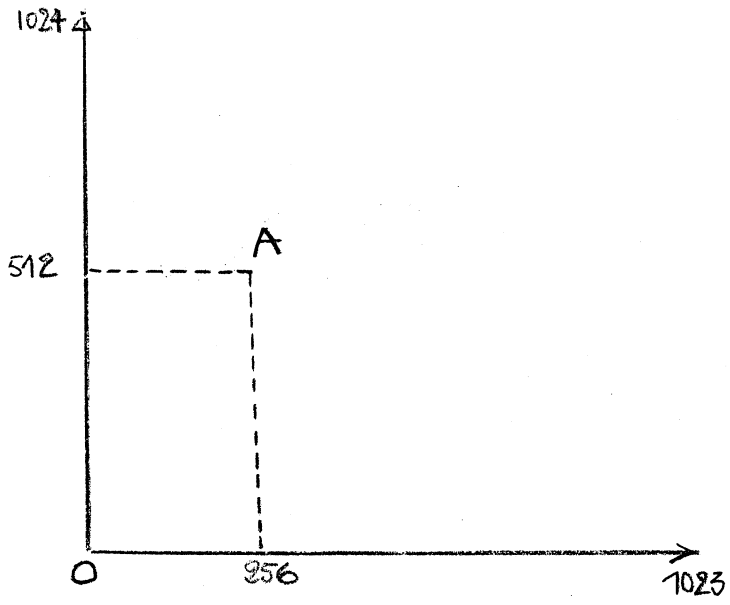
CR est le crayon optique. Ce dispositif (à cellule photo électrique) est capable de détecter l'occurrence d'un rayon lumineux sur l'écran lorsque celui-ci est en coïncidence avec l'extrémité du crayon et que la dernière instruction rencontrée par le compteur ordinal dans MTG est une instruction de mise en fonctionnement de celui-ci.(*). Lorsque ces éventualités se produisent, une interruption intervient et deux informations sont envoyées à l'unité centrale via le canal (flèche (3)). Avec ces informations on dispose :

(I_n) . des coordonnées du point de l'écran où a été réalisée la coïncidence entre le rayon lumineux et le crayon optique.

(I_A) . de l'adresse de l'instruction qui a provoqué la déflection du rayon lumineux en ce point.

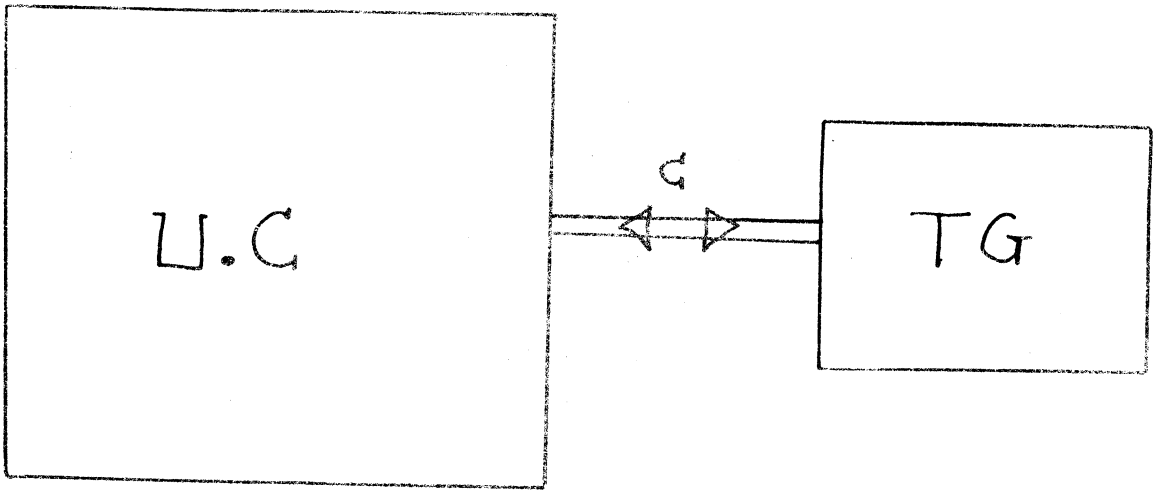
E est l'écran. Un point de l'écran a ses coordonnées prises dans $\mathbb{Z}/_{1024} \times \mathbb{Z}/_{1024}$.

En particulier (256, 512) et (1280, 1536)
désignent le même point A.

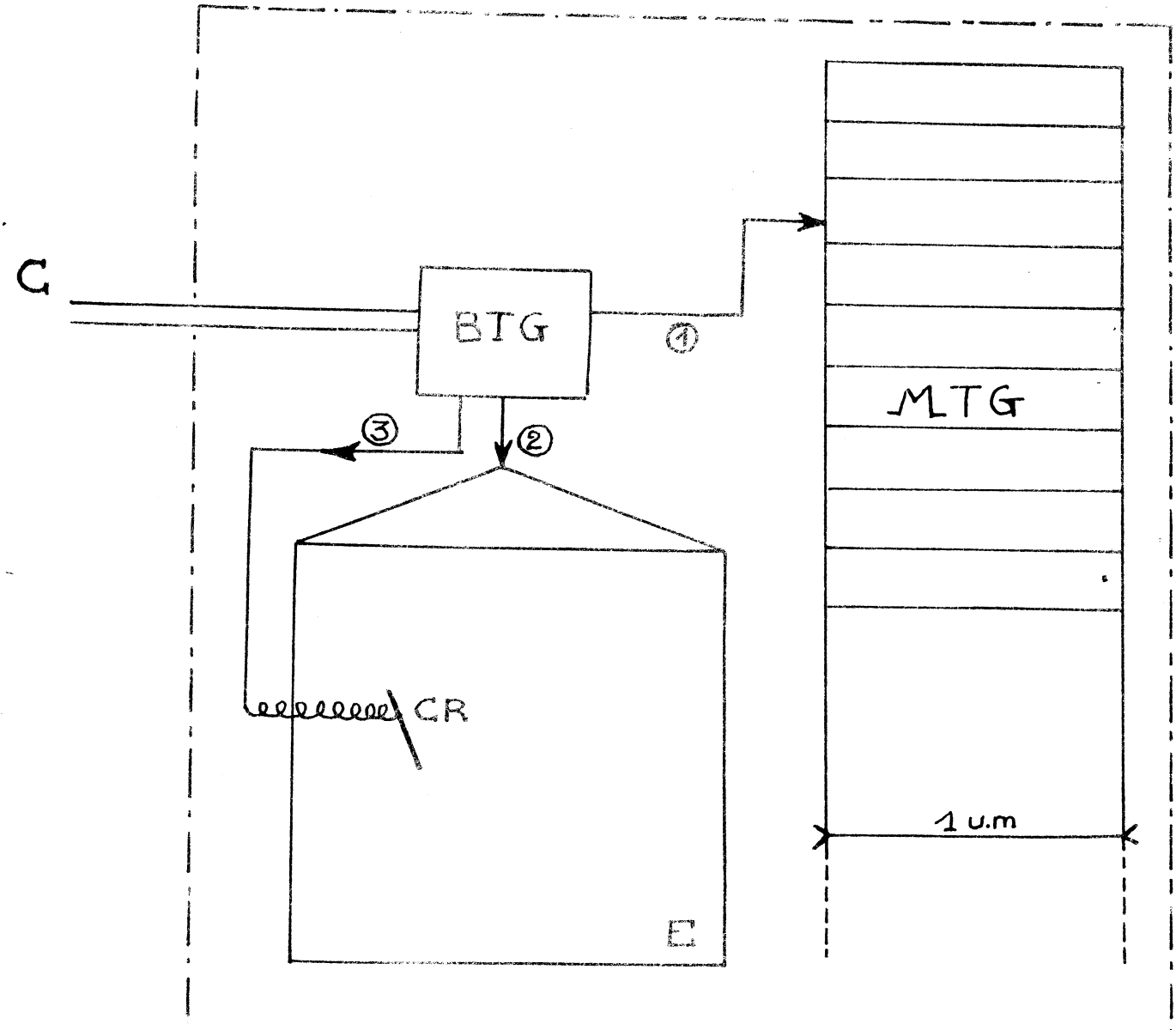


(*) Il s'agit de la dernière instruction rencontrée qui concerne le crayon optique.

shéma b1.



shéma b2.



1.b.2. Le jeu d'instructions.

Le programme graphique est une liste d'instructions prises parmi les six qui suivent.

dav x_i, y_i, b_i — déplacement absolu du rayon lumineux de la position courante à la position (x_i, y_i). Ce déplacement est lumineux si $b_i = 0$ (non lumineux si $b_i = 1$) et produit un vecteur.

drv x_i, y_i, b_i — a le même effet que dav mais l'extrémité du vecteur (x_i, y_i) est repérée dans un repère relatif lié à la position courante.

dap x_i, y_i, b_i
drp x_i, y_i, b_i — positionnent le point courant dans le repère absolu ou relatif et permettent de dessiner des points.

tr n_i — transfert inconditionnel à l'adresse n_i de MTG (et mise à jour cohérente du compteur ordinal).

acr et nacr — instructions qui permettent d'activer ou de déconnecter le crayon optique.

1.b.3. Le point de vue du programmeur.

Le programmeur agit sur le fonctionnement du terminal graphique par un programme qui est chargé à l'exécution en mémoire centrale U.C. Ce programme (programme principal P_p) a pour effet

- d'initialiser le programme graphique (la version initiale du programme graphique se trouve dans M.C et est recopiée dans MTG)
- de gérer l'interruption due au crayon optique. En se servant en particulier de (I_n) ou (I_A), il peut modifier le programme graphique.
- de commander l'arrêt ou le départ du fonctionnement de TG.

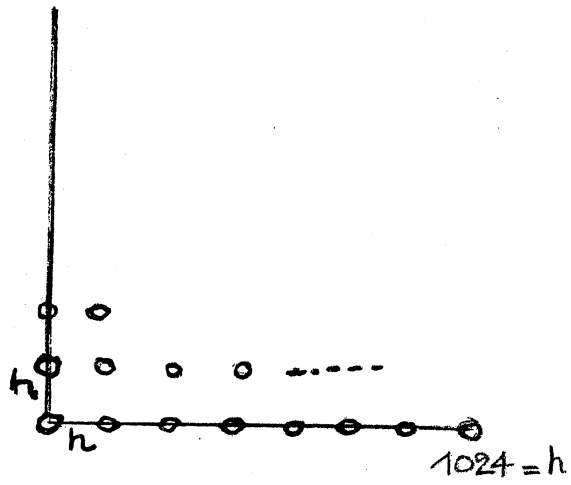
§ 2. PROGRAMMES.

2.a. MEMORISATION D'UN TRACE LINEAIRE PAR MORCEAUX.

2.a.1. Génération de la grille.

Le programme graphique de génération de la grille peut être

```
      acr
      dav   0, 0, 0
      drv   h, 0, 0
n     drv   h, 0, 0
      --------
      -------
      drv   0, h, 0
      tr    2
```



(On tire parti du fait que les coordonnées sont prises modulo 1024 sur les deux axes).

2.a.2. Génération du tracé linéaire par morceaux.

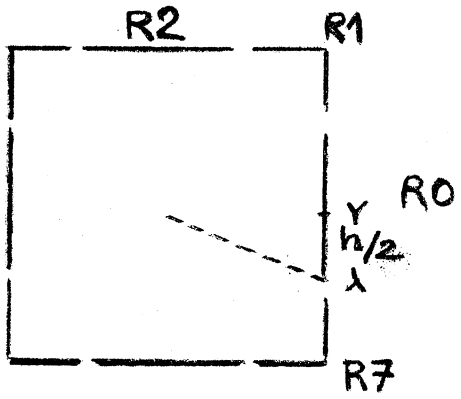
A chaque fois que le crayon optique détecte un point (ceci est possible grâce à l'instruction acr), le programme P_p construit une instruction dav x_I^p , y_I^p , 0 où x_I^p et y_I^p sont les coordonnées du point détecté. (p signifie que l'on en est à la $p^{ième}$ détection)et l'envoie à l'adresse n + 3 + (p-1). L'état de la mémoire MTG à ce stade est donné par le tableau :
Pour visualiser le tracé linéaire, il suffit alors de mettre une instruction tr n+3 à l'adresse n + 3 + p et de lancer l'exécution de TG à l'adresse n+3.

0	acr	
	dav	0,0
	drv	h,0,0
	⋮	
	drv	0,h,0
n + 3	tr	2
	dav	x_I^1, y_I^1, 0
	dav	x_I^2, y_I^2, 0
	⋮	
	dav	x_I^p, y_I^p, 0
	tr	n + 3

2.b. MEMORISATION D'UN TRACE DISCRETISE.

2.b.1. Génération de la figure test.

Pour V_8 donnons la séquence d'instructions qui génèrent la figure test correspondante. (le point courant est supposé au centre).




```
I0      acr
I1      arp   h, -h/2, 0
        drv   0, h, 0          région 0
        drv   0, h/2, 0
        drp   -h/2, 0, 0       région 1
        :
        drp   h/2, 0, 0
        drp   0, h/2, 0        région 7
I17     nacr
        tr    0
```

2.b.2. Génération et affichage du tracé.

Lorsqu'une interruption due au crayon optique se produit, le programme P_p calcule l'indice de la région de la figure test où le crayon optique a traversé la figure test. Ceci se fait en calculant la division entière

$$i = (Ad_I - 1) : 2$$

Il suffit alors d'aller chercher dans la table des vecteurs incrémentaux du vocabulaire V_8

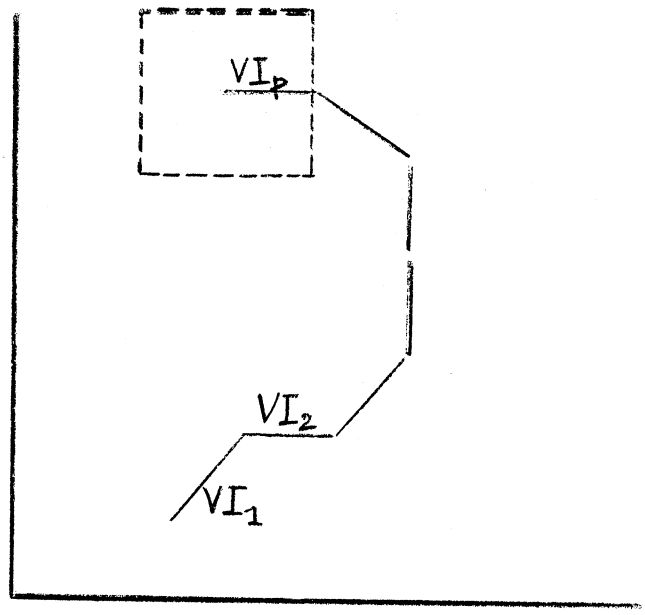
```
VI0     drv     h, 0, 0
VI1     drv     h, h, 0
VI2     drv     0, h, 0
        :
VI7     drv     h, -h, 0
```

le vecteur VI_i.

On substitue alors l'instruction qui génère ce vecteur VI_i à la dernière instruction tr 0 (cette instruction est replacée à l'adresse suivante).

A la $p^{ième}$ interruption, l'état de la mémoire et de l'écran est donné par les figures suivantes.

acr
drp h, -h/2, 0
drv 0, h, 0
⋮
drp 0, h/2, 0
nacr
VI_{i_1}
VI_{i_2}
⋮
VI_{i_p}
tr 0



2.b.3. Modification de la figure test.

On voit sur l'exemple précédent (vocabulaire V_8) que pour changer la figure test, il suffit que le programme P_p

- remplace les instructions I_1, I_2 ... I_{16} par les instructions générant la nouvelle figure test
- change dans la division entière le diviseur (nombre d'instructions générant une région)
- construise la nouvelle table des vecteurs incrémentaux.

Sans interrompre le tracé, on peut donc changer les pas de discrétisation et les vecteurs du système de référence (V_8, V_6, V_{16}, ...).

2.c. COMMANDES DE MODIFICATION DE FIGURES

On peut utiliser le crayon optique pour repérer un arc de courbe afin de le supprimer ou de le modifier. Il suffit de substituer à l'instruction nacr qui est en tête du programme graphique générant cet arc une instruction acr. En pointant le crayon optique sur l'origine et l'extrémité de l'arc on provoque deux interruptions. Le programme qui gère les interruptions enregistre I_A^1 et I_A^2 qui sont les adresses des deux instructions générant l'origine et l'extrémité de l'arc.

- Pour supprimer l'arc il suffit de mettre à 1 le bit de luminosité de toute instruction graphique générant un vecteur et dont l'adresse est comprise entre I_A^1 et I_A^2.

- Pour le modifier, on peut insérer une instruction tr I_ℓ à l'adresse I_A^1. (I_ℓ est l'adresse du début d'une zone libre de la mémoire MTG repérée par un compteur spécial). A partir de l'adresse I_ℓ on construit le programme graphique de l'arc modifié, ce programme se terminant par une instruction tr I_A^2.

2.d. EXEMPLES

Les programmes qui suivent illustrent les paragraphes précédents. On utilise dans ces programmes les possibilités supplémentaires du display 2250 qui sont données par le clavier alphanumérique : (ce clavier permet d'afficher des caractères qui sont transmis à l'unité centrale) et le tableau des fonctions programmées qui est un ensemble de clefs. Chaque clef permet de donner le contrôle à un programme déterminé. (Ce programme peut réaliser la modification de résolution, de figure, etc...).

Le tracé de la photo 1 est obtenu à partir du mouvement du crayon optique. En cours de route on a fait varier la résolution et le système de vecteurs de référence (V_3,V_8,V_{16}). On trouve sur la page suivante le code hexadécimal de la zone de mémoire du terminal graphique où se trouve le programme générant ce tracé obtenu par le programme de poursuite du crayon optique COD.

La figure 2 est un dessin libre exécuté avec le système de vecteurs V_{16}. De nombreuses modifications de figure ont été nécessaires (pour produire un résultat qui ne fasse pas trop honte à son auteur). La longueur du code hexadécimal du programme graphique en témoigne.

```
print 50
*************************************************************************
* PP EST LE PROGRAMME PRINCIPAL QUI REND ACTIFS LES PROGRAMMES DE *
* TRAITEMENT DES INTERRUPTIONS COD PFK ET RETCHAR, IL INITIALISE  *
* LE SYSTEME GOG ET SE PLACE EN MODE ATTENTE JUSQU'A LA FIN D'UNE *
* SESSION DE MANIPULATION DES FIGURES,DE PLUS PP APELLE LES SOUS- *
* PROGRAMMES NECESSAIRES A L'UTILISATION DE GOG SOUS CMS OU OS    *
*************************************************************************
GOG      CSECT
         SAVE  (14,12)
         BALR  2,0
         USING HERE,2,3,4,8,9
HERE     L     3,ADDR3
         L     4,ADDR4
         L     8,ADDR8
         L     9,ADDR9
         B     SVREG
ADDR3    DC    A(HERE+4096)
ADDR4    DC    A(HERE+8192)
ADDR8    DC    A(HERE+12288)
ADDR9    DC    A(HERE+16384)
SVREG    LA    10,ZONE
         ST    10,8(13)
         ST    13,4(10)
         LR    13,10
JUMP     CALL  OSDE2250
         OPEN  (BCDE)
         SPAR  (INT0),PRTY=0
         SPAR  (INT1),PRTY=1
         SPAR  (INT2),PRTY=2
         CALL  GSTOR,(OCBPT,PARTABT)
         CALL  GSTOR,(OCBPG,PARTABG)
         SR    7,7
         LA    6,BIB
         LA    5,INDEX
RE       ST    6,8(5)
         LA    5,16(5)
         LA    6,768(6)
         LA    7,1(7)
         C     7,=F'10'
         BC    4,RE
         GWRITE DECBT,STR,BCDE,OCBPT
         WAIT  ECB=DECBT
         GCNTRL DCPP,IND,BCDE,K0
         WAIT  ECB=DCPP
         WAIT  ECB=COMSAVE1+16
         CLOSE (BCDE)
         DAR   (INT0,INT1,INT2)
         CALL  OSFI2250
         L     13,4(13)
         RETURN (14,12)
```

```
*****************************************************************
* COD EST LE PROGRAMME DE TRAITEMENT DES INTERRUPTIONS DUES *
* AU CRAYON OPTIQUE.                                            *
* IL GENERE LA FIGURE TEST ET L'ARC DEJA TRACE DE FACON DYN *
* AMIQUE.                                                       *
* AU FUR ET A MESURE DU TRACE LE CODE ALGEBRIQUE DU DESSIN  *
* EST STOCKE EN MEMOIRE DU 360.                                 *
*****************************************************************
COD      SAVE  (14,12)
         DROP  2,3,4,8,9
         BALR  6,0
         USING HEREC,6,7,10,12,14
HEREC    L     7,ADDRC7
         L     10,ADDRC10
         L     12,ADDRC12
         L     14,ADDRC14
         B     SVREGC
ADDRC7   DC    A(HEREC+4096)
ADDRC10  DC    A(HEREC+8192)
ADDRC12  DC    A(HEREC+12288)
ADDRC14  DC    A(HEREC+16384)
SVREGC   LA    9,SVAREA
         ST    9,8(13)
         ST    13,4(9)
         LR    13,9
*****************************************************
* INITIALISATION DE R9 ET R11 AVEC LES VALEURS *
* CONTENUES PAR PTBUF ET PTBUF-1                *
*****************************************************
         L     9,PTBUF
         LR    11,9
         S     11,=F'1'
*************************
* VIENT ON DE LA GRILLE *
*************************
         CLI   GRIL,C'O'
         BC    8,COMCON
*************************
* CALCUL DE AIBUF ADRESSE*
* DE L'INST.GRAPHIQUE QUI*
* PROVOQUE L'INTERRUPTION*
*************************
         SR    2,2
         NI    COMSAVE+6,X'1F'
         LH    2,COMSAVE+6
         CH    2,=H'126'
         BC    2,TESTMOD
*********************************
* L'INTERRUPTION EST PROVOQUEE *
* PAR LA FIGURE TEST           *
*********************************
         CLI   GOM,C'O'
         BC    8,ECRIT
         CLI   SUPGOM,C'O'
         BC    8,ECRIT
         CLI   INTER,C'O'
         BC    8,ECRIT
         CLI   TRACE,C'O'
         BC    7,SAUT1
         NI    0(11),X'FE'
```

```
*****************************************
* CALCUL DU NOUVEAU VECTEUR INCREMENTAL *
* ET REMPLISSAGE DE GDOA PAR LE GDV ET  *
* LE GTRU CONVENABLES (VECTEUR BLANC)   *
*****************************************
SAUT1    LR    5,2
         S     5,=F'10'
         SRA   5,1(0)
         SR    4,4
         D     4,DIVDENDE
         ST    5,MEMO
         SLA   5,1(0)
         SR    4,4
         L     4,AVECTEUR
         AR    4,5
         MVC   4(2,9),2(9)
         MVC   2(2,9),0(9)
         MVC   0(2,9),0(4)
**********************************
* INCREMENTATION ET SAUVEGARDE   *
* DE PTBUF ET TRANS+16           *
**********************************
         LA    9,2(9)
         ST    9,PTBUF
         LA    9,4(9)
         ST    9,TRANS+16
         B     ECRIT
****************************
* EST ON EN MODE GOM       *
* EST ON EN MODE SUPERGOM  *
****************************
TESTMOD  CLI   SUPGOM,C'O'
         BC    7,SAUTO
         LA    3,GDOA
         AR    2,3
         XI    PARITE,X'01'
         CLI   PARITE,X'01'
         BC    7,JUMPAR
         ST    2,AINT1
         B     ECRIT
JUMPAR   ST    2,AINT2
         CLC   AINT1(4),AINT2
         BC    12,CHARG
         MVC   MAREA(4),AINT2
         MVC   AINT2(4),AINT1
         MVC   AINT1(4),MAREA
CHARG    L     1,AINT1
MODIF    OI    0(1),X'01'
         LA    1,2(1)
         C     1,AINT2
         BC    10,ECRIT
         CLC   0(2,1),GEPM
         BC    6,MODIF
         LA    1,8(1)
         B     MODIF
SAUTO    CLI   GOM,C'O'
         BC    7,TESTRAC
         LA    3,GDOA
         AR    2,3
         OI    1(2),X'01'
         B     ECRIT
```

```
TESTRAC  CLI    TRACE,C'O'
         BC     7,TESTINT
         NI     0(11),X'FE'
         B      ECRIT
TESTINT  CLI    INTER,C'O'
         BC     7,ECRIT
*********************************
* CREATION D UNE NOUVELLE COM *
* POSANTE DE LA FIGURE SOIT A *
* PARTIR DE LA GRILLE SOIT A  *
* PARTIR DE CE QUI EST DEJA   *
* TRACE SUR L'ECRAN           *
*********************************
COMCON   NC     COMSAVE+8(4),MASK
         MVC    ACENTRE(4),COMSAVE+8
         OI     COMSAVE+8,X'40'
         MVC    10(4,9),0(9)
         MVC    0(4,9),GEPM
         MVC    2(4,9),COMSAVE+8
         MVC    6(2,9),GEVI2
         MVC    8(2,9),VPHI
*********************************
* INCREMENTATION ET SAUVEGARDE *
* DE PTBUF ET TRANS+16         *
*********************************
         LA     9,10(9)
         ST     9,PTBUF
         LA     9,4(9)
         ST     9,TRANS+16
         MVI    GRIL,C'N'
ECRIT    GWRITE DECOD,STR,BCDE,OCBP
         WAIT   ECB=DECOD
         L      13,4(13)
         RETURN (14,12)
```

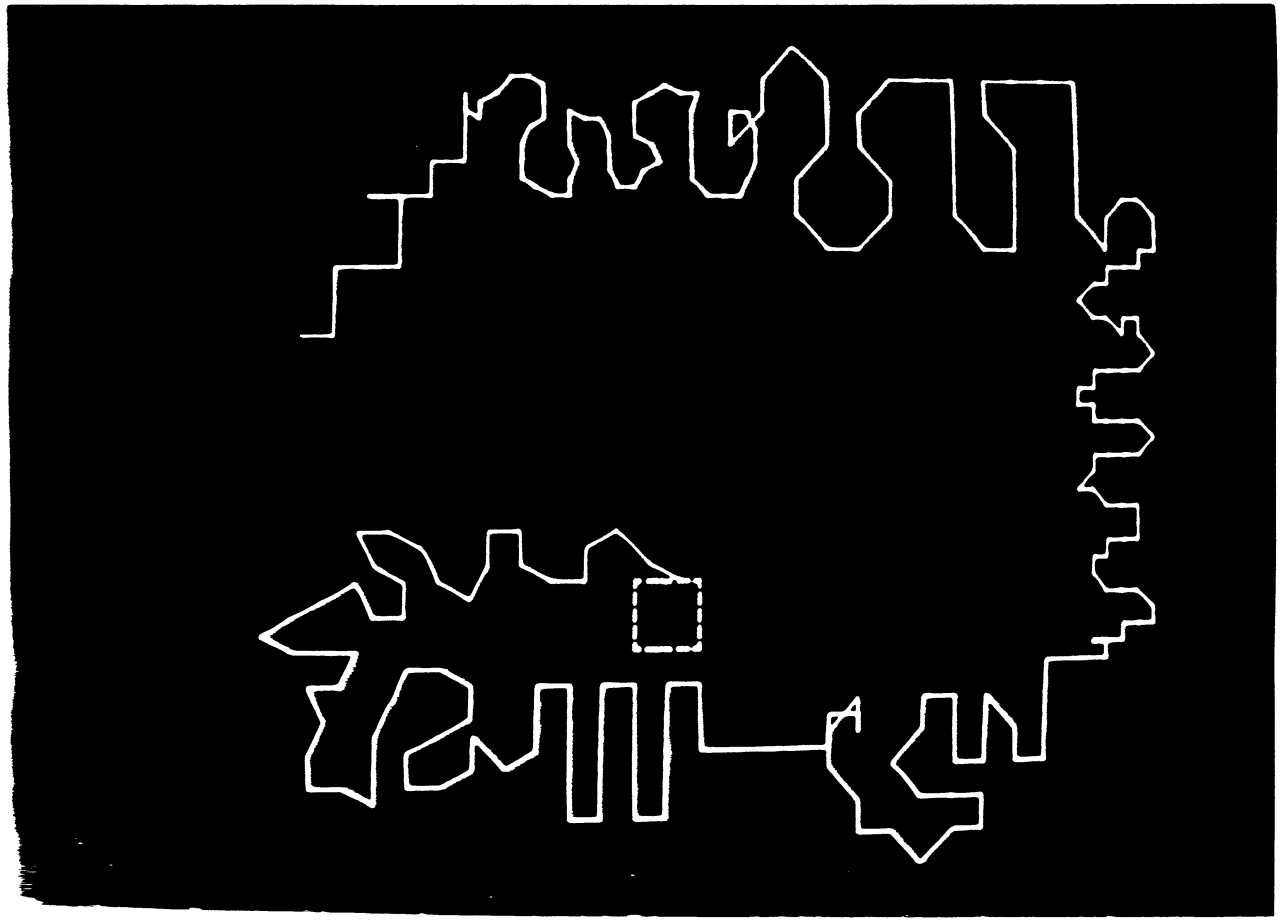
```
*************************************************************
* RETCHAR GERE LES INTERRUPTIONS PROVOQUEES A PARTIR DE*
* LA TOUCHE (ALT+5) DU CLAVIER ALPHANUMERIQUE,SUIVANT  *
* LA VALEUR DE LA CLE PRESSEE AU CLAVIER DE FPS LA FIGU*
* RE RECHERCHEE EST GENEREE,LA FIGURE TEST CHOISIE EST *
* SELECTIONNEE ET LES PARAMETRES CORRESPONDANTS (DIVDEN*
* DE,GRIL,MASK,AVECTEUR) SONT TRANSMIS AU PROGRAMME COD*
*************************************************************
RETCHAR  SAVE   (14,12)
         DROP   8,9,10,11,12
         BALR   8,0
         USING  HERET,8,9,10,11,12
HERET    L      9,ADDRT9
         L      10,ADDRT10
         L      11,ADDRT11
         L      12,ADDRT12
         B      SVREGT
ADDRT9   DC     A(HERET+4096)
ADDRT10  DC     A(HERET+8192)
ADDRT11  DC     A(HERET+12288)
ADDRT12  DC     A(HERET+16384)
SVREGT   LA     3,SVAREA2
         ST     3,8(13)
         ST     13,4(3)
         LR     13,3
         MVC    GDOA(2),GSRT
         MVC    GDOA+2(4),TRARC
         MVC    GDOA+6(2),GENSD
*********************************
* VIENT ON DE LA CLE 0 OU 1 *
*********************************
         CLI    IDCLE,X'01'
         BC     8,CLEF1
CLEF0    GREAD  DECBID,CUR,BCDE,8,IDFIG,BACUR
         WAIT   ECB=DECBID
         MVC    GDOA+126(2),GDPD
         LA     3,INDEX
         LR     4,3
         LA     4,160(4)
*************************************
* RECHERCHE DE IDFIG DANS INDEX     *
* SI CETTE RECHERCHE EST POSITIVE   *
* ON AFFICHE LA FIGURE CORRESPON    *
* DANTE SINON ON PREPARE L'AFFI-    *
* CHAGE DE LA GRILLE                *
*************************************
RECH     CLC    0(7,3),IDFIG
         BC     8,FIGEXIST
         LA     3,16(3)
         CR     3,4
         BC     4,RECH
         MVI    EXIST,C'N'
         LA     5,GDOA
         LA     5,132(5)
         ST     5,TRANS+16
         MVC    PTBUF(4),PTBUF0
         MVC    GDOA+126(2),GDPD
         MVC    GDOA+128(4),TRFIGT
         B      ALLUM01
```

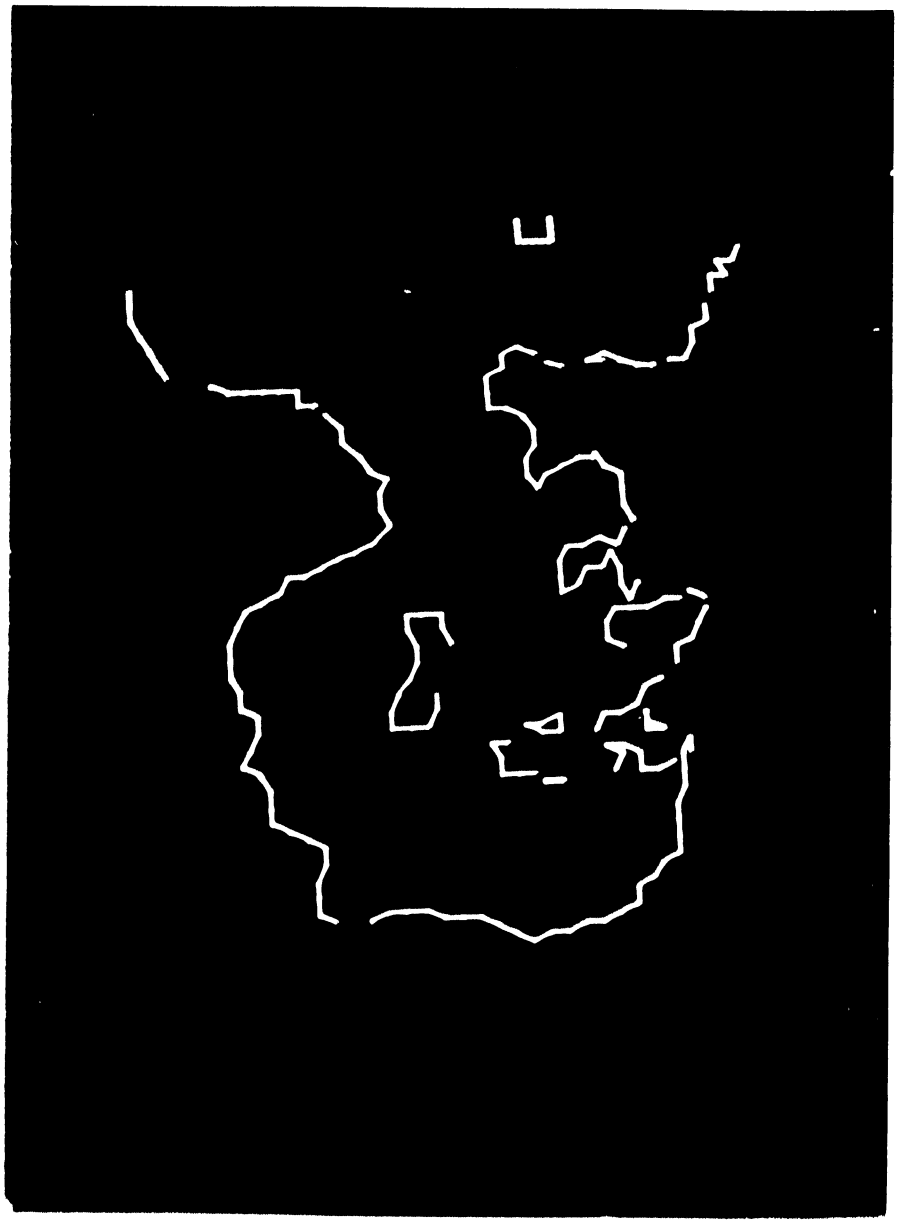
```
         B       ALLUM01
FIGEXIST ST      3,AINDFIG
         MVI     EXIST,C'0'
         LA      5,GDOA
         LA      5,128(5)
         L       6,12(3)
         AR      6,5
         ST      6,TRANS+16
         L       6,8(3)
         MVC     GDOA+126(2),GDPD
         MVC     GDOA+8(4),TRDES
         MVC     0(256,5),0(6)
         MVC     256(256,5),256(6)
         MVC     512(256,5),512(6)
         GWRITE  DECRT0,STR,BCDE,OCBP
         WAIT    ECB=DECRT0
ALLUM01  GCNTRL  DECRT1,IND,BCDE,K01
         WAIT    ECB=DECRT1
```

```
*********************************************************
* PFK EST LE PROGRAMME QUI GERE LES INTERRUPTIONS       *
* PROVOQUEES A PARTIR DU TABLEAU DE FONCTIONS PRO       *
*                                                       *
* GRAMMEES.                                             *
* CLE0 : RENTRE IDFIG.CLE1: RENTRE IDFIGTEST            *
* CLE2 : RETOUR A GRILLE (FIGURE SAUVEGARDEE)           *
* CLE3 : RETOUR A GRILLE (FIGURE EFFACEE)               *
* CLE4 : MODE TRACANT. CLE8 : MODE D'INTERSECTION       *
* CLE18: MODE GOMME SIMPLE.CLE25: MODE SUPERGOMME       *
* CLE27: ENREGISTREMENT DE LA FIGURE                    *
* CLE31: FIN RETOUR SOUS CMS                            *
*********************************************************
PFK      SAVE  (14,12)
         DROP  6,7,10,12,14
         BALR  8,0
         USING HEREK,8,9,10,11,12
HEREK    L     9,ADDRK9
         L     10,ADDRK10
         L     11,ADDRK11
         L     12,ADDRK12
         B     SVREGK
ADDRK9   DC    A(HEREK+4096)
ADDRK10  DC    A(HEREK+8192)
ADDRK11  DC    A(HEREK+12288)
ADDRK12  DC    A(HEREK+16384)
SVREGK   LA    3,SVAREA1
         ST    3,8(13)
         ST    13,4(3)
         LR    13,3
         MVC   PFKC(1),COMSAVE1+2
         CLI   PFKC,X'00'
         BC    7,PFK1
         MVI   IDCLE,X'00'
         MVC   PARTABT(8),PARTABT0
         MVC   BACUR(2),ACURSOR0+2
         MVC   TRANST+16(4),TRANST
REMPLIS  CALL  GSTOR,(OCBPT,PARTABT)
         GWRITE DECPFK1,STR,BCDE,OCBPT
         WAIT  ECB=DECPFK1
         GCNTRL DEPFK2,HLT,BCDE,BACUR
         WAIT  ECB=DEPFK2
         GCNTRL DEPFK3,INS,BCDE,BACUR
         WAIT  ECB=DEPFK3
         GCNTRL DEPFK4,STR,BCDE,STADR
         WAIT  ECB=DEPFK3
         GREADR INPUT,MIP,BCDE,MIPAREA
         WAIT  ECB=INPUT
         B     SORPFK
PFK1     CLI   PFKC,X'01'
         BC    7,PFK2
         MVI   IDCLE,X'01'
         MVC   BACUR(2),ACURSOR1+2
         MVC   TRANST+16(4),TRANST
         MVC   PARTABT(8),PARTABT1
         B     REMPLIS
PFK2     CLI   PFKC,X'02'
         BC    7,PFK3
         MVI   GRIL,C'0'
         GWRITE DEPFK5,STR,BCDE,OCBPG
         WAIT  ECB=DEPFK5
ALL      GCNTRL DEPFK6,IND,BCDE,KALL
         WAIT  ECB=DEPFK6
         B     SORPFK
```

```
PFK3     CLI   PFKC,X'03'
         BC    7,PFK4
         MVI   GRIL,C'O'
         MVC   PTBUF(4),PTBUF0
         L     3,PTBUF0
         MVC   GDOA+128(4),TRFIGT
         MVC   GDOA+126(2),GDPD
         LA    3,4(3)
         ST    3,TRANS+16
         B     ALL
PFK4     CLI   PFKC,X'04'
         BC    7,PFK11
         XI    PARITE4,X'01'
         CLI   PARITE4,X'01'
         BC    7,SOPAR4
         MVI   TRACE,C'O'
         B     ALL
SOPAR4   MVI   TRACE,C'N'
         GCNTRL DEPFK7,IND,BCDE,K4
         WAIT  ECB=DEPFK7
         B     SORPFK
PFK11    CLI   PFKC,X'0B'
         BC    7,PFK18
         XI    PARITEB,X'01'
         CLI   PARITEB,X'01'
         BC    7,SOPARB
         MVI   INTER,C'O'
         MVC   GDOA+126(2),GENSD
         GCNTRL DEPFK8,IND,BCDE,KB
         WAIT  ECB=DEPFK8
         B     RECRIT
SOPARB   MVI   INTER,C'N'
         MVC   GDOA+126(2),GDPD
         GCNTRL DEPFQ8,IND,BCDE,KALL
         WAIT  ECB=DEPFQ8
RECRIT   GWRITE DEPFK9,STR,BCDE,OCBP
         WAIT  ECB=DEPFK9
         B     SORPFK
PFK18    CLI   PFKC,X'12'
         BC    7,PFK25
         XI    PARITE12,X'01'
         CLI   PARITE12,X'01'
         BC    7,SOPAR12
         MVI   GOM,C'O'
         MVC   GDOA+126(2),GENSD
         GCNTRL DPFK10,IND,BCDE,K12
         WAIT  ECB=DPFK10
         B     RECRIT
SOPAR12  MVI   GOM,C'N'
         MVC   GDOA+126(2),GDPD
         GCNTRL DPFK11,IND,BCDE,KALL
         WAIT  ECB=DPFK11
         B     RECRIT
PFK25    CLI   PFKC,X'19'
         BC    7,PFK27
         XI    PARITE19,X'01'
         CLI   PARITE19,X'01'
         BC    7,SOPAR19
         MVI   SUPGOM,C'O'
         MVC   GDOA+126(2),GESD
         GCNTRL DPFK12,IND,BCDE,K25
         WAIT  ECB=DPFK12
SOPAR19  MVI   SUPGOM,C'N'
```

```
         MVC   GDOA+126(2),GDPD
         GCNTRL FOLKLO,IND,BCDE,KALL
         WAIT  ECB=FOLKLO
         B     RECRIT
PFK27    CLI   PFKC,X'1B'
         BC    7,PFK31
         B     ENRMIN
PFK31    CLI   PFKC,X'1F'
         BC    7,SORPFK
         POST  COMSAVE1+16
```





```
display 13a40-13d40
L  13A40 = 00000000  00000000  2A822AFF  007E2A80  2A044100  01F80110  01100108  01080110
L  13A60 = 0108F900  F100F900  F900F100  F900F900  F100F900  F900F100  F90001F8  01F001F8
L  13A80 = 01F801F0  01F801F8  01F001F8  01F801F0  01F80900  11000900  09001100  09000900
L  13AA0 = 11000900  09001100  09000108  01100108  01082AFF  00002A05  21E10120  01200100
L  13AC0 = E100E100  01002A85  2A004880  00002A05  0101C101  01C101C1  01C04100  410001C0
L  13AE0 = 01C04100  41000140  01C001C0  410001C0  41004100  E100F1E0  210011E0  21E001E0
L  13B00 = F1E0E100  E100F120  E110E100  E100F1E0  F1E001E0  210021F0  21102100  2100F1E0
L  13B20 = 01E0E1F0  E100E100  E1F001E0  21F011E0  21101120  21002100  11E011E0  F1E001E0
L  13B40 = E110E100  E100E100  E100E1E0  01E001E0  21F021F0  21002100  21100120  C10041C0
L  13B60 = 410041C0  C1C0C100  C100C140  C100C1C0  01C041C0  41004140  410041C0  01C001C0
L  13B80 = C100C100  C100C100  C1C001C0  41004100  41004140  410001C0  01C001C0  C100C100
L  13BA0 = C100C100  C1C04100  21E001E0  E1E0E100  E1000120  E1000120  0120E100  0120E120
L  13BC0 = E1E001E0  E1E02100  01E0E100  E1E0E120  01200120  0120E100  0120E100  01E0E100
L  13BE0 = 01E001E0  01E0E1E0  E1200120  01200120  E100E120  01E0E1E0  01E001E0  E100E100
L  13C00 = 01200120  E1000120  E100E1E0  01E001E0  E1E0E100  01200120  E1000120  012001E0
L  13C20 = 012001E0  E1000140  0140C100  C100C100  01404100  4140C100  C1000140  41004100
L  13C40 = 0140C100  C140C1C0  01C001C0  C1000140  C1404140  01404100  41404100  41C0C100
L  13C60 = 21000140  C1000140  01400140  01404100  41000140  C100C100  C100C100  01404100
L  13C80 = 41004100  41000140  C100C100  C100C100  01404100  41004100  41000140  C100C100
L  13CA0 = E1404140  C100E140  01404100  21C021C0  41002140  0140C120  C120C100  C1002140
L  13CC0 = 01404100  41E04120  01C041E0  01400140  214021C0  21C021C0  C1E001C0  41002140
L  13CE0 = 412001C0  E1C0C1E0  E1C041E0  410001C0  C100E1C0  01C04100  21C0C1C0  E1C0C121
L  13D00 = 2AFF0006  00000000  00000000  00000000  00000000  00000000  00000000  00000000
L  13D20 = 00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
L  13D40 = 00000000
```

```
L  13A40 = 00000000 00000000 2A822AFF 007E2A86 2A042100 01FC0108 01080104 01040108
L  13A60 = 0104FD00 F900FD00 FD00F900 FD00FD00 F900FD00 FD00F900 FD0001FC 01F801FC
L  13A80 = 01FC01F8 01FC01FC 01F801FC 01FC01F8 01FC0500 09000500 05000900 05000500
L  13AA0 = 09000500 05000900 05000104 01080104 01042AFF 00002A04 410001E8 01080108
L  13AC0 = 01100108 01082A86 2A004880 0B802A05 010111E1 210111E1 21F001E0 01E0F1E0
L  13AE0 = 01E021F0 21F021F0 01E001E0 11E021F0 F1E0F1E0 01E021F0 01E011E0 01E001E0
L  13B00 = F1E0F1E0 E1F0E1F0 F1E0E100 E1F0E1F0 E1F0E1F0 E1E0F1E1 E1F1E101 01E1E101
L  13B20 = E101E101 E101E1F1 2A004880 0B002A05 0100E1F0 E100E100 E110E100 E100E110
L  13B40 = E110E110 E1F0E100 E1F0E100 E1F0E1F0 01E0E1F0 F1E0E1F0 01E001E0 01E0F1E0
L  13B60 = 01E001E0 F1E00120 21112110 210001E0 21F02100 E1101120 F1E01121 2A004600
L  13B80 = 09002A05 0100E100 2100E100 2100F1E1 F1E12111 11212100 210001E0 11E0E100
L  13BA0 = E1012A00 44800780 2A050101 11212111 21111120 F1E0E100 E1F0F1E0 E1F001E1
L  13BC0 = E10101E0 E1F0F1E0 F1E0E1F1 01E10121 210121F0 21011121 11211121 21F101E1
L  13BE0 = 11210121 E11101E1 2A004480 06002A05 01001120 11E001E0 11E01120 21001120
L  13C00 = 211001E0 01E0F1E0 E100E1F0 E110F1E0 F1E121F1 11E101E1 21012101 11E101E1
L  13C20 = 11E1F1E1 0121E101 2A004900 02802A05 01011121 E1110121 E1200120 E120F120
L  13C40 = E1101120 0120F120 01E10121 E10101E1 2A004900 03002A05 01002100 01E02100
L  13C60 = 21002100 210021F0 21012101 F1E12111 E1012101 11E011E0 11E011E0 01E001E0
L  13C80 = 0120E101 2A004400 03002A05 0101E101 2101E101 21012121 21112101 E101E101
L  13CA0 = E1F1E101 F1E1E101 F1E1E101 E1F1E101 01E10121 F1E111E1 01E101E1 E101E111
L  13CC0 = 2A004700 07802A05 01012101 012121E1 E1112101 01201120 21002100 01E0F1E0
L  13CE0 = F1E0F1E0 01E011E0 01E0E100 E1000120 F120E101 2A004600 08802A05 010101E1
L  13D00 = 21F0E100 E1F00120 2100F121 F121E111 2A004C00 05802A05 01012101 E1012A00
L  13D20 = 44000600 2A050101 F1212100 21102100 21001120 0120E110 E1112101 E10121F1
L  13D40 = 2A004480 08002A05 01000120 0121F1E1 1121F1E1 F1E12111 E1102100 2101E101
L  13D60 = F1E10121 E111E1F1 E101F1E1 E101E1F1 E101F1E1 E101E101 E1F101E1 01E101E1
L  13D80 = 11E1F1E1 E1112A00 44800500 2A050100 11E001E0 01E021F0 11E02101 21012111
L  13DA0 = 01212101 210101E1 F1E1E101 01E111E1 21112101 01E101E1 01E101E1 E121E1F1
L  13DC0 = 2A004600 03802A05 01012101 F1E101E1 21F12111 2101F1E1 11E1F1E1 01E101E1
L  13DE0 = 01E1E1E1 01E12A00 45000400 2A050100 21002110 21101120 11E001E0 F1E001E0
L  13E00 = 11E021F0 210001E0 01E0E1F0 01E0E1F0 E110E121 01E10121 21010121 2101E101
L  13E20 = E10101E1 E11101E1 E11101E1 E101E1F1 F1E1E101 E1F1E1F1 F1E10121 F1E1E1F1
L  13E40 = 11E1E101 01E1F1E1 E11101E1 2111E101 01E1F121 F1E1F1E1 1121F1E1 21210121
L  13E60 = F1E11121 F1E10121 01E11121 11202100 F1202100 01200121 01E11121 01202110
L  13E80 = 01201120 21002101 21012101 21012100 012101E1 01212101 21112101 2111E101
L  13EA0 = 11E1E101 E1E101E1 11E101E1 01E121F1 01E111E1 2101F1E1 01E12101 E101E101
L  13EC0 = E1012A00 44000280 2A050100 210021F0 21F02110 21F11121 01E12111 E110E101
L  13EE0 = 11E111E1 11E121E1 11E121E1 21E121F1 E111E101 E1F1E1E1 01E12AFF 00060000
L  13F00 = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
L  13F20 = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
L  13F40 = 00000000 00000000 "
```

REFERENCES BIBLIOGRAPHIQUES

[cag] J.M. CAGNAT — *Structures de représentation des données en ordinateur. Application aux traitements graphiques.* Thèse de 3ème Cycle. Université de Grenoble. Février 1971

[cle] E. CLEEMANN — *Un macro-langage pour la programmation des terminaux graphiques.* Thèse de 3ème Cycle. Université de Grenoble. Mars 1969.

[col] L. COLLATZ — *The Numerical Treatment of differential equations.* Springer-Verlag. Berlin 1966.

[des] A. DESSENNE — *Rapport sur la série des isolants.* Mathématiques Appliquées - Grenoble juin 1969

[fed] J.F. FEDER — *Languages of encoded line patterns.* Information and control. n° 13, pp. 230-244. 1968.

[for] G.E. FORSYTHE et W.R. WASOW - *Finite-Difference methods for partial differential equations.* John Wiley & Sons. New-York. 1960

[fre] H. FREEMAN — *On the encoding of arbitrary geometric configurations.* IRE Transactions on Electronic Computers. pp. 260-268 Juin 1961.

[gin] S. GINSBURG — *The mathematical theory of context free languages.* Mc Graw Hill Book Company New-York 1966.

[hop] J.E. HOPCROFT et J.D. ULLMAN - *Formal languages and their relation to automata.* Addison - Wesley Publishing Company. 1969.

[Ibm 1] IBM SYSTEM/360 Component description : IBM 2250 display unit model 1, form A27-2701-1.

[Ibm 2] IBM SYSTEM/360 operating system : Graphic programming services for IBM 2250 display unit, form C27-6909-4.

[lec] O. LECARME *Contribution à l'étude des problèmes d'utilisation des terminaux graphiques. Un système de programmation graphique conversationnelle.*
Thèse de Sciences Appliquées. Université de Grenoble. Septembre 1970.

[mag] Projet MAGOG rapport d'activité 70-71.
(A paraître.) Mathématiques Appliquées - Grenoble

[mie] J.M. MIERMONT *Langages vectoriels et calculabilité de figures curvilignes.*
Mathématiques Appliquées - Grenoble - Février 1970.

[mon] G.U. MONTANARI *On limit properties in digitization schemes.*
Journal of the A.C.M. Vol 17, n° 2, pp. 348-360. Avril 1

[roz] A. ROSENFELD *Picture processing by computer.*
Computing surveys, Vol. 1, n° 3, Septembre 1969.

[til] C.C. TILLMAN Jr *EPS : an interactive system for solving elliptic boundar value problems.*
User's Guide MAC.TR.62. ESL.R.395. Juin 1969.