Conception et réalisation d'un logiciel de saisie et de restitution de cartes élémentaires

Éric Secher

IRIA

CENTRE DE RENNES
IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél. 954 90 20

Rapports Techniques

N° 28

CONCEPTION ET RÉALISATION D'UN LOGICIEL DE SAISIE ET DE RESTITUTION DE CARTES ÉLÉMENTAIRES

Eric SECHER

Août 1983

Conception et réalisation d'un logiciel de saisie et de restitution de cartes élémentaires

Par Eric SECHER

Publication Interne n°188
45 pages
Janvier 1983

RESUME

Ce rapport analyse quelques problèmes rencontrés lors de la conception et de la réalisation d'un logiciel de création interactive de cartes thématiques. Les différentes phases d'un tel travail sont étudiées (création et modification des fonds de cartes, acquisition des séries de valeurs, choix des paliers et affichage de la carte). Une attention particulière a été apportée à l'analyse des algorithmes de remplissage des zones de la carte. Deux techniques ont été utilisées: un balayage zone par zone, ou un balayage de l'ensemble des zones suivant une ligne verticale. Cette deuxième méthode s'avère la meilleure.

ABSTRACT

This report presents some problems encountered when designing and implementing an interactive package for thematic maps drawing. The successive steps of such a creation are analyzed (creation and modification of contour maps, data input, gray scale simulation by cross-hatching, final map computation). Detailed analysis of algorithms for cross-hatching are provided. Two general techniques are studied: cross-hatching by processing one area after the other, or global cross-hatching by swweping all the areas with a vertical line. The second method is proved to be the best one.

Ce travail a été effectué dans le cade d'un stage du DEA d'informatique de l'Université de Rennes, au Centre de Calcul Universitaire de Nantes sous la direction de Michel Lucas.

Sommaire

Ce document a été photocomposé à l'IRISA sur GS8800 par Maryse Fouché

INTRODUCTION

La graphique est un vaste domaine qui s'étend de la représentation non figurative comme les arts plastiques à la réalisation de diagrammes, réseaux cartes. Elle est dans tous les cas un outil efficace de communication.

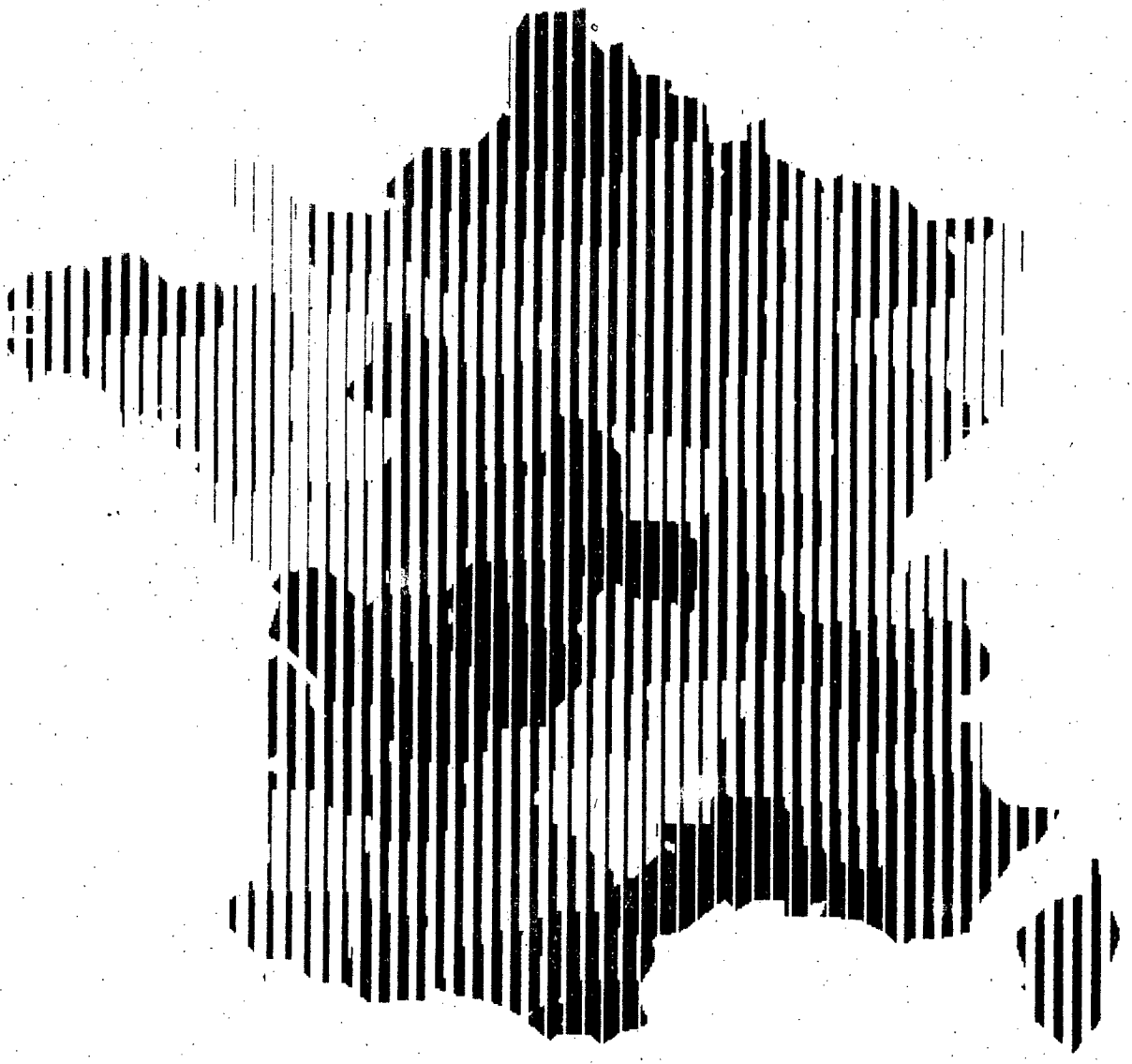
Là nous nous intéresserons à la cartographie qui est un moyen, par les tableaux, les diagrammes, les réseaux ou les courbes, de la représentation graphique de données sur tel ou tel objet. Mais la carte a une propriété bien particulière qu'elle est seule à avoir, elle fournit l'information de proximité topographique immédiatement. Quelle soit cartographique ou non, la représentation graphique demande de respecter certaines règles pour une meilleure compréhension et rapidité d'analyse. Sur ce sujet, on peut lire : « La graphique et le traitement graphique » de Bertin.

La cartographie se découpe en deux parties bien différentes, mais complémentaires : la saisie du fond de carte et son utilisation. L'utilisation qui est traitée ici concerne le hachurage des différentes régions constituant la carte.

Deux parties constituent ce rapport. La première est consacrée à la présentation et aux problèmes généraux rencontrés. La seconde présente les algorithmes pour se terminer sur une comparaison entre les deux méthodes proposées de hachurage :

- un remplissage région par région
- un remplissage en une seule passe par balayage horizontal de l'ensemble des régions

Première partie
Présentation et généralités



I. La saisie du fond de carte

I.1 Niveau de simplification du fond de carte

Le premier problème qu'on peut soulever est celui de la complexité ou de la simplification d'un fond de carte. Par exemple, doit-on représenter la France par un simple hexagone ou au contraire suivre avec précision ses côtes et ses frontières. La première contrainte est bien évidemment la taille du fond de carte. Plus elle est petite, moins de détails pourront apparaître.

Ensuite intervient l'information qu'elle doit contenir. Si cette information est limitée à quelques points (villes,...) une carte grossière pourra peut être suffire. Par contre, si l'information est connue pour un ensemble de zones ayant des limites précises (régions, départements) plus de détails pourront être nécessaires. Intervient également l'information recherchée sur la carte, c'est-à-dire les groupements à découvrir entre les différentes zones ou d'autres renseignements. (Voir J.Bertin « La graphique et le traitement graphique de l'information » p.145).

Et il me semble qu'un dernier aspect peut intervenir dans le niveau de simplification d'un fond de carte, celui de l'utilisation future de la carte (publication). Par exemple, un lecteur d'un quotidien d'information générale sera plus attiré par une carte détaillée que par une carte schématisée.

Mais le choix incombe à l'utilisateur et en tant que concepteur d'un logiciel de saisie de carte, on ne peut que lui permettre de mémoriser un fond de carte détaillé en lui laissant de la place mémoire.

L'utilisation d'un micro-ordinateur le permettant et comme il est toujours difficile d'évaluer la taille mémoire nécessaire, à chaque nouvelle définition d'un élément de la carte, un test permet de savoir si la mémoire réservée est pleine ou non. En cas de dépassement, une procédure arrête le programme pour permettre à l'utilisateur de modifier la place réservée et de modifier les lignes de programme en cause, un manuel d'utilisation indiquant la marche à suivre.

I.2 La saisie

Nous nous intéressons à des cartes découpées en zones ayant toutes un pourtour polygonal. Par conséquent, le pourtour d'une zone sera défini par l'ensemble des sommets du polygone la représentant. Deux zones contiguës ont nécessairement un ou plusieurs sommets communs. Mais pour limiter le travail de l'utilisateur du logiciel de saisie de fond de cartes, nous devons faire en sorte qu'il entre un minimum de points et par conséquent éviter qu'il ne rentre plusieurs fois le même point. Il ne devra rentrer que les points nécessaires à la délimitation d'une nouvelle région, c'est-à-dire les points jusqu'à présent inconnus et les éventuels points de rattachements déjà connus. Mais ces derniers points ne peuvent suffire à caractériser une région, il devra donc indiquer un sens de finition pour que le système retrouve automatiquement les autres points nécessaires pour caractériser la région.

Tout ceci m'a amené à introduire la notion de **périmètre vide**. C'est un pourtour à l'intérieur duquel aucune région n'a encore été délimitée. Comme une carte peut être constituée de parties indépendantes, n'ayant aucun sommet commun, il a fallu

introduire également la notion de **sous-carte**, de telle manière que les pourtours de deux sous-cartes n'ont aucun sommet, ni aucun points communs. Un utilisateur doit donc commencer par délimiter le pourtour d'une sous-carte, puis délimiter une région à l'intérieur de cette sous-carte qui s'appuie sur son pourtour, jusqu'à la dernière région dont il n'aura à indiquer aucun sommet, puisqu'elle sera délimitée par les pourtours des autres régions de la sous-carte. Eventuellement, il pourra redéfinir une sous-carte et délimiter les régions la constituant de la même manière que précédemment.

Un pourtour donné peut servir de base à un autre fond de carte. Par exemple, le fond de carte constitué de la France et de ses régions administratives peut servir pour la France et ses départements. Il fallait donc donner la possibilité de définir un nouveau fond de carte, à partir d'un autre déjà mémorisé, sans quoi le travail demandé à un utilisateur serait beaucoup plus important. L'utilisateur a donc la possibilité de définir de nouvelles zones à l'intérieur d'une zone existante. Dans ce cas, le périmètre vide est le pourtour de la première zone.

Il a fallu résoudre un problème pratique dû à l'utilisation du micro-ordinateur HP 9845. En effet, le générateur de segment du HP n'est pas au point. Un même segment, suivant qu'il est tracé dans un sens ou dans un autre, n'a pas la même représentation. Pour résoudre ce problème, il y avait plusieurs solutions.

La première aurait été de récrire un générateur de segment, mais elle aurait ralenti considérablement l'affichage du pourtour du fond de carte, ce ne pouvait donc pas être la meilleure.

La seconde consistait à n'afficher qu'une seule fois chaque segment de la carte. En effet, les ennuis venaient du fait qu'un segment appartient nécessairement à deux régions et n'était pas forcément orienté de la même manière. Et à la visualisation, on avait des traits épais. Cette solution avait l'avantage de gagner du temps à l'affichage puisqu'il y avait deux fois moins de segments à tracer. De plus, elle est facile à mettre en œuvre. Les segments à tracer sont ceux que l'utilisateur rentre pour délimiter une région. Par contre, lorsque l'utilisateur définit de nouvelles régions à partir d'une région préalablement rentrée, il devient difficile de retrouver les segments à afficher et la mémorisation des segments à afficher devient plus importante et par conséquent le temps d'affichage est plus élevé. Il fallait donc prendre une autre méthode.

La solution retenue consiste à orienter tous les segments de la carte de façon à ce qu'un affichage répété du même segment donne le même résultat pour ne pas avoir de traits épais.

II. Le remplissage du fond de carte

Le choix de la nature du remplissage du fond de carte est très important. Jacques Bertin dans « La graphique et le traitement graphique de l'information »donne quelques règles essentielles pour construire des « cartes à voir ». La « carte à voir » doit permettre de percevoir instantanément la répartition du caractère représenté pour répondre à la question : « tel caractère où est-il? ». Elle montre les groupements définis par la proximité géographique ; et si le caractère est quantitatif, les groupements définis par la proximité numérique. Et elle permet également de répondre à la question : « En tel endroit, quel est le caractère? ». Elle contient donc

tous les niveaux de l'information. La « carte à lire » est celle qui ne permet de répondre qu'à une seule question : « En tel endroit, quel caractère y-a-t-il ? ».

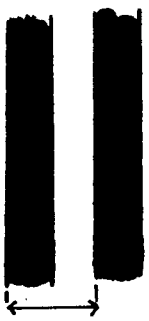
Pour réaliser une « carte à voir », il faut donc utiliser des variables visuelles ordonnées visuellement. Ni la forme, ni l'orientation, ni la couleur, ni l'emplacement d'une tâche ne sont ordonnés. Pour transcrire la représentation d'un caractère ordonné ou quantitatif, seuls la taille, la valeur et le grain peuvent convenir à condition de respecter la règle essentielle suivante : « L'ordre de la variation visuelle doit correspondre à l'ordre de la composante ».

Par conséquent, si le caractère est quantitatif, une petite valeur sera représentée soit par une petite taille, soit par une valeur claire pour atteindre les grandes valeurs avec une grosse taille ou une valeur foncée.

II.1 Le hachurage

Les fonds de carte que nous utilisons correspondent à des cartes comportant des régions. Nous nous intéressons donc à une implantation zonale des caractères ordonnés ou quantitatifs et le hachurage est une technique simple qui permet de transcrire ces caractères.

Le hachurage est une variable visuelle assimilée à la valeur, l'intensité variant avec l'épaisseur de la bande de hachurage. Une épaisseur nulle correspondant à une valeur claire et une épaisseur égale au pas du hachurage correspondant à une valeur foncée (noir).



2 bandes de hachurage

pas du hachurage

Le hachurage peut aussi être assimilé au grain. Mais dans la mesure où le pas de hachurage est constant, on s'écarte de cette variable visuelle. Un pas irrégulier permettrait de comparer le hachurage au grain.

Dans une certaine mesure, mais avec une grande prudence, le hachurage par pas réguliers permet de définir une certaine notion de taille. La taille étant représentée par l'épaisseur de la bande de hachurage, mais on doit alors exclure l'épaisseur nulle.

Le hachurage a l'inconvénient d'avoir un nombre limité de paliers. Ce nombre est égal au pas + 1. Et comme il est difficile d'avoir un pas très grand, le nombre de paliers est limité à une petite dizaine. Un pas de sept ou de huit donne un aspect agréable aux cartes obtenues par cette technique de remplissage. En deça et au delà ou bien c'est trop serré ou bien pas assez. Et comme on l'a déjà dit, il a l'avantage d'être simple et par conséquent rapide d'exécution.

Nous proposons deux techniques de remplissage, un remplissage région après région et un remplissage par balayage de l'ensemble des régions. Comme nous le verrons à la fin de ce rapport, la deuxième technique est plus rapide en général, de plus elle a l'avantage de donner une vue d'ensemble sans attirer l'œil sur une région particulière, même si la tache qui la représente est grande, du moins à l'affichage.

III. Options autour du hachurage

Malgré le titre du paragraphe, commençons par voir ce que nous avons imposé. La première contrainte, évidente, c'est que seules les cartes rentrées par le logiciel de saisie associé au logiciel de hachurage pourront être hachurées. Ensuite, une seule direction de hachurage est permise, ce qui veut dire qu'il n'y a pas moyen de faire du hachurage croisé. La direction choisie est verticale et toutes les bandes de hachurage sont alignées.

L'utilisateur peut choisir la valeur *Pas* du pas de hachurage, elle n'est donc pas fixée arbitrairement, mais d'une manière standard, elle est égale à *7*. Il peut également déterminer le nombre de paliers qui doit être inférieur ou égal à la valeur du pas de hachurage plus une unité. Si le pas est égal à *7*, il doit choisir un nombre de paliers inférieur ou égal à *8* allant d'une épaisseur de bande de hachurage nulle (zone toute blanche) à une épaisseur égale à *7* (zone noire) par étapes progressives. D'une façon standard, le nombre de paliers est égal à *Pas + 1* et l'épaisseur de la bande de hachurage varie de *0* à *Pas*. Par contre, s'il choisit un nombre de paliers différent de *Pas + 1*, il devra obligatoirement déterminer l'épaisseur pour chaque palier de telle manière que l'épaisseur associée au palier $i+1$ soit supérieur au palier i et que l'épaisseur du dernier palier soit inférieure ou égale à *Pas*. Voilà pour les options liées directement au hachurage.

Avant l'exécution du hachurage, l'utilisateur devra associer à chaque zone de la carte, une valeur numérique. Le logiciel lui propose trois possibilités :

- lecture sur fichier
- entrée des données à l'aide du clavier
  - soit dans l'ordre qu'il désire en indiquant le nom de la région et sa valeur associée
  - soit dans l'ordre imposé en indiquant seulement la valeur associée à la région dont le nom apparaît sur l'écran.

Ceci nous amène au **problème de l'échelle**. Considérons, dans un premier temps, les hachures comme variables de l'image associée à la valeur. Dans ce cas, nous avons bien une échelle de paliers visuellement équidistants dans la mesure où tous les paliers, au nombre de *Pas + 1*, sont pris en considération et par conséquent l'échelle numérique associée doit être une échelle de paliers équidistants. Le logiciel calcule automatiquement cette échelle. Voyons un exemple où le pas de hachurage est égal à *7* et le nombre de paliers égal à *8*. Le logiciel détermine alors les valeurs minimale et maximale des valeurs associées aux régions de la carte, puis il détermine les limites de chaque palier. Les limites du 2ème palier sont :

$Min + (i-1)*(Max-Min)/8$ pour la limite inférieure et
$Min + i*(Max-Min)/8$ pour la limite supérieure.

Pour déterminer l'épaisseur de la bande de hachurage d'une région donnée, il suffit de comparer sa valeur à ces limites. Cette méthode présente des inconvénients.

Elle est excellente lorsque la répartition des valeurs est bonne, par contre si on a une mauvaise répartition des valeurs, elle peut fausser toute étude en créant de très mauvais groupements. En effet, imaginons une série de valeurs comprises entre *Min* et *Max* dont la quasi-totalité est groupée autour d'une valeur moyenne. La méthode ci-dessous risquerait de couper arbitrairement le groupement central et surtout d'uniformiser la couleur de la carte. Pour pouvoir étudier sérieusement cette série de valeurs, il serait intéressant d'« éliminer » les valeurs extrêmes. Une méthode simple consiste à redéfinir les paliers pour réussir les valeurs extrêmes de la série avec les valeurs extrêmes du groupement central. L'utilisateur doit alors définir la limite supérieure du premier palier et la limite inférieure du dernier palier. Quant aux autres paliers, le calcul de limites se fait automatiquement de la même façon que précédemment.

Maintenant, supposons que la répartition des valeurs soit bonne mais que l'utilisateur veuille s'intéresser aux petites ou grandes valeurs seulement. Il aimera pouvoir « éliminer » les autres valeurs. Dans ce cas, il indiquera suivant les cas une limite inférieure du dernier palier ou une limite supérieure du premier palier. Le calcul automatique des autres paliers se fera de la même manière que dans le cas général en prenant pour valeurs extrêmes la limite indiquée et l'extrémum correspondant.

Peut-on utiliser une échelle logarithmique? Cela semble difficile, pour deux raisons. La première est liée au type de remplissage : le hachurage. Une règle importante de la graphique veut qu'il y ait correspondance entre les données et sa représentation graphique. Pour transcrire une échelle logarithmique, il faut donc que l'échelle des paliers visuels soit également logarithmique (1, 2, 4, 8, 16, ...). Or, il est bien évident que de passer d'une épaisseur i de la bande de hachurage à une épaisseur $i + 1$ ne peut transcrire visuellement le passage d'un palier numérique i au palier $2 * i$. Il m'apparaît donc impossible de traduire une échelle logarithmique à partir des hachures en utilisant la totalité des paliers.

Mais pourquoi ne pas utiliser les paliers visuels liés à une épaisseur de la bande de hachurage égale à 1, 2, 4, 8, etc.? Dans ce cas, on arrive à un nombre limité de paliers. En effet, supposons que nous prenions un pas de hachurage égal à 16, ce qui est énorme, nous n'avons que cinq paliers. Et il est impossible de prendre des paliers intermédiaires puisqu'entre 1 et 2 il n'en existe pas.

Une solution pourrait consister à prendre, comme base, la progression géométrique suivante : $\sqrt{2}$, 2, $2\sqrt{2}$, 4, $4\sqrt{2}$, 8, $8\sqrt{2}$, 16 et de lui associer les épaisseurs suivantes : 1 pour $\sqrt{2} \simeq 1.4$, 2 pour 2, 3 pour $2\sqrt{2} \simeq 2.8$, 4 pour 4, 6 pour $4\sqrt{2} \simeq 5.65$, 8 pour 8, 11 pour $8\sqrt{2} \simeq 11.3$ et 16 pour 16, ce qui donne 8 paliers. Mais les approximations sont importantes, surtout pour les premières valeurs, et il me semble que ce serait trahir la réalité numérique que de l'employer puisque dans ce cas le système de perception doit être fondé sur la progression géométrique de la surface des bandes de hachurage.

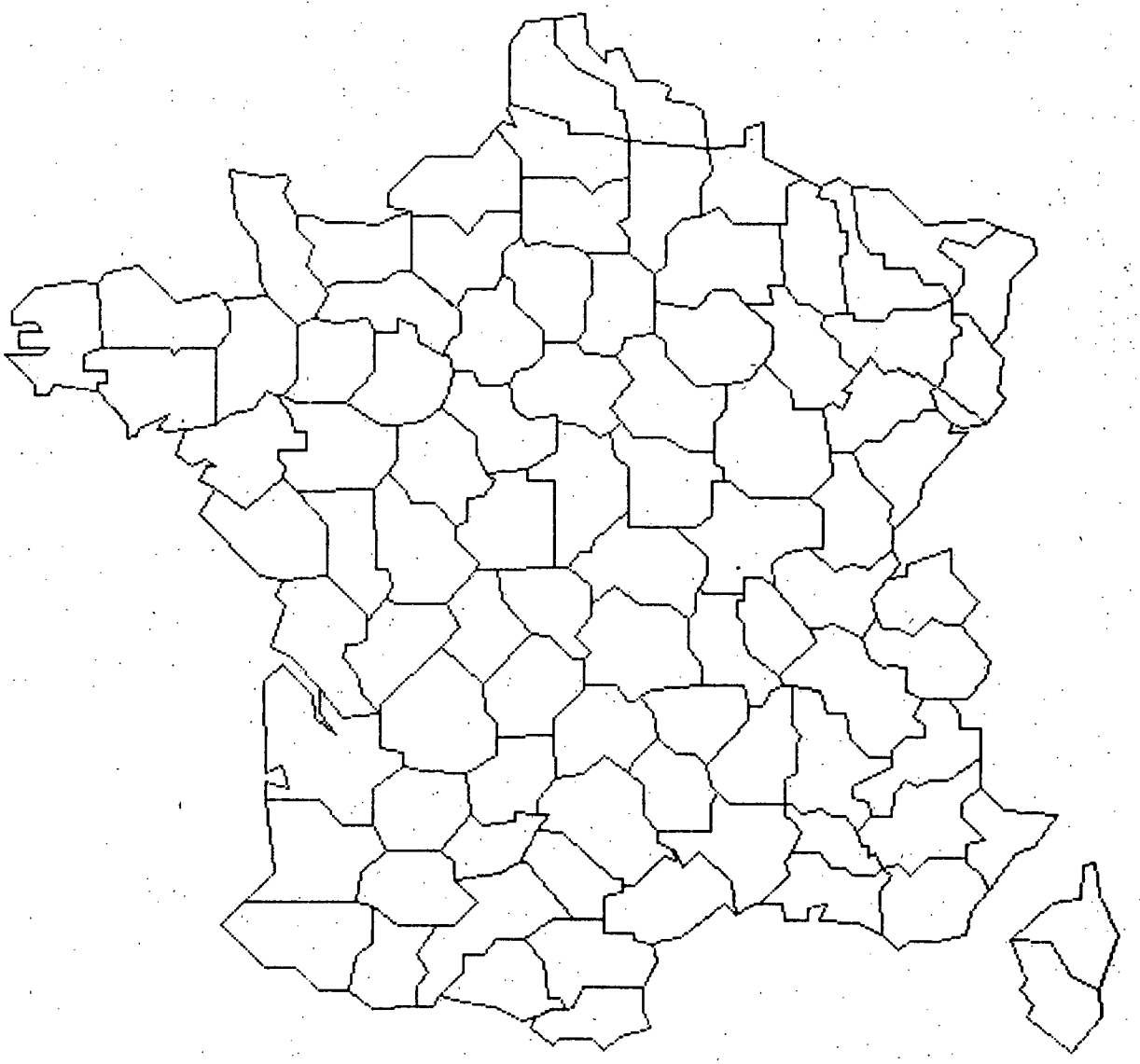
Le hachurage reste donc fondé sur une progression arithmétique des quantités à transcrire, puisque le système de perception se base sur l'équidistance visuelle des paliers de valeur.

L'affichage du pourtour des régions a tendance à gêner une perception correcte de l'ensemble de la carte. En effet, le tracé du pourtour est relativement épais même

dans de bonnes conditions d'affichage et il perturbe la vision de la carte en attirant l'œil. Il semble donc intéressant de permettre l'effacement des pourtours. Mais, si cet effacement est réalisé après le hachurage, le problème reste identique, il faut donc pouvoir éteindre les contours avant le hachurage. Or, il est possible qu'après hachurage une région côtière ou frontalière soit blanche, il faut donc pouvoir afficher le pourtour des sous-cartes pour retrouver cette région.

Tout ceci, fait que l'utilisateur peut afficher ou effacer les pourtours avant ou après le hachurage et que cet affichage (ou effacement) doit être sélectif en distinguant les pourtours des sous-cartes des pourtours de régions délimitées à l'intérieur des sous-cartes.

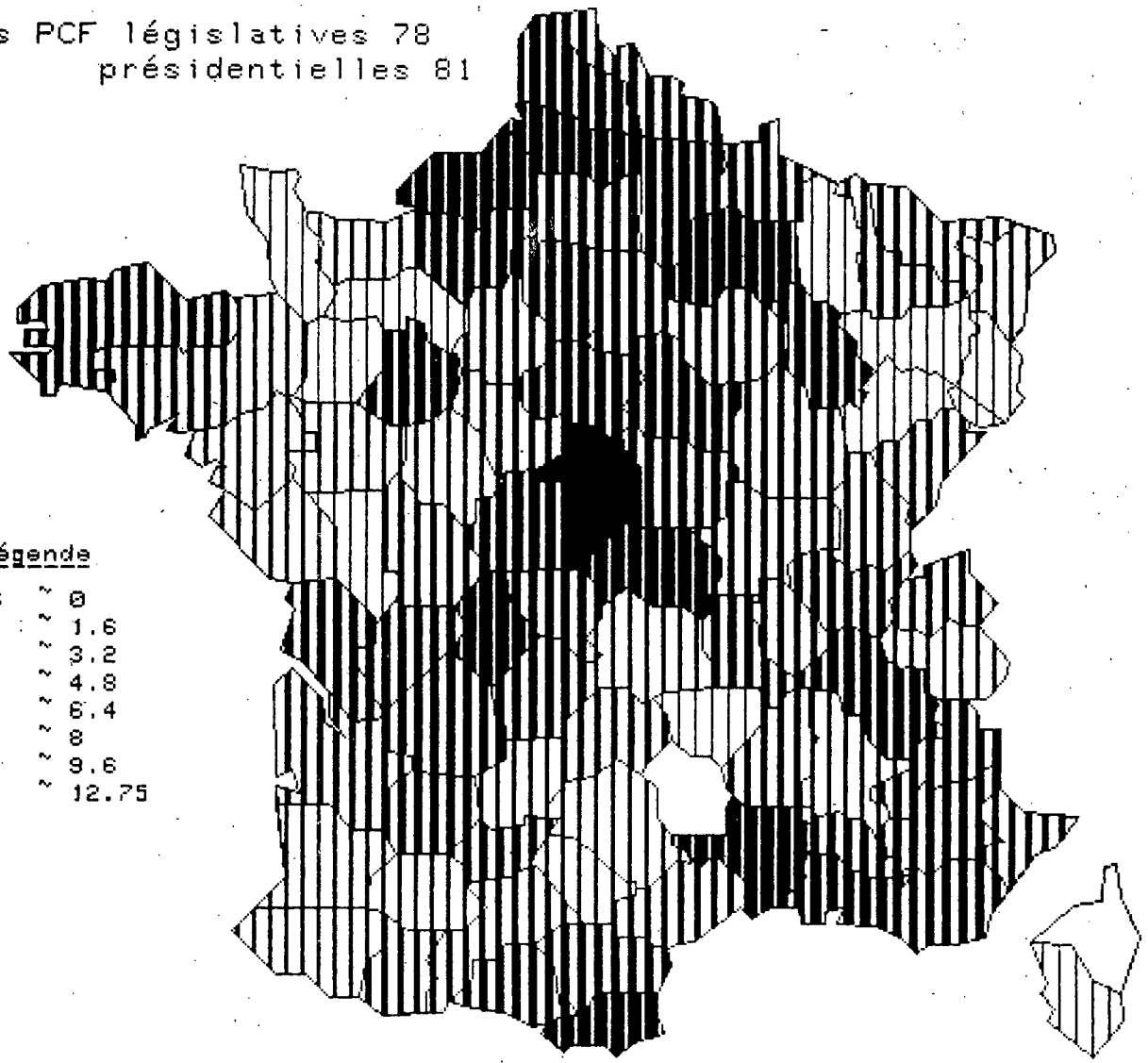
Avant de parler des algorithmes, il ne reste plus qu'à signaler la possibilité qu'a l'utilisateur d'afficher un titre et un sous-titre sur sa carte et de donner la légende en côté ou sous la carte, et bien évidemment l'impression sur papier pour garder une trace de son travail.



un exemple de fond de carte: France par départements

programme CARTOTHEM (université de Nantes)

pertes PCF législatives 78
présidentielles 81



LE MONDE 26 Avril 1981

Fichier PC8178

un exemple de carte hachurée

programme CARTOTHEM (université de Nantes)

Seconde partie
Les algorithmes

I. Prise du fond de carte

Ne disposant pas de tablette graphique, la prise de fond de carte s'effectuera, au choix de l'utilisateur, de trois manières différentes :

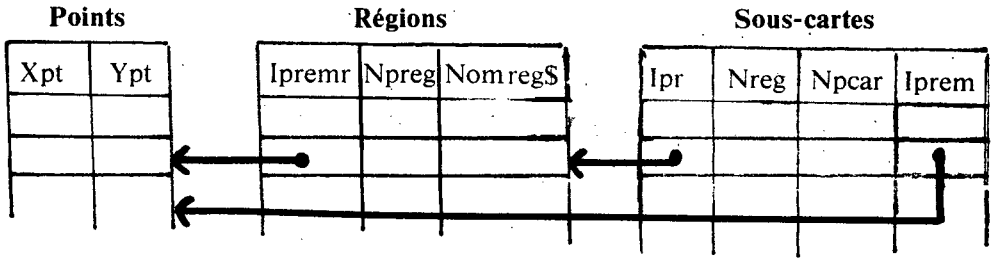
- Par déplacement du réticule sur l'écran avec enregistrement des différents sommets constituant la carte
- En rentrant les coordonnées des sommets par l'intermédiaire du clavier
- Par lecture sur fichier d'un fond de carte préalablement enregistré.

I.1. Structure

Nous voulons obtenir une carte pour la hachurer verticalement, seule l'épaisseur du pas de hachurage variant d'une région à l'autre. Ce qui importe, ce sont les limites des différentes régions la constituant, et par contre il ne nous intéresse pas d'avoir des renseignements topographiques du type rivière, montagne, si ce n'est par les zones qu'ils peuvent constituer. Par conséquent, une carte est un ensemble de zones ou **régions** qui peuvent se regrouper en **sous-cartes**, de telle manière que deux sous-cartes n'ont aucune arête ou frontière commune, par exemple une métropole et ses îles. Une sous-carte pouvant être constituée d'une ou plusieurs régions, par contre une région ne pourra pas s'étendre sur deux sous-cartes. Une région sera définie par un ensemble de **points** constituant les différents sommets du polygone représentant la région. Un point étant défini par ses **coordonnées** dans le plan.

Finalement, nous obtenons la structure suivante :

- Une liste de points avec pour chaque point :
  un numéro
  son abscisse (Xpt)
  son ordonnée (Ypt)
- Une liste de régions avec :
  un numéro
  son nom (Nomreg$)
  le nombre de points la délimitant (Npreg)
  l'indice du premier point dans la liste de points (Ipremr)
- Une liste de sous-cartes avec :
  un numéro
  le nombre de points la délimitant (Npcar)
  l'indice du premier point dans la liste de points (Iprem)
  le nombre de régions délimitées à l'intérieur (Nreg)
  l'indice de la première région dans la liste de régions (Ipr)



Cette structure a l'inconvénient de dupliquer, au minimum, tous les points de la carte. En effet, tout point appartient au moins à deux régions ou à une région et au pourtour d'une sous-carte. Mais elle a l'avantage de limiter les accès tableaux. Par contre, si nous n'avions pas dupliqué les points, il aurait fallu créer tout un système de pointeurs pour retrouver tous les sommets d'une région donnée, alors que dans la structure choisie, nous ne conservons qu'un seul pointeur par région, l'indice du premier point dans les tableaux Xpt, Ypt.

Donc relative perte de place, car les pointeurs occuperaient de la place mémoire, mais gain de temps allié à une grande simplicité.

Les tableaux Ipremr, Npreg, Ipr, Nreg, Npcar, Iprem sont déclarés en **entier**, ainsi que Xpt et Ypt car nous mémorisons les points en coordonnées écran et non utilisateur.

I.2. Algorithme général

L'utilisateur a deux possibilités pour hachurer un fond de carte, soit se servir d'un fond de carte préalablement enregistré, soit en créer un nouveau.

Dans le premier cas, le problème se résume à la lecture sur fichier après avoir indiqué un nom.

Dans le second, il a le choix entre utiliser le réticule qu'il peut déplacer sur écran et entrer les coordonnées par le clavier après avoir indiqué les limites Xmin, Xmax, Ymin, Ymax de son espace, sans se préoccuper des coordonnées écran. La différence se situant à la **lecture des coordonnées**, l'algorithme général restant le même.

Par le choix de structure, les points appartenant à plusieurs régions sont dupliqués. L'utilisateur n'aura pas à rentrer plusieurs fois ces points, ceci sera fait automatiquement à condition qu'il précise, bien entendu, le sens de finition de la région en cours. Par conséquent, il n'aura aucun point à rentrer pour la dernière région de chaque sous-carte.

Il devra tracer le pourtour des sous-cartes dans le sens des aiguilles d'une montre ; puis, si la sous-carte est constituée de plusieurs régions, il devra commencer par délimiter une région qui a au moins deux points communs avec le pourtour, et les premier et dernier points qu'il rentrera devront appartenir à un segment du pourtour de la sous-carte. Ce pourtour constituant la première forme du **périmètre vide**. Ce périmètre vide se modifiant au fur et à mesure de la création de régions, de telle manière qu'à l'intérieur il n'y ait aucune région déjà délimitée et ce jusqu'à ce qu'il coïncide avec la dernière région de la sous-carte. Une nouvelle sous-carte créant et définissant un nouveau périmètre vide sauf si elle est constituée d'une seule région. En effet, le périmètre vide servira à retrouver les autres sommets non rentrés délimitant une région.

```
Prise de fond de carte
début
  Initialisations
  Mpt := Ncar := Nreg ppv 0 !respectivement nbre de pts, sous-cartes et régions
  fin ppv faux
  tq ¬Fin faire
  début
    Ncar ppv Ncar + 1
    lprem(Ncar) ppv Npt + 1
    lpr(Ncar) ppv Nreg + 1
    obtenir pourtour
    Npcar ppv Npt - lprem(Ncar) + 1
    si il y a une seule région alors
      Nreg ppv Nreg + 1
      lpremr(Nreg) ppv lprem(Ncar)
      Npreg(Nreg) ppv Npcar(Ncar)
      Nreg(Ncar) ppv 1
    sinon
      Créer Périmètre Vide
      Ilyaregadel ppv vrai
      tq ilyaregadel faire
      début
        Nreg ppv Nreg + 1
        lprem(Nreg) ppv Npt + 1
        Obtenir Pourtour
        Lire sens de finition
        Terminer Pourtour
        Mise à jour Périmètre Vide
        Mise à jour ilyaregadel
      ftq
      Traiter dernière région
      Nreg(Ncar) ppv Nreg-lpr(Ncar) + 1
    fsi
    Mise à jour Fin
  ftq
fin
```

Obtenir Pourtour consiste à lire les coordonnées, les transformer en coordonnées espace écran et les ranger dans Xpt, Ypt et incrémenter Npt.

De la même manière qu'une région, le périmètre vide sera mémorisé par les coordonnées des sommets le définissant : Xp et Yp, et par le nombre de points Np.

Créer Périmètre Vide : consiste donc à mettre à jour Xp, Yp et Np.

I.3. Finition de la région en cours de création

Cette partie correspond aux trois étapes suivantes de l'algorithme général :

- Lire sens de finition
- Terminer Pourtour
- Mise à jour périmètre vide

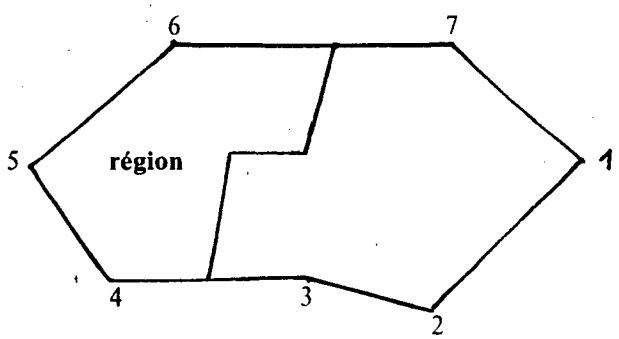
Les points que l'utilisateur vient de rentrer partagent l'intérieur du périmètre vide en deux zones. L'une correspond à la région en cours de création, l'autre représente l'intérieur du prochain périmètre vide. Pour savoir laquelle est la région, et, donc, pour pouvoir **terminer le pourtour** de la région, l'utilisateur doit indiquer un **sens de finition :**

- sens montre : si, en partant du premier point rentré (de la région en cours) et en suivant un des deux pourtours dans le sens des aiguilles d'une montre, on contourne la région
- sens inverse : dans l'autre cas

Pour **terminer le pourtour**, c'est-à-dire retrouver les autres sommets délimitant la région, il faut connaître l'indice du premier point du périmètre vide appartenant à la région, le nombre de points du périmètre vide s'ajoutant à la région et la valeur d'incrémentation (plus ou moins un) dans les tableaux Xp et Yp pour mettre à jour les tableaux Xpt et Ypt.

I.3.1. Cas général

Plaçons-nous dans le cas où le premier et le dernier point indiqué de la région ne correspondent pas à des points du périmètre vide et sont distincts.



Nous pouvons alors déterminer les sommets du périmètre vide tels que :

- le sommet d'indice I1 n'appartienne pas à la région, et le suivant d'indice I1 ⊕ 1 (⊕ addition module Np, nombre de sommets du périmètre vide) appartienne à la région.
- le sommet d'indice I2 appartienne à la région et le suivant d'indice I2 ⊕ 1 n'appartienne pas à la région.

Dans la figure précédente, nous obtenons :

$$I1 = 3 \text{ et } I2 = 6$$

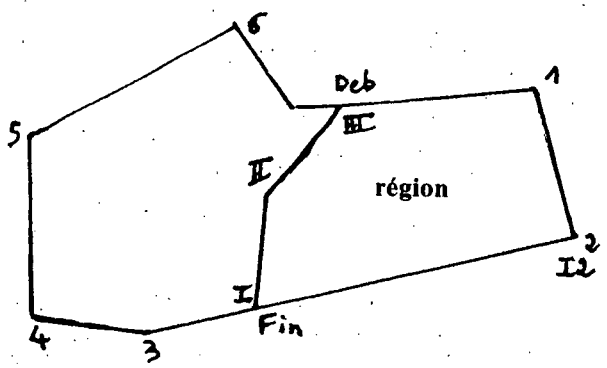
La différence entre I1 et I2 nous donnant le nombre de points qui appartiennent à la région et donc qui disparaissent du périmètre vide. Soit Nsreg ce nombre,

$$Nsreg = I2 \ominus I1 \quad (\ominus \text{ soustraction modulo Np})$$

Quant au nombre Npv de points qui rentrent dans le périmètre vide, il est égal au nombre de points que l'utilisateur a rentré pour délimiter sa région, et finalement Ndif représente le nombre de sommets qui, globalement, apparaissent dans le périmètre vide :

$$Ndif = Npv - Nsreg.$$

Pour trouver I1 et I2, il faut commencer par orienter la ligne de partage de façon à ce qu'elle soit parcourue dans le bon sens en suivant le nouveau périmètre vide. Si le sens de finition indiqué par l'utilisateur est le sens montre, le début est le dernier point rentré, la fin le premier et inversement dans l'autre sens.



I1 = 7 I2 = 2
Mpv = 3
Msreg = 2 ⊖ 7 = 2 − 7 + 7 = 2
Ndif = 3 − 2 = 1

I.3.2. Cas particuliers

1. Le premier et le dernier points de la région sont confondus

Dans ce cas-là, I1 = I2 et aucun point ne disparaît du périmètre vide.
Si ces deux points sont confondus avec un sommet du périmètre, alor un point de moins et un seul apparaît dans le périmètre vide par rapport au nombre de points indiqués par l'utilisateur pour délimiter la région.

2. Le premier ou le dernier point de la région coïncide avec un sommet du périmètre vide

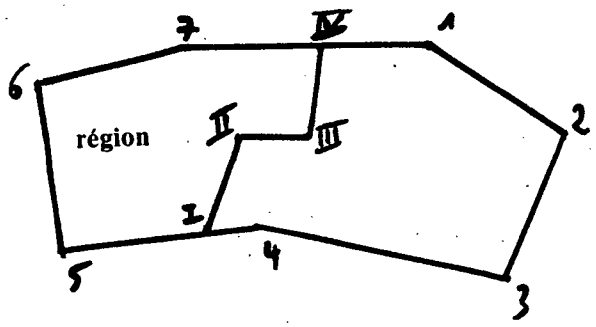
Dans les deux cas, il y a un point de moins qui rentre dans le périmètre vide, que de points indiqués pour délimiter la région.

Si le début de la ligne de partage coïncide avec un sommet du périmètre vide, le premier point nouveau du périmètre vide sera le suivant sur la ligne de partage.

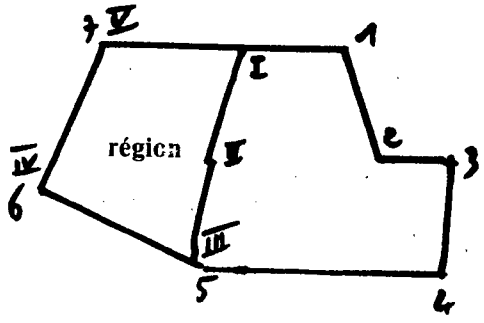
Si la fin coïncide, alors le dernier point nouveau de la région sera le point précédent I2.

I.3.3. Algorithme de Terminer Pourtour

Si le sens de finition et le sens montre, alors le premier point nouveau de la région est le point du périmètre vide d'indice I1 ⊕ 1. Dans l'autre cas, il s'agit du point d'indice I2.



I1 = 4 I2 = 7
Mpv = 4, Msreg = 3, Mdif = 1

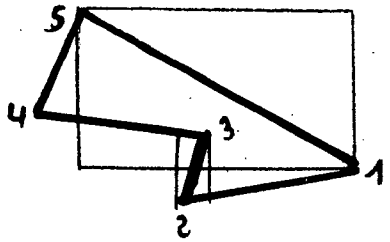


I1 = 5 I2 = 7
Mpv = 3, Msreg = 2, Mdif = 1

Chiffre romain : numérotation de points de la région, chiffre arabe : périmètre vide avant.

Pour déterminer I1 et I2, on fait appel à une procédure Rech-seg-pv qui a, comme paramètres, le périmètre vide (Xp, Yp et Np) et les coordonnées du point que l'on cherche à situer.

Pour éviter trop de calculs, pour un segment donné du périmètre vide, on vérifie si le point est à l'intérieur du rectangle formé par les deux extrémités du segment. Mais ceci n'est pas suffisant, un point du périmètre vide pouvant appartenir à plusieurs rectangles ainsi définis.



```
Terminer Pourtour
début
  si sens de finition = sens montre alors
    Deb ppv Npt
    Fin ppv lpremr(Nreg)
    Indic pp - 1
  sinon
    Deb ppv lpremr(Nreg)
    Fin ppv Npt
    Indic ppv1
  fsi
  Npv ppv Npt - lpremr(Nreg) + 1
  I2 ppv Rech - seg - pv(Xp(*),Yp(*),Np,Fin)
  si I2 = 0 alors le point n'appartient pas au p.v., il faut le corriger
  si Xpt(Deb) = Xpt(Fin) et Ypt(Deb) = Ypt(fin) alors
    I1 ppv I2
  sinon
    Si Xp(I2) = Xpt(Fin) et Yp(I2) = Ypt(Fin) alors
      Npv ppv Npv - 1
      I2 ppv I2 - 1
    fsi
    I1 ppv Rech - seg - pv(Xp(*),Yp(*),Np,Deb)
    si I1 = 0 alors le point n'appartient pas au p.v., il faut le corriger
  fsi
  si Xp(I1) = Xpt(Deb) et Yp(I1) = Ypt(Deb) alors
    Npv ppv Npv - 1
    Deb ppv Deb + Indic
  fsi
  Nsreg ppv I2 - I1
  si Nsreg < 0 alors Nsreg ppv Nreg + Np
  si Indic = - 1 alors
    Ip2 ppvI1 + 1
  sinon
    Ip2 ppv I2
  fsi
  pour I = 1 à Nsreg faire
    si Ip2 = 0 alors
      Ip2 ppv Np
    sinon
      Si Ip2 < Np alors Ip2 = 1
    fsi
    Npt ppv Npt + 1
    Xpt(Npt ppv Xp(Ip2)
    Ypt(Npt) ppv Yp(Ip2)
    Ip2 ppv Ip2 - Indic
  fpour
  Npreg(Nreg) ppv Npt - lpremr(Nreg) + 1
fin
```

Alors, pour ne pas faire intervenir l'utilisateur, la procédure regarde si le point vérifie l'équation du segment.

De plus, si le point recherché coïncide avec un sommet du périmètre vide, la procédure doit retourner l'indice de ce sommet et non l'indice d'un sommet adjacent. Pour être sûr de ce résultat, il faut au départ commencer par tester si le point cherché ne coïncide pas avec le premier sommet du périmètre vide, puis étudier le segment Np-1, puis Np-1-Np, jusqu'au segment 1.2.

```
Rech-seg-pv(fonction entière)(X(*),Y(*);N,Ipt)
début
  si Xpt(Ipt) = X(1) et Ypt(Ipt) = Y(1) alors
    Ir1 := 1
  sinon
    Ir1 := N
    Ir2 := 1
    Trouve := 0
    tq Trouve = 0 et Ir1 < 0 faire
      si(X(Ir1)-Xpt(Iprt))*(X(Ir2)-Xpt(Ipt)et (Y(Ir1)-Ypt(Ipt))*(Y(Ir2)-Ypt(Ipt)) alors
        A := Y(Ir1)-Y(Ir2)
        B := X(Ir1)-X(Ir2)
        C := B*Y(Ir1)-A*X(Ir1)
        Si |A*Xpt(Ipt)-B*Ypt(Ipt) + C| ≤ SQRT(A^2 + B^2) alors T
          Trouve := 1
        fsinon
          Ir2 := Ir1
          Ir1 := Ir1-1
        fsi
      sinon
        Ir2 := Ir1
        Ir1 := Ir1-1
      fsi
    ftq
  fsi
  retour Ir1
fin
```

I.3.4. Mise à jour du périmètre vide

Mettre à jour le périmètre vide consiste à déterminer les points qui restent dans le périmètre vide et à rajouter les points indiqués par l'utilisateur lors de la délimitation d'une région tout en maintenant un ordre de rangement qui coïncide avec un parcours du périmètre vide dans le sens des aiguilles d'une montre.

La principale difficulté consiste à maintenir cet ordre. La solution choisie consiste à limiter le nombre d'accès tableaux et donc à laisser les points à leur rang dans la mesure du possible. Nous allons donc distinguer deux grands cas :

- soit le sommet d'indice 1 dans Xp et Yp reste dans le nouveau périmètre vide
- soit il en sort.

Le sommet d'indice 1 appartient au nouveau périmètre vide

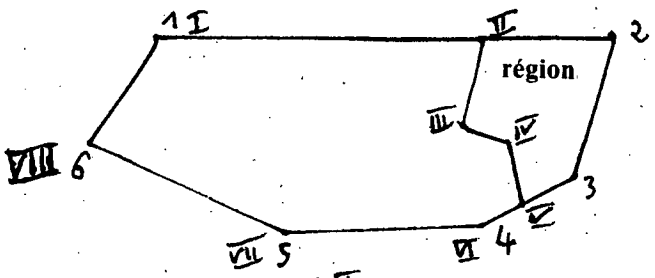
- les I1 premiers sommets restent dans le p.v. avec le même indice
- les Npv sommets de la région rentrent dans le p.v. avec les indices I1 + 1 à I1 + Npv.
- les Nsreg sommets d'indice I1 + 1 à I2 quittent le p.v.
- les Np – I2 derniers sommets restent dans le p.v. avec les indices I1 + Npv + 1 à I1 + Npv + 1 + Np – I2.

Suivant qu'il y a plus de rentrants que de sortants ou inversement, la technique de mise à jour ne sera pas tout-à-fait la même.

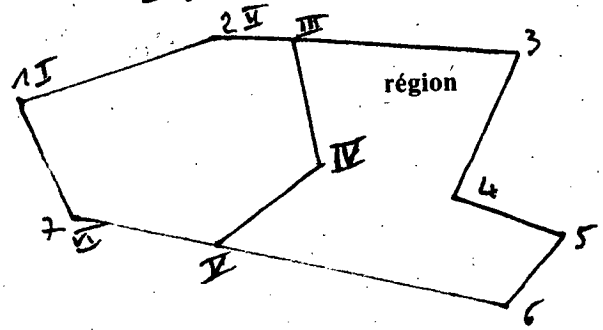
On se trouve dans ce cas, si le sommet 1 ne rentre pas dans la région, c'est-à-dire si I2 ⩾ I1.

Exemples

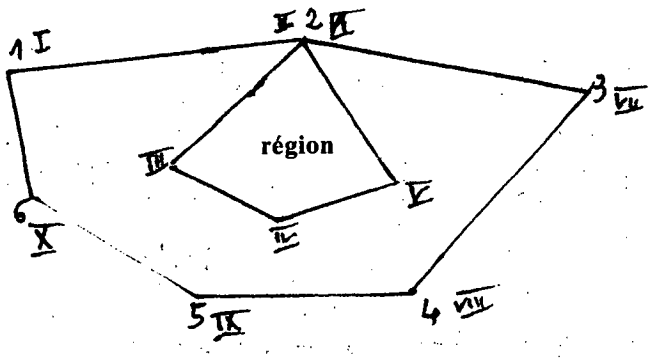
Les chiffres arabes représentent la numérotation des sommets de l'ancien périmètre vide, les chiffres romains les nouveaux.



I1 = 1 I2 = 3
Mpv = 4 Msreg = 2
Ndif = 2



I1 = 2 I2 = 6
Mpv = 3 Msreg = 4
Ndif = – 1



I1 = 2 I2 = 2
Mpv = 4 Msreg = 0
Ndif = 4

Le sommet d'indice 1 appartient à la région

Ce cas se présente quand I2 < I1. En effet, les points du périmètre vide qui appartiennent à la région sont ceux d'indice compris entre I1 + 1 et I2 modulo Np, c'est-à-dire entre I1 + 1 et Np inclus d'une part et 1 et I2 d'autre part.

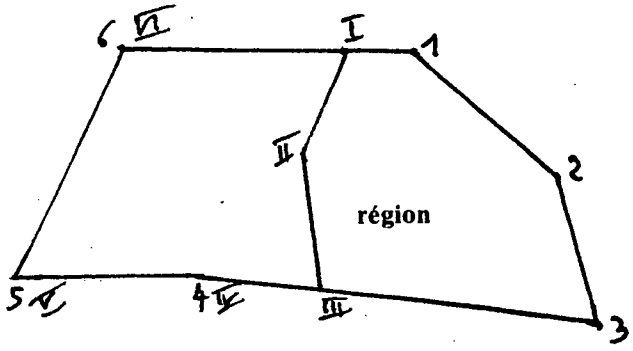
Et les points d'indice I2 + 1, ..., I1 restent dans le nouveau périmètre vide. Et pour que ces points puissent garder le même rang, il faut qu'il y ait au moins I2 points nouveaux à rentrer dans le périmètre vide, c'est-à-dire si Npv ⩾ I2.

Etudions les 3 cas :

- **Npv = I2**

  Dans ce cas, les Npv nouveaux points rentrent dans le nouveau périmètre vide à partir de l'indice 1 jusqu'à l'indice I2. C'est la seule manipulation pour mettre à jour le périmètre vide.

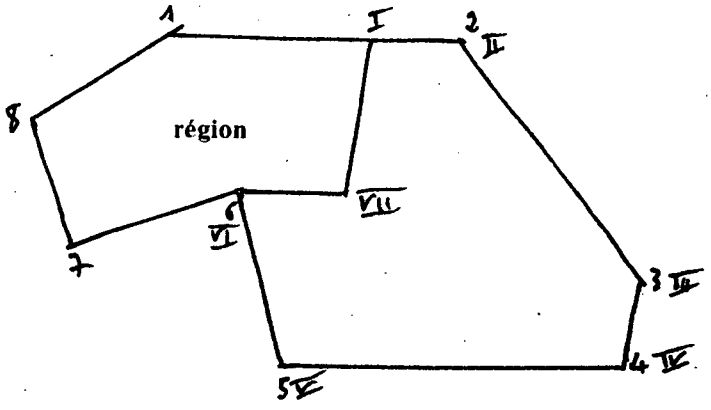
I1 = 6 I2 = 3
Mpv = 3 Msreg = 3
Ndif = 0



- **Npv > I2**

  I2 nouveaux points prennent les rangs d'indice 1 à I2 et les Npv-I2 restants, les rangs d'indice I1 + 1 à Np + Ndif.
  Remarque : Np + Ndif représentant le nombre de points du nouveau périmètre vide.

I1 = 6 I2 = 1
Npv = 2 Nsreg = 3
Ndif = − 1



I.4. Erreurs de l'utilisateur lors de la prise de fond de carte

Une erreur est toujours possible lors de la prise du fond de carte. Comme il n'est pas pensable d'obliger l'utilisateur à recommencer dès le début, il faut lui donner la possibilité de la corriger.

Au départ, il y a deux possibilités : soit lui permettre de corriger son erreur dès qu'il s'en aperçoit en lui permettant d'appeler une procédure de correction à tout moment (réalisable en préservant une touche d'interruption pour cette procédure), soit limiter les possibilités d'appel de la procédure.

La première solution demande un très long travail de recherche pour savoir où reprendre le programme, on a donc décidé d'utiliser la seconde possibilité.

Elle présente l'avantage d'être simple à réaliser. Nous avons décidé que l'utilisateur ne pourrait corriger une erreur qu'après avoir rentré tout un pourtour. Après qu'il ait rentré le dernier point d'une sous-carte ou d'une région, nous lui demandons de valider ce pourtour. Soit il est correct et la mise à jour du périmètre vide s'effectue, soit il y a une erreur et il la corrige à cet instant. Sachant qu'il ne peut corriger que les points du pourtour qu'il vient de rentrer. Il pourra corriger une erreur plus ancienne après avoir rentré tout le fond de carte.

A l'usage, nous pouvons dire que ce n'est pas très contraignant.

Si, après avoir rentré un pourtour, l'utilisateur s'aperçoit d'une erreur, il devra indiquer le point erroné s'il s'agit d'un point à déplacer ou à supprimer, ou une extrémité du segment à modifier s'il veut insérer un nouveau point omis.

Algorithme de correction
début
  si erreur **alors**
    chercher point
    suivant
        1. Point à insérer : Chercher segment erroné
                             Insérer nouveau point dans Xpt et Ypt
                             Npt ppv Npt + 1
      2. Point à supprimer : Supprimer point dans Xpt et Ypt
                             Npt ppv Npt-1
        3. Point à déplacer : Corriger point dans Xpt et Ypt
    fincas
  fsi
fin

Voilà le fond de carte mémorisé, nous pouvons passer à son exploitation : le hachurage.

II. Hachurage

Le hachurage est un cas particulier de remplissage de taches. C'est-à-dire une opération qui consiste à remplir une zone dont la frontière est définie soit par un contour polygonal, soit point à point dans une mémoire image. Dans le cas qui nous intéresse, nous avons vu que les frontières étaient définies par un contour polygonal. De plus, pour garantir le bon fonctionnement de tout algorithme, il est nécessaire que toutes les zones à remplir aient un contour fermé. Or, la méthode de prise de fond de carte présentée dans le paragraphe précédent, nous garantit cette condition. En effet, l'utilisation du périmètre vide et la détermination des indices I1 et I2 (voir I.3.3.) nous assurent que tout contour est bien fermé. Donc, aucun risque d'erreur de ce côté.

Pour remplir une tache, il y a deux techniques principales :

- Remplissage à partir d'un point intérieur
- alayage ligne par ligne.

La première méthode énoncée ne peut pas s'appliquer simplement au hachurage. En effet, elle nécessiterait des calculs à chaque étape pour déterminer s'il s'agit d'une zone éteinte ou allumée suivant d'une part le pas de hachurage, l'épaisseur de la bande de hachurage et l'abscisse (en cas de hachurage vertical) de la ligne en cours d'étude d'autre part. Et ceci, sans parler de la détermination du point intérieur à chaque zone. De plus, cette technique ne peut s'appliquer que pour un remplissage tache par tache, ce qui ne peut convenir à un balayage de l'ensemble des taches. Dans le cas d'un hachurage région par région, elle présente l'inconvénient de ne pas présenter un remplissage régulier dû à des retours en arrière. Il reste donc la technique de balayage ligne par ligne. Technique qui s'adapte parfaitement à un hachurage simultané de toutes les régions qui constituent une carte par balayage de l'écran.

La technique de balayage ligne par ligne se découpe elle-même en trois grands types d'algorithmes qui peuvent se caractériser par :

- Codage des points du contour lors de la numérisation des arêtes. (Algorithme de LUCAS et PAVLIDIS)
- Détermination d'un ensemble de bords gauches et de bords droits dans le cas d'un hachurage horizontal (soit bords supérieurs et inférieurs un hachurage vertical qui nous intéresse) puis tracé des segments qui joignent ces bords (Algorithme XY (NEWMAN et SPROULL), suivi de contour, méthode des jetons)
- Inversions logiques successives de l'image.

Or, d'après l'étude faite par G. HEGRON, il apparaît que la technique de suivi de contours ou la méthode des jetons sont les plus performantes. De plus, ces deux techniques semblaient, a priori, bien s'adapter aux deux méthodes de hachurage que nous proposons à un utilisateur, à savoir un hachurage région par région et un hachurage par balayage de l'ensemble des régions constituant la carte.

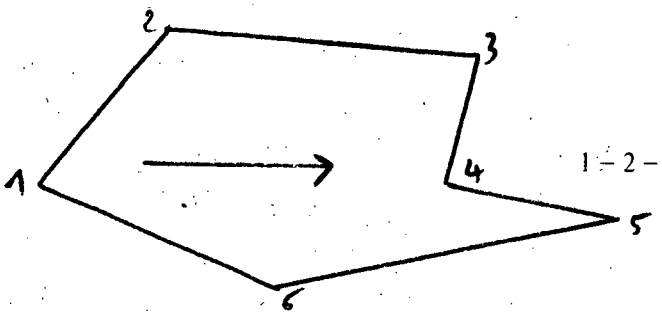
Ces deux techniques présentant l'inconvénient d'avoir une préparation préliminaire relativement importante. La pratique m'a amené à utiliser les deux méthodes qui, comme nous le verrons, sont deux techniques pratiquement identiques, où seul le vocabulaire change.

II.1. Hachurage vertical région par régions

A cette étape, j'ai choisi la méthode dite « **des jetons** ». Elle s'appuie sur le principe qu'entre deux sommets du contour polygonal d'une région la situation sur une ligne se déduit rapidement de celle qu'on avait sur la ligne de balayage précédente. La succession de deux sommets n'étant pas prise au sens ordinaire (deux sommets adjacents sur un contour), mais déterminée par leurs abscisses. La méthode comporte deux étapes distinctes : la préparation et la génération des hachures qui se fait en s'appuyant sur les jetons.

La préparation est relativement importante, mais par contre la génération des hachures est elle très rapide. Mais ceci entraîne un temps d'attente relativement long suivant l'importance de la région. Or, pour un utilisateur qui se trouve devant un écran où rien ne se passe, le temps paraît long. C'est pourquoi j'ai limité la préparation au strict minimum :

Les sommets sont triés suivant les valeurs croissantes de leurs abscisses, ce qui donne l'ordre suivant dans le cas de la figure :



1 – 2 – 6 – 4 – 3 – 5

Mais que se passe-t-il, lorsque deux points consécutifs du pourtour ont même abscisse? En étudiant les différents cas, on s'aperçoit qu'il s'agit de cas particuliers des trois situations suivantes :

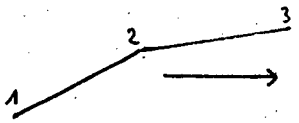


fig. 1.1

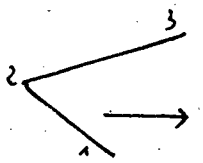


fig. 1.2

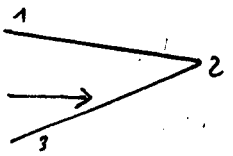


fig. 1.3

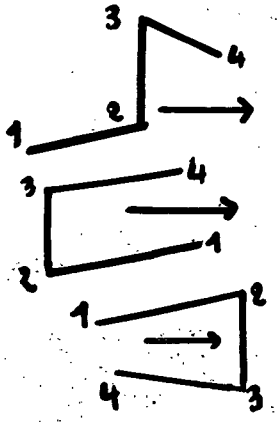
La flèche indique le sens de déplacement du jeton.

Figure 1.1. : le jeton qui se déplaçait sur le segment joignant les sommets 1 et 2 disparaît en atteignant le sommet 2 et est remplacé par le jeton se déplaçant sur le segment joignant 2 et 3. Il y a un **changement de jetons.**

Figure 1.2. : aucun jeton n'atteint le sommet 2, il faut donc **créer deux nouveaux jetons** qui se déplaceront sur les segments joignant 2-3 d'une part et 2-1 d'autre part.

Figure 1.3. :Deux jetons, se déplaçant sur le segment 1-2 d'une part et 3-2 d'autre part, atteignent en même temps le sommet 2. Et aucun ne peut continuer son chemin : il y a donc **suppression de deux jetons.**

Cas particuliers de ces configurations



Dans ce cas de figure, il est bien évident qu'il s'agit d'un **changement** de jeton. Le jeton se déplaçant sur le segment 1-2, en atteignant le sommet 2, il continuera son déplacement sur le segment joignant les sommets 3-4.

Ici, on retrouve le cas d'une **création** de deux nouveaux jetons. Un jeton se déplaçant sur le segment 3-4, l'autre sur le segment 2-1.

Et dans ce dernier cas particulier, on retrouve le problème de la suppression de deux jetons. Un se déplaçant sur le segment joignant les sommets 1 et 2, l'autre sur le segment 4-3.

Dans tous les cas, lorsque deux sommets adjacents du pourtour ont la même abscisse, le problème est identique au cas correspondant avec un seul point. Par conséquent, lors du tri des sommets du pourtour, nous ne conservons qu'un seul point sur tous ceux ayant même abscisse et étant consécutifs sur le pourtour. Sur les Npreg points que comporte la région, seulement Nrg seront triés.

Epaisseur est un indicateur qui permet de savoir si il faut incrémenter l'abscisse de la ligne de hachurage d'une unité, tant qu'on a pas atteint l'épaisseur de la bande de hachurage pour la région en cours, ou autrement lorsqu'on la dépasse.

Dans ce second cas, l'incrémentation de l'abscisse de hachurage sera telle que la ligne se déplacera soit jusqu'à la bande suivante, soit au sommet suivant selon que l'abscisse de l'un ou l'autre est inférieure.

Le premier sommet de l'ordre établi aura un traitement différent dans la mesure où on est sûr qu'en ce point il faudra créer les deux premiers jetons.
Mettre à jour les jetons consiste à appliquer le déplacement vertical suivant la pente sur laquelle il glisse. Ce qui demande d'avoir pour chaque jeton son ordonnée et la pente du segment qu'il parcourt.

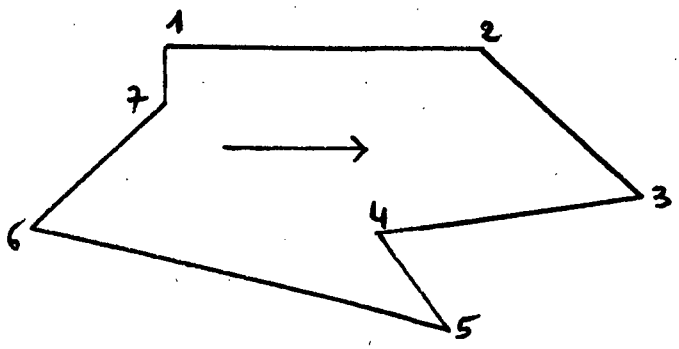
Pour traiter un nouveau sommet, il est nécessaire de connaître les sommets adjacents du pourtour d'abscisse différente de la sienne. Suivant les cas, il faudra changer un jeton ou en créer ou supprimer deux.

```
Traiter sommet (d'adresse Icour)
début
  Déterminer Pred et Suiv tel que :
      -Pred < Icour < Suiv (ordre module le nombre de sommets de la région)
      -Xpt(Pred)/Xpt(Icour)/Xpt(Suiv)
  Pred ppv Pred⊕1 (⊕ addition modulo le nombre de sommets)
  Suivm ppv Suiv⊖1
  si (Xpt(Pred)-Xpt(Icour))*(Xpt(Suiv)-Xpt(Icour))<0 alors
      Changement de jeton :remplacer le jeton qui arrivait sur Predppar celui qui
                           part de Suivm
  sinon
      si Xpt(Pred)\Xpt(Icour)alors
      Création de 2 jetons : les insérer devant le premier jeton dont l'ordonnée
                             actuelle est inférieure aux ordonnées des 2 jetons qui
                             partent de Predp et Suivm
      sinon
        Suppression des 2 jetons qui arrivent sur Predp et Suivm
      fsi
  fsi
  Icour ppv Suiv(Icour)
fin
```

II.1.1. Algorithme général

```
    Hachurage région par région (principe de la méthode des jetons)
début
  pour chaque région de la carte faire
    Initialisations
    Ordonner les sommets et déterminer Nreg
    Traiter premier sommet
    pour I ppv 2 à Nreg faire
      tq l'abscisse de la ligne de hachurage < l'abscisse du sommet SUIVANT
         faire
        si Epaisseur < Epaisseur(Ireg) alors
          Visualiser la barre de hachurage
          Incrémenter l'abscisse de la ligne de hachurage d'une unité
          Epaisseur ppv Epaisseur + 1
          Mettre à jour les jetons
        sinon
          Incrémenter la ligne de hachurage autant que possible
          Mettre à jour les jetons et Epaisseur
        fsi
      ftq
      Traiter sommet SUIVANT
      Mettre à jour SUIVANT
    pour
  fpour
fin
```

Ordonner les sommets consiste à tirer les sommets de la région en ne conservant que les sommets qui sont tels que le sommet suivant sur le pourtour a une abscisse différente.



Ordre obtenu : 6 – 7 – 4 – 5 – 2 – 3

Et pour visualiser, la barre de hachurage, il suffit de prendre les jetons par couple et de tracer, sur la ligne de hachurage, un segment reliant les deux ordonnées, puis de passer au couple suivant avec deux nouveaux jetons.
Nous avons obtenu un algorithme correct de hachurage dont le bon fonctionnement est garanti par la certitude de travailler sur des zones au pourtour fermé et non croisé.

Seulement, pour obtenir que les temps d'attente, où rien ne se passe sur l'écran, ne soient trop longs, nous arrivons à multiplier le travail, et donc à faire que le hachurage soit plus long dans son ensemble. La partie incriminée de l'algorithme est celle qui concerne la détermination de pred et de suiv, c'est-à-dire les indices des points du contour qui précèdent et suivent le point courant tout en n'ayant des abscisses différentes à celles du point courant. Il est clair que le faire pour chaque point entraîne un double travail. En effet, si deux points sont tels que : I1 = Icour et I2 = Pred, alors on a également I1 = Suiv et I2 = Icour.

Finalement, il serait beaucoup plus performant de déterminer ces Pred et Suiv dans la procédure ordonner, en créant des tableaux supplémentaires par exemple Droite et Gauche, qui pour le ième point de la région, seraient tels que :

Droite(i) = Pred et Gauche(i) = Suiv avec i = Icour.

II.2. Hachurage vertical par balayage horizontal de l'ensemble des régions

La méthode des jetons est très pratique pour hachurer une seule région avec son système de jetons que l'on prend deux par deux, en traçant, sur la ligne de hachurage, le segment qui les relie. Par contre, si on prend l'ensemble des régions, le problème est un peu différent dans la mesure où en utilisant des jetons, on a la situation suivante :

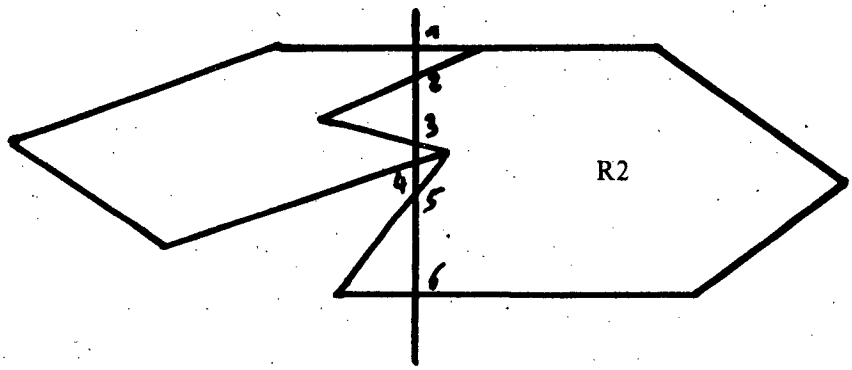


Figure 2

La figure représente une carte simplifiée composée de deux régions. Le trait vertical représente la ligne de hachurage en cours de réalisation. Les croix indiquent la position des jetons.

Si le hachurage se fait région par région :

- Pour R1 : jetons 1 et 2, 3 et 4 ; tracé entre 1 et 2 d'une part et 3 et 4 d'autre part
- Pour R2 : jetons 2, 3, 5 et 6 ; tracé entre 2 et 3, puis 5 et 6.

Pour un hachurage par balayage, le problème se résume à tracer entre 1 et 2, 2 et 3, 3 et 4 et 5 et 6. Ce qui entraîne une certaine complexité à s'y retrouver. J'ai donc préféré étudier une autre méthode, celle du suivi de contour.

II.2.1. Le suivi de contour

Si la méthode des jetons s'intéresse aux sommets du contour et fait glisser des jetons entre ces sommets, la méthode par suivi de contour s'intéresse aux arêtes qui constituent le pourtour.

Une arête est définie par les sommets qui la délimitent. Le travail de préparation de la méthode consiste à trier les arêtes et non plus les sommets. On commence par orienter les arêtes. Pour un hachurage vertical par balayage horizontal, on oriente les arêtes dans le sens du sommet d'abscisse le plus faible vers le sommet de plus grande abscisse, comme sur la figure 3.

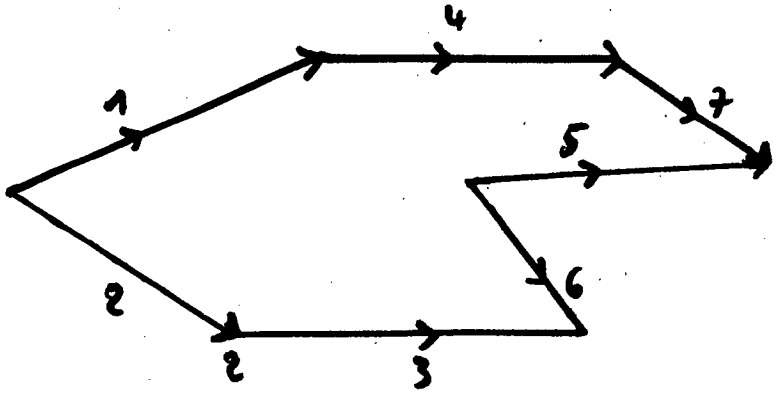


Figure 3

Puis, on trie les arêtes d'abord sur l'abscisse de leur origine. A abscisses égales, le tri se fait sur l'ordonnée de leur origine et à ordonnée égale, le tri se fait sur la pente. Les arêtes A1 et A2 ayant même origine, A1 précède A2 si la pente de A1 est supérieure à celle de A2. Les chiffres de la figure 3 indiquant l'ordre des arêtes.

Pour le hachurage proprement dit, on ne s'intéresse qu'aux arêtes que la ligne de hachurage coupe.

De plus, une arête n'est commune qu'à deux régions, en considérant que l'extérieur du pourtour d'une sous-carte forme une région limitée par la taille de l'écran. Et pour hachurer par balayage de l'ensemble des régions, on peut savoir en coupant une arête de quelle à quelle région on passe. Par contre, un point peut appartenir à plus de deux régions et ceci peut poser quelques problèmes dans la méthode des jetons.

Mais utiliser la méthode de suivi de contour, soulève un problème. Lors de la prise de fond de cartes, nous avons vu que l'élément de base de mémorisation était le point. Or avec le suivi de contour, nous ne nous intéressons pas aux points mais aux arêtes que « forment » deux points distincts. Il est donc important, à ce stade, de s'interroger sur la structure à choisir et, donc, de peut-être remettre en cause la mémorisation d'un fond de carte par les points que constituent les différents sommets.

II.2.2. Points ou Arêtes

Maintenant que nous avons choisi de traiter le problème du hachurage par balayage de l'ensemble des régions par les arêtes, ne faut-il pas stocker un fond de carte en stockant les arêtes qui le constituent.

Sur le plan de l'occupation mémoire, les deux méthodes sont identiques. En effet, une arête se définit par les coordonnées de ses extrémités. Or, tout sommet appartient à deux arêtes et par conséquent les données relatives à ce point sont stockées en double. Or, nous avons vu que une région était mémorisée par l'ensemble de ses points, et de même tout point appartient à deux régions ou une sous-carte et une région. Nous avons donc une occupation mémoire identique.

Mais nous avions dupliqué les points pour faciliter une éventuelle recherche d'un point appartenant à cette région, et pour faciliter l'affichage du fond de carte région par région. Affichage rapide et agréable à suivre lorsqu'on a la liste de tous les sommets délimitant une région. Par contre, en ne conservant que les arêtes, nous n'avons plus cet affichage. Soit l'affichage est lent, soit il est dispersé : des arêtes s'inscrivant dans n'importe quel ordre, dans la mesure où une arête peut être coupée en deux ou plus lors de la description d'une nouvelle région. Ce côté présentation n'étant pas négligeable, il est apparu préférable de conserver les sommets pour chaque région. Bien que ce choix entraîne une plus grande occupation mémoire dans la mesure où des renseignements relatifs aux arêtes devront être stockés. Donc, on a finalement Points et Arêtes.

Nous avons vu que dans le travail préliminaire au hachurage par suivi de contour, il fallait trier les arêtes après les avoir orientées. Ce qui importe surtout, c'est d'avoir plusieurs listes. Chaque liste est composée d'arêtes ayant même abscisse de leur origine et chaque liste est alors triée suivant l'ordonnée de leur origine et leur pente.

Pour chaque arête, les renseignements nécessaires sont donc l'ordonnée de leur origine, l'abscisse de leur point final et la pente. De plus, comme il s'agit du balayage de l'ensemble des régions, il faudra l'indice des deux régions qu'elle sépare. Et pour chaque liste d'arêtes, on conservera l'abscisse de l'origine de chaque arête.

De plus, vu le nombre important d'arêtes, il n'est pas possible de trier les arêtes directement dans les tableaux, il faudra donc un pointeur sur l'arête suivante. Ce qui donne la structure suivante.

Arêtes

Y	Xfin	Reg1	Reg2	Pente	Suiv

Listes

Abscisse	Adresse du premier

II.2.3. Préparation pour le hachurage

Cette préparation concerne donc les arêtes. C'est un travail long qui demande beaucoup de comparaisons (orientation des arêtes et leur tri) et le calcul de la pente pour chaque arête. Alors, pour qu'il soit relativement transparent à l'utilisateur, il doit être effectué en même temps que la prise du fond de carte, mais ceci le rend plus compliqué du fait des erreurs éventuelles et des modifications.

Tous les renseignements nécessaires à une arête vont s'obtenir à deux périodes différentes de la prise du fond de carte. Le plus gros des renseignements est obtenu lors de la prise du pourtour d'une sous-carte ou d'une région, ce sera la partie création de l'arête et mise en place de la liste correspondante. Le dernier renseignement concernant la deuxième région que sépare l'arête s'obtiendra lors de la finition automatique de la région.

Les arêtes verticales ne seront pas prises en compte. En effet, celles-ci sont parallèles aux lignes de hachurage et il n'est donc pas utile de les prendre en compte

puisque ce sont des arêtes qui disparaîtront en passant à la ligne suivante de hachurage. Par conséquent, si le pourtour des régions apparaît sur l'écran, il est inutile de les retracer et si le pourtour est éteint, une arête verticale sera considérée comme étant à l'intérieur de la région située à sa gauche vers les abscisses croissantes.

Pour chaque pourtour, à partir du deuxième point, on fait appel à une procédure **Traiter-arête** si l'abscisse du point courant et celle du point précédemment rentré sont différentes.

Cette procédure a pour paramètres les coordonnées du point courant et du point précédent, le numéro de la région en cours (– 1 si le pourtour est celui d'une sous-carte pour le distinguer) et le nombre d'arêtes créées.

```
Traiter arête (entier X1, Y1, X2, Y2, N1, N2, Na)
début
  Na ppv Na + 1                ! 1 arête supplémentaire
  Y(Na) ppv ordonnée de l'origine
  Xfin(Na) ppv abscisse du point final – 1
  Pente(Na) ppv (Y1 – Y2)/(X1 – X2)
  Reg1(Na) ppv N1
  Reg2(Na) ppv N2
  Insérer arête Na dans la liste des arêtes ayant même abscisse que son
  origine
fin
```

Pour chaque région, lors de la finition, il faut mettre à jour Reg2 après avoir recherché l'arête déjà créée.

On a vu que lors de la délimitation d'une nouvelle région à l'intérieur du périmètre vide, le premier et dernier point indiqués doivent appartenir au périmètre vide, sans toutefois coïncider obligatoirement avec un sommet. Ce qui entraîne qu'une arête existante peut être coupée en deux. Dans ce cas, il faut modifier l'arête et dans le cas général, en créer une nouvelle (Figure 4).

Il se peut qu'une arête ayant ses extrémités d'abscisses différentes, en la coupant en deux, un des deux segments obtenus soit vertical. Ceci est la conséquence d'une surface d'affichage point par point. Et plus la pente, en valeur absolue, de l'arête originale est accentuée, plus le risque est grand d'obtenir une arête verticale et une arête oblique avec une pente encore plus accentuée.

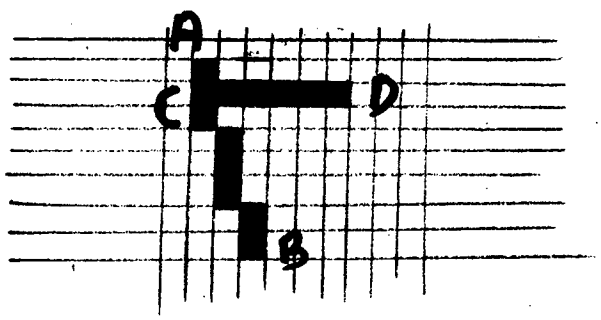


Figure 4

L'arête AB est coupée par l'arête CD, à partir de l'arête AB nous obtenons deux nouvelles arêtes AC et CB. Or, l'arête AC est verticale, donc la modification de AB doit se résumer à remplacer dans la liste des arêtes ayant pour abscisse XA, l'arête AB par l'arête CB, tout en vérifiant sa place dans la liste. En effet, imaginons que A soit une pointe et qu'en plus de AB soit issue une autre arête AE de pente encore plus accentuée. Dans ce cas, AB précède AE dans la liste, mais du coup AE précèderait CB puisque l'ordonnée de l'origine A est plus grande que celle de C, origine de CB.

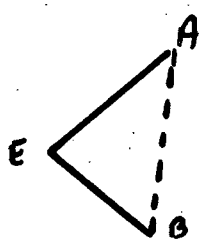
```
Modifier arête (Entiers X1, Y1, X2, Y2, X3, Y3)
début
    ! X1, Y1, X2, Y2 coordonnées des extrémités de l'arête
    ! X3, Y3 coordonnées du point d'intersection
  Chercher arête (X1,Y1)-(X2,Y2)
  si X1 = X3 alors ! arête verticale
    mettre à jour Y et Pente
    rechaîner la liste
  sinon
    mettre à jour Xfois et Pente
    si X1 ≠ X2 alors Appel traiter-arête
  fsi
fin
```

Voici pour le travail préliminaire, sans oublier toutefois les corrections à apporter aux arêtes lors de la correction d'un point par l'utilisateur.

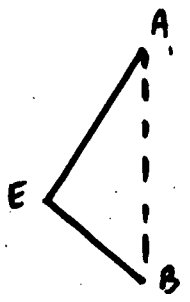
Dans le cas de la suppression d'un point, il y a à modifier une arête et en supprimer une (Fig.5.1) ou en supprimer deux (Fig.5.2.). Pour une insertion d'un nouveau point, soit modifier une arête et en créer une nouvelle (Fig.6.1.) soit modifier une seule arête (Fig.6.2.), soit créer deux arêtes (Fig.6.3.). Et pour un déplacement, plusieurs cas se présentent aussi. Modifier deux arêtes (Fig.7.1.), modifier une arête et en créer une (Fig.7.2.) ou bien créer deux nouvelles arêtes (Fig.7.3.) ou encore modifier une seule arête (Fig.7.4.).

Dans les figures 5 à 7 le point E désigne le point erroné et le point C le point corrigé. En trait plein sont tracées les arêtes avant correction.

Suppression

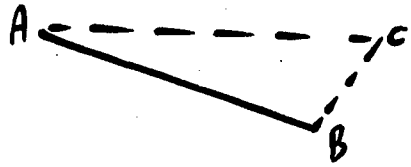


modifier AE, supprimer EB
fig. 5.1

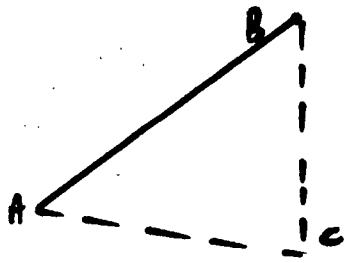


supprimer AE et EB
fig. 5.2

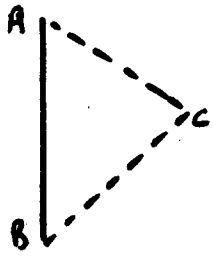
Insertion



modifier AB par AC, créer BC
fig. 6.1

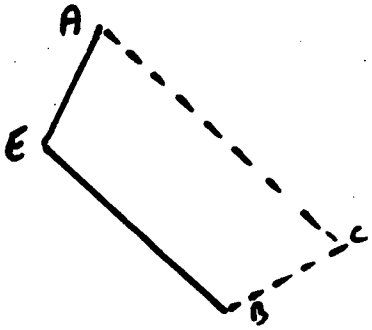


modifier AB par AC
fig. 6.2

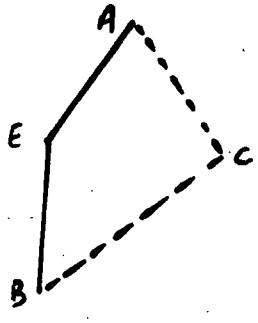


créer Ac et BC
fig. 6.3

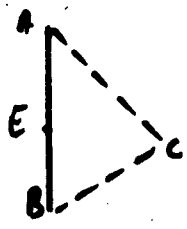
Déplacement



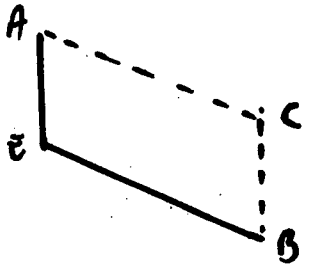
modifier AE par AC
remplacer EB par BC
fig. 7.1



modifier AE par AC
créer BC
fig. 7.2



créer AB et BC
fig. 7.3



remplacer EB par AC
fig. 7.4

Maintenant que le travail préliminaire est fait, tout en restant transparent à l'utilisateur puisqu'il se fait en même temps que la prise de fond de carte, il sera enregistré sur fichier en même temps que le fond de carte. Le hachurage de la carte pourra être réalisé relativement rapidement.

II.2.4. Hachurage par balayage - Algorithme

Pour une ligne de hachurage donnée, le travail se résume à peu de chose : soit liste courante, la liste de toutes les arêtes coupées par la ligne de hachurage, soit X l'abscisse de la ligne de hachurage et soit Liste X, la liste des arêtes ayant leur origine sur la ligne d'abscisse X. Il faut fusionner la liste courante et la liste X puis traiter la nouvelle liste courante obtenue. Et on obtient l'algorithme suivant à condition de supprimer les arêtes qui se terminent en X au pas X − 1.

hachurage suivi de contour
début
  X_o ppv Xmin de tous les sommets de la carte
  X_n ppv Xmax de tous les sommets de la carte
  Liste X_n ppv liste nulle (≠ de liste vide)
  Liste courante ppv Liste X_o (Prem ppv Act(X_o))
  tq Liste courante non vide **faire**
  début
    Vider liste X_o
    tq liste X_o est vide **faire**
    début
      Traiter liste courante et supprimer arête finie en X_o + 1
      X_o ppv X_o + 1 :Epaisseur ⊕ 1 (⊕ addition modulo Pas)
    fin
    si liste X_o non nulle **alors** fusionner liste courante et liste X
  ftq
fin

Pour distinguer liste vide de liste nulle, on utilisera le tableau Adr qui pointe pour une abscisse X donnée sur la liste des arêtes ayant leur origine sur la ligne d'abscisse X. La liste X sera non vide lorsque Adr(X) sera positif, elle sera vide lorsque Adr(X) sera nul et elle sera nulle lorsque Adr(X) sera égal à − 1. Et pour vider une liste, il suffira d'annuler l'adresse correspondante (Adr(X) ppv 0). Rendre nulle liste X , ne pose aucun problème dans la mesure où, ne tenant pas compte des arêtes verticales, liste X est vide.

Fusionner liste courante et liste X se fera en maintenant l'ordre établi sur les arêtes : trier sur les ordonnées des points d'intersection avec la ligne de hachurage dans un ordre décroissant et à égalité trier sur les pentes décroissantes.

Traiter liste courante et supprimer arête finissant en X + 1 se décompose en deux parties. La première vérifie s'il faut ou non joindre les deux arêtes en fonction de la région traversée et de l'épaisseur associée à cette région, puis pour chaque arête étudier si elle reste ou non dans la liste courante. Pour cette dernière étape, il suffit de comparer Xfin qui est égale, rappelons-le, à l'abscisse précédant la fin de l'arête, à X , abscisse courante de la ligne de hachurage et on obtient l'algorithme suivant :

```
Traiter liste courante
Début
  Ipred ppv 0
  Ir ppv Prem !Prem étant l'adresse de la première arête de liste courante
  Tq Ir ≠ 0 faire
    si lreg = - 1 alors
      Aller plume haute en X0, Y(Ir)
    sinon
      si epaisseur (lreg) < Epaisseur alors
        Aller plume haute en X0, Y(Ir)
      sinon
        Tracer plume basse en X0, Y(Ir)
      fsi
    fsi
    si lreg = Reg1(Ir) alors !Mise à jour de région courante
      lreg ppv Reg2(Ir)
    Sinon
      lreg ppv Reg1(Ir)
    fsi
    si Xfin(Ir) = X0 alors
      si Ipred = 0 alors
        Prem ppv Suiv(Ir)
      sinon
        Suiv(Ipred) ppv Suiv(Ir)
      fsi
    sinon
      Y(Ir) ppv Y(Ir) + Pente(Ir)
      Ipred ppv Ir
    fsi
    Ir ppv Suiv(Ir)
  ftq
fin
```

Dans cet algorithme, deux étapes sont primordiales. La première concerne la détermination de la région dans laquelle on se trouve. Tout au début du hachurage, il faut considérer que l'on se trouve à l'extérieur des sous-cartes et que par conséquent il ne faut pas afficher de segment. Ensuite, après avoir franchi une arête, il faut comparer la région courante, déterminée par lreg, rang de la région dans la liste des régions, aux deux régions que sépare l'arête rencontrée. Et lreg, région courante, est mise à jour en prenant le numéro de région différent de celui de la région précédente. Et comme la dernière arête de la liste courante est nécessairement une arête séparant une région de l'extérieur de la sous-carte, après avoir franchi cette dernière arête, on retrouve un numéro de région (égal à - 1) indiquant qu'on est à l'extérieur de la carte. On est prêt pour attaquer une nouvelle liste courante dont la première arête séparera une région de l'extérieur de la sous-carte.

II.3. Retour au hachurage région par région

Nous avons utilisé deux méthodes de hachurage, l'une **méthode des jetons** pour le hachurage région par région, l'autre **suivi de contour** pour le hachurage par balayage de l'ensemble des régions. Il est clair que ces deux méthodes sont identiques bien que présentées différemment. L'une se basant sur le fait que l'information élémentaire est le point, l'autre l'information élémentaire est l'arête. La première s'occupe des jetons qui glissent entre deux points, la seconde tient compte des intersections des arêtes avec la ligne de hachurage. Mais il est clair que le jeton se situe à l'intersection de la ligne de hachurage et du segment joignant deux sommets et une arête se définit bien comme un segment joignant deux points.

Par conséquent, excepté la terminologie employée, les deux méthodes sont identiques.

A partir de cette constatation, il semble aberrant de refaire tous les calculs pour le hachurage région par région. Il faut donc se servir du travail préliminaire au hachurage par balayage de l'ensemble des régions, pour hachurer l'ensemble de la carte région par région.

L'algorithme général **hachurage suivi de contour** s'applique au hachurage région par région, pour chaque région de la carte. Mais, avant de l'appliquer, il est nécessaire de retrouver les arêtes délimitant le pourtour de la région. Du fait des structures adoptées, la recherche se fera en comparant les numéros de région Reg1 et Reg2 des arêtes au numéro de la région en cours de traitement. La comparaison ne se fera pas sur l'ensemble des arêtes de la carte, mais seulement sur les listes d'arêtes correspondant aux sommets de la région.

De plus, comme le traitement détruit le chaînage des arêtes et modifie l'ordonnée de l'origine de chaque arête, il est nécessaire d'adopter une nouvelle structure, constituée des mêmes éléments exceptés Reg1 et Reg2 mais de noms différents. Les renseignements qui se modifient au cours du traitement sont Adr, Suiv et Y, on les remplacera par Adreg, Sreg et Yreg pour ne pas être obligé de relire sur fichier ces renseignements à chaque hachurage de nouvelles régions.

II.3.1. Préparation au hachurage région par région

La préparation se limite à retrouver les arêtes constituant le pourtour de la région lreg en cours de traitement.

```
Recherche arêtes de lreg
début
  lpt ppv lpremr (lreg)
  pour I ppv 1 à Npreg(lreg) faire !pour tout point de la région
  début
    si liste Xpt(lpt) non vide alors
      lpred ppv 0
      lr ppv Adreg(Xpt(lpt))
      tq lr ≠ 0 faire
      début
        si Reg1(lr) = lreg ou Reg2(lr) = lreg alors
          si lpred = 0 alors
            Adreg(Xpt(lpt)) ppv lr
          sinon
            Suiv(lpred) ppv lr
          fsi
          Yreg(lr) ppv Y(lr)
        fsi
        lr ppv Suiv(lr)
      ftq
    fsi
      lpt ppv lpt + 1
  fpour
fin
```

Dans l'algorithme général, la partie **traiter liste courante** devient plus simple.

II.3.2. Traiter liste courante dans le cas du hachurage région par région

On peut se permettre de traiter les arêtes deux par deux, car on est sûr d'avoir un nombre pair d'arêtes, et il n'y a plus à comparer de numéro de régions puisqu'on connaît la région en cours de traitement. Nous obtenons alors l'algorithme de Traiter liste courante suivant.

```
Traiter liste courante - hachurage région par région
début
  si Epaisseur(lreg) < Epaisseur alors
       mettre la plume en position haute
  sinon
       en position basse
  fsi
    lpred ppv 0
    lr ppv Prem
  tq lr ≠ 0 faire
       si Xfin(lr) + X0 alors
          Supprimer lr de liste courante
       sinon
          Yreg(lr) ppv Yreg(lr) + Pente(lr)
          lpred ppv lr
       fsi
       lr ppv Sreg(lr) aller en X0,Y(lr)
       si Xfin(lr) = X0 alors
          Supprimer lr de liste courante
       sinon
          Yreg(lr) ppv Yreg(lr) + Pente(lr)
          lpred ppv lr
       fsi
       lr ppv Sreg(lr)
       Tracer plume haute en X0, Y(lr)
  ftq
fin
```

III. Evaluation des algorithmes de hachurage par la méthode du suivi de contour

Soit D la distance entre les deux sommets extrêmes de la carte, distance projetée sur l'axe horizontal.

Soit Nlnv le nombre de listes non vides entre Xmin et Xmax. Ceci décompte toutes les lignes de hachurage qui rencontreront un sommet de la carte.

Hachurage suivi de contour	
début	
Adr(Xmax) ppv − 1	1(ppv,AT)
Prem ppv Adr(X_o)	1(ppv,AT)
Tq Prem ≠ 0 **faire**	(Npnv + 1)?
Début	
Adr(X_o) ppv 0	Npnv(ppv,AT)
Tq Adr(X_o) = 0 **faire**	(D + Npnv)(?,AT)
début	
Traiter liste courante	D
Ep ppv Ep + 1	D(ppv, +)
Si Ep > Pa **alors** Ep ppv 1	D?,(D/Pas)ppv
X_o ppv X_o + 1	D(ppv, +)
ftq	
Si Adr(X_o) > 0 **alors** Fusion	Nlnv?,(Nplnv-1)fusion
ftq	
Fin	

Les notations correspondent à :
- ppv : affectation
- AT : accès tableau
- ? : comparaison
- + : opération arithmétique (addition)

Le test de la liste courante vide se fait à chaque fois qu'on est tombé sur une liste X_o non vide, plus une fois pour la dernière liste nulle, soit (Nlnv + 1) tests. Par conséquent, vider liste X_o et le test sur liste X_o non nulle sont effectués Nlnv fois.

Le corps du **tant que** intérieur est effectué pour chaque ligne de hachurage compris entre Xmin et Xmax, soit D fois. L'affectation Ep ppv 1 est réalisée à chaque fois que l'épaisseur depuis la dernière bande de hachurage est supérieure au pas du hachurage, par conséquent (D/Pas) fois à une unité près.

Le test de ce **tant que** est donc effectué D fois plus le nombre de fois que l'on rencontre une liste non vide. Soit en tout (D + Nlnv) fois.

Et on obtient :

2D + D/Pas + Nlnv + 2	ppv	7D + 7Nlnv + D/Pas + 5
D + 3Nlnv + 2	AT	
2D + 3Nlnv + 1	?	
2D	+	

et D fois traiter liste courante (Nlnv-1) fois fusion de liste courante et liste X_o

Soit D_i et $Nlnv_i$ la longueur et le nombre de listes non vides de la région i. Pour chaque région i, on a donc sans compter traiter-liste courante ni la fusion :

$7D_i + 7Nlnv_i + Di/Pas + 5$ soit un total de :

$$\sum_{i=1}^{Nreg} (7D + 7Nlnv_i + D_i/Pas + 5)$$

Le problème est de comparer D et $\sum D_i$ et Nlnv et $\sum Nlnv_i$. Les seules affirmations qu'on puisse faire sont celles-là :

$$D \leqslant \sum D_i \text{ et } Nlnv \leqslant 2 \sum Nlnv_i$$

L'égalité entre D et $\sum D_i$ est atteinte lorsque la carte est découpée en lamelles, ce qui doit se présenter très rarement, pour ne pas dire jamais. Ce qu'on peut dire également, c'est qu'en général plus il y a de régions, plus le rapport $\sum D_i/D$ est grand. Par exemple, pour la France et ses 22 régions, le rapport est égal à 4,3 alors que pour la France et ses départements le rapport tourne autour de 10. On peut en conclure que le rapport n'est pas proportionnel au nombre de régions, ce qui était bien évident. Si le meilleur des cas, égalité, est atteint lorsque la carte est découpée en lamelles verticales, le pire des cas est obtenu lorsque la carte est découpée en tranches horizontales et dans ce cas, on a : $\sum D_i = Nreg*D$. Par conséquent, le rapport $\sum D_i/D$ est fonction de la configuration de la carte et fonction croissante du nombre de régions.

Nous pouvons quand même dire que $\sum D_i$ est plusieurs fois plus grand que D, en général, sans plus de précision.

Mais ceci est important dans la mesure où D est grand. L'écran comporte 454 lignes de 560 points et pour une carte utilisant au maximum l'écran, D serait de l'ordre de 500 (pour la France D = 480).

Quant au rapport entre Nlnv et $\sum Nlnv_i/Nlnv$ il est supérieur à 2, car toute arête appartient à deux régions exceptées les arêtes du pourtour des sous-cartes qui n'appartiennent qu'à une seule région. Comme pour D, plus le nombre de régions est important, plus le rapport est grand. Une ligne de hachurage coupera d'autant plus de régions que leur nombre est élevé. Et par conséquent, à une abscisse donnée, le nombre de régions possédant une arête ayant son origine sur cette ligne, risquera d'être élevé et donc le rapport $\sum Nlnv_i/Nlnv$ sera plus important.

Mais voyons ce qu'il en est à l'étape : Traiter ligne courante. Une arête appartient à la liste courante à partir de l'instant où la ligne de hachurage atteint son origine jusqu'à ce que cette même ligne de hachurage atteigne l'abscisse précédant son point final. Ceci fait que l'évaluation de Traiter liste courante est en partie indépendante du nombre d'appels.

Nous appelons **arête supérieure du pourtour d'une sous-carte**, une arête du pourtour d'une sous-carte, telle qu'en descendant une ligne de hachurage l'intersectant, on rencontre cette arête alors qu'on se trouve à l'extérieur de toutes les régions de la carte. Soit

Nat le nombre total d'arêtes non verticales de la carte
Naps le nombre d'arêtes supérieures des pourtours des sous-cartes
Narst le nombre d'arêtes restant (Nat = Naps + Narst)
L : la longueur moyenne d'une arête projetée sur l'axe des X.

L'évaluation suivante porte sur l'ensemble des appels de **Traiter liste courante** et non sur un seul passage.

```
Traiter liste courante - ensemble des régions
début
  Initialisations                          D
  tq lr ≠ 0 faire                          Nat*L + D)?
  début
    si lreg = − 1 alors                    (Nat*L)?
      Trait1                               Naps*L
    sinon
      Trait2                               Narst*L
    fsi
    Déterminer région à traverser          Nat*L
    si Xfin(lr) = 0 alors                  (Nat*L)(?,AT)
      Trait3                               Nat
    sinon
      Trait4                               Nat*(L-1)
    fsi
    lr ppv Suiv(lr)
Nat*L(ppv,AT)
  ftq
fin
```

Les initialisations se font à chaque appel, soit D fois. Il s'agit de deux affectations : 2D ppv

Le corps de la boucle Tant que est effectué pour toute arête autant de fois qu'elle appartient à la liste courante. Soit Nat*L réalisations du corps de la boucle pour l'ensemble du hachurage.

Trait1 est un appel d'une fonction d'affichage (faf). Cet appel est fait lorsque l'arête en cours de traitement de la liste courante est une arête supérieure. Soit Naps*M (faf,AT).

Trait2 est donc effectué dans les autres cas, il se décompose en un test et quelque soit le résultat, il y a un appel à une fonction d'affichage. Soit : Narst*L(faf,!)AT).

La détermination de la région est constituée d'un test et d'une affectation quelque soit le résultat du test. Avec deux accès tableaux. Soit : (Nat*1)(?,ppv,2AT).

Trait3 représente le rechaînage de liste courante après suppression d'une arête. Il y a un test et une affectation. Et suivant le résultat du test, il y a 1 ou 2 accès tableaux. Il y a un seul accès tableau lorsque l'arête à supprimer est la première arête de la liste courante. Ceci se présente donc, en général, beaucoup plus rarement que l'autre cas. De plus, le nombre de passage dans Trait3 est minime par rapport à Trait4 et au reste de la procédure. Quelle que soit la longueur de l'arête, elle ne «passe» qu'une seule fois dans Trait3 ; lors de sa suppression. Soit : Nat(?,ppv,2AT).

Trait4 ou mise à jour de l'ordonnée de l'arête courante se décompose en deux affectations et une addition avec trois accès tableaux. Soit : **Nat**(L-1) (2ppv, + ,3AT).

L'évaluation totale de Traiter liste courante dans l'hypothèse d'un hachurage par balayage de l'ensemble des régions se décompose de la manière suivante :

4Nat*L – Nat + 2D	ppv	
8Nat*L – Nat	AT	
4Nat*L + Nat + Narst*L + D	?	18Nat*L – 2Nat + Narst*L + 3D
Nat*(L – 1)	+	
Nat*L	faf	

Pour le hachurage région par région, ce n'est plus les arêtes supérieures qui sont intéressantes mais les arêtes des pourtours des sous-cartes car elles ne sont traitées qu'une fois, alors que les arêtes délimitant deux régions entre elles, sont traitées deux fois, lorsque le hachurage complet de la carte est réalisée. Soit

Nap : le nombre d'arêtes du pourtour des sous-cartes
Nar : le nombre d'arêtes restant (Nap + Nar = Nat).

Là aussi, l'évaluation suivante concerne l'ensemble du hachurage et non pas l'évaluation d'un seul passage ou pour une seule région.

Traiter liste courante - région par région

Début	
Initialisations	$\sum D_i$
tq lr ≠ 0 **faire**	$\sum D_i + (2Nar + Nap)*L/2$
début	
si Xfin(lr) = X_0 **alors**	(2Nar + Nap)*L
Trait3	2Nar + Nap
sinon	
Trait4	(2Nar + Nap)*(L – 1)
fsi	
lr ppv Suiv(lr)	(2Nar + Nap)*L
ftq affichage	(2Nar + Nap)*L
ftq	
fin	

Le traitement du corps de la boucle se fait par couples d'arêtes, le test du tant que n'est donc fait qu'une arête sur deux.

Les initialisations portent sur un test et un appel fonction d'affichage, puis deux affectations, soit : $\sum D_i$(?,faf,2ppv,AT).

Trait3 : 1 test, 1 affectation, 2 accès tableaux, soit : 2Nar + Nap(?,ppv,2AT).

Trait4 : 2 affectations, 1 addition et 3 accès tableaux, soit : 2Nar + Nap)*(L-1) (2ppv, 1 + , 3AT).

Soit au total, pour l'ensemble du hachurage région par région, dans **Traiter liste courante** :

$3[2Nar + Nap] + L - (2Nar + Nap) + 2\sum D_i$	ppv	
$5[(2Nar + Nap)] + L - (2Nar + Nap) + \sum D_i$	AT	
$3/2[2Nar + Nap)] + L + (2Nar + Nap) + 2\sum D_i$	?	$23/2[2Nar + Nap] + L$
$[2Nar + Nap]*(L - 1)$	+	$- 2(2Nar + Nap) + 6\sum D_i$
$\sum D_i + (2Nar + Nap)*L$	faf	

Sachant que Nat = Nap + Nar, on obtient :

$$\mathbf{23/2(Nat*L) + 23/2(Nar*L) - 2Nat - 2Nar + 6\sum D_i}$$

Comparaison des deux évaluations de Traiter liste courante

Ensemble des régions		Région par région
$4Nat*L - Nat + 2D$	ppv	$3Nat*L - Nat + 3Nar*L - Nar + 2\sum D_i$
$8Nat*L - Nat$	AT	$5Nat*L\text{-}Nat + 5Nar*L - Nar + \sum D_i$
$4Nat*L + Nat + Narst*L + D$	?	$3/2Nat*L + Nat + 3/2Nar*L + Nar + 2\sum D_i$
$Nat(L - 1)$	+	$Nat*(L - 1) + Nar*(L - 1)$
$Nat*L$	faf	$Nat*L + Nar*L + \sum D_i$
$18Nat*L - 2Nat + Narst*L + 3D$	**Total**	**$23/2Nat*L - 2Nat + 23/2Nar*L - 2Nar + 6\sum D_i$**

Si on élimine les coûts communs, on obtient le tableau suivant des différences d'évaluation.

$Nat*L + 2D$	ppv	$3Nar*L - Nar + 2\sum D_i$
$3Nat*L$	AT	$5Nar*L - Nar + \sum D_i$
$5/2Nat*L + Narst*L + D$	?	$3/2Nar*L + Nar + 2\sum D_i$
	+	$Nar*L - Nar$
	faf	$Nar*L + \sum D_i$
$13/2Nat*L + Narst*L + 3D$	**Total**	**$23/2Nar*L - 2Nar + 6\sum D_i$**

Or Nar = Nar + Nap = Narst + Naps. Si il n'y a aucun rapport entre Nap et Naps, il n'en est pas de même entre Nap*L et Naps*L. En effet, la longueur des arêtes supérieures est nécessairement égale à la longueur des arêtes inférieures et par conséquent : Nap*L = 2Naps*L, donc :

$$Narst*L = Nat*L - 1/2Nap*L = Nar*L + 1/2Nap*L$$

On obtient un nouveau tableau des différences :

Ensemble des régions		Région par région
$Nap*L + 2D$	ppv	$2Nar*L - Nar + 2\sum D_i$
$3Nap*L$	AT	$2Nar*L - Nar + \sum D_i$
$3Nap*L + 2Nar*L + D$	?	$Nar + 2\sum D_i$
	+	$Nar*L - Nar$
	faf	$Nar*L + \sum D_i$
$7Nap*L + 3D$	**Total**	**$4Nar*L - 2Nar + 6\sum D_i$**

Première constatation, le nombre d'appels de fonction d'affichage est plus élevé lors du hachurage par la méthode région par région que dans l'autre. Mais ces appels concernent les fonctions : déplacer la plume en tel point pour Nar*L occurrences et lever ou baisser la plume pour $\sum D_i$ occurrences. Ce sont des fonctions qui demandent très peu de temps comparé à l'autre fonction : Tracer jusqu'en tel point.

On peut également remarquer que, suivant la configuration de la carte telle ou telle méthode est meilleure. Le hachurage région par région sera meilleur, pour la partie **Traiter liste courante**, si les sous-cartes correspondent chacune à une seule région (Nar = 0) ou si la carte est découpée en lamelles verticales (Nar = 0 et $\sum D = D$). Par contre, dans le cas général, le nombre d'arêtes délimitant uniquement les régions deux à deux sera plus important que le nombre d'arêtes délimitant le pourtour des sous-cartes et $\sum D_i$ sera égale à plusieurs fois D. Ce qui fait que l'algorithme hachurage par balayage de l'ensemble des régions sera nettement meilleur que l'autre, toujours dans la partie **Traiter liste courante**. Et ce résultat est conforté par un meilleur comportement dans la partie de l'algorithme général. Ceci nous amène à parler de la dernière partie : fusion de liste courante et de liste X_o.

Fusion de deux listes n'est pas facile à évaluer et nous ne rentrerons pas dans les détails. Mais nous pouvons faire quelques remarques.

Nous avons vu que le nombre d'appels de **fusion** était plus important dans le hachurage région après région que dans l'autre cas. Mais ceci est très insuffisant pour conclure, puisque dans l'évaluation d'un algorithme de fusion intervient le nombre des éléments de chaque liste. Or, il est évident que dans le hachurage par balayage de l'ensemble des régions les listes courantes et X_o sont plus longues, d'où la difficulté à comparer les deux comportements.

Mais quels que soient ces comportements, ils influent peu sur le coût total du hachurage. En effet, le nombre d'appels de cette partie de l'algorithme de hachurage est peu élevé, nous avons vu qu'il était égal à Nlnv et $\sum Nlnv_i$, ce qui est relativement faible comparé aux autres éléments de l'évaluation de l'algorithme. Par conséquent, la différence de comportement sur cette partie **fusion** est minime par rapport aux différences obtenues pour l'algorithme général et l'algorithme de **Traiter liste courante**.

CONCLUSION

On peut donc affirmer sans risque d'erreur que l'algorithme de hachurage par balayage de l'ensemble des régions est plus efficace (plus rapide) que l'algorithme région après région et ceci d'une manière assez sensible. C'est un résultat qui confirme ce que nous pouvions penser et qui est d'ailleurs confirmé par l'examen des temps d'exécution sur l'ensemble de la France et ses régions.

Pour terminer, il me reste à parler de l'aspect interactif du logiciel ainsi réalisé. La principale difficulté liée à cet aspect est de trouver le juste milieu. En effet, il ne doit pas y avoir trop d'interruptions pour le dialogue entre le système et l'utilisateur sans quoi le travail de l'utilisateur devient vite irritant. Par contre, pour l'utilisateur il est intéressant qu'il puisse effectuer son travail avec le plus de liberté possible et par conséquent le nombre d'interruptions doit être relativement important pour qu'il décide lui-même d'effectuer telle ou telle partie de son travail, sans nécessairement suivre un ordre de travail trop rigide.

Si lors de la saisie d'une carte, il est difficile de lui donner toute liberté pour rentrer son fond de carte, lors du hachurage, il devra pouvoir réaliser ce qu'il veut à n'importe quel instant (modification du pas de hachurage, de l'échelle des paliers...).

Après avoir apporté une solution à ces considérations d'ordre général, il a fallu résoudre un problème technique. Une grande partie du dialogue entre le système et l'utilisateur s'effectue en mode graphique, car il est impensable de demander à l'utilisateur des directives concernant la carte sur laquelle il travaille sans qu'il ait celle-ci sous les yeux. Or, le HP présente l'inconvénient d'avoir un affichage de messages assez lentsen mode graphique. Par conséquent, il est difficile de lui imposer des messages trop longs, qu'il finira par connaître par cœur. Il faut donc lui imposer des messages aussi courts que possibles.

La méthode choisie consiste à afficher sur l'écran un numéro de message qui lui permet de se reporter au manuel d'utilisation et de lire le message correspondant. Bien sûr, cette méthode n'est pas très efficace pour un débutant, mais le nombre de messages est assez limité (autour de 25) et rapidement il les connaîtra sans avoir à se reporter à chaque fois au manuel.

Bibliographie

La cartographie en tant que telle n'a pas fait l'objet d'une grande littérature, elle est souvent traitée comme aspect de la graphique. C'est pourquoi la bibliographie reste succinte.

BERTIN Jacques
La graphique et le traitement graphique de l'information, *Flammarion 1977*

BERTIN Jacques
De la «CAO» à la «CAG» (conception assistée par la graphique), *Informatique et Sciences Humaines n°52, publication de l'Institut des Sciences Humaines Appliquées. PARIS 1982*

LEDRU Alain
Algorithmes de remplissage pour la synthèse d'images, *Manuscrit n°79. IMAG GRENOBLE 1979*

LEDUC Alain
Saisie de carte zonale, *Cartographie Informatisée et Géographie Humaine (Tome 3). Groupe Image ROUEN 1980*

LEDUC Alain
Le Système Cartovec, *Cahiers géographiques de ROUEN n°10-11 1979*

PINCHON Chantal
Cartographie Automatique des Données Quantitatives, *Cahiers géographiques de ROUEN n°10-11 1979*

PINCHON Chantal
Représentations cartographiques d'un caractère, *Cartographie Informatisée et Géographie Humaine (Tome 3). Groupe Image ROUEN 1980*

NEWMAN et SPROUL
Principles of Interactive Computer Graphics, *Mc Graw Hill, New-York 1979 (2è édition)*

Ouvrage collectif
La réalisation des logiciels graphiques interactifs, *Ecole d'Eté du Bréau Sans Nappe. Juillet 1979. Sous la direction de Michel LUCAS. Eyrolles, 1981.*

Liste des Publications Internes IRISA

PI 150 **Construction automatique et évaluation d'un graphe d'«implication» issu de données binaires, dans le cadre de la didactique des mathématiques**
H. Rostam , 112 pages ; Juin 1981

PI 151 **Réalisation d'un outil d'évaluation de mécanismes de détection de pannes] -]Projet Pilote SURF**
B. Decouty, G. Michel, C. Wagner, Y. Crouzet , 59 pages ; Juillet 1981

PI 152 **Règle maximale**
J. Pellaumail , 18 pages ; Septembre 1981

PI 153 **Corrélation partielle dans le cas « qualitatif »**
I.C. Lerman , 125 pages ; Octobre 1981

PI 154 **Stability analysis of adt ptively controlled not-necessarily minimum phase systems with disturbances**
Cl. Samson , 40 pages ; Octobre 1981

PI 155 **Analyses d'opinions d'instituteurs à l'égard de l'appropriation des nombres naturels par les élèves de cycle préparatoire**
R. Gras , 37 pages ; Octobre 1981

PI 156 **Récursion induction principle revisited**
G. Boudol, L. Kott , 49 pages ; Décembre 1981

PI 157 **Loi d'une variable aléatoire à valeur R^+ réalisant le minimum des moments d'ordre supérieur à deux lorsque les deux premiers sont fixés**
M.Kowalowka, R. Marie , 8 pages ; Décembre 1981

PI 158 **Réalisations stochastiques de signaux non stationnaires, et identification sur un seul échantillon**
A. Benveniste J.J. Fuchs , 33 pages ; Mars 1982

PI 159 **Méthode d'interprétation d'une classification hiérarchique d'attributs-modalites pour l'«explication» d'une variable ; application à la recherche de seuil critique de la tension artérielle systolique et des indicateurs de risque cardiovasculaire**
B. Tallur , 34 pages ; Janvier 1982

PI 160 **Probabilite stationnaire d'un réseau de files d'attente multiclasse à serveur central et à routages dépendant de l'état**
L.M. Le Ny , 18 pages ; Janvier 1982

PI 161 **Détection séqentielle de changements brusques des caractéristiques spectrales d'un signal numérique**
M. Basseville, A. Benveniste , pages ; Mars 1982

PI 162 **Actes regroupés des journées de Classification de Toulouse (Mai 1980), et de Nancy (Juin 1981)**
I.C. Lerman , 304 pages ;

PI 163 **Modélisation et Identification des caractéristiques d'une structure vibratoire : un problème de réalisation stochastique d'un grand système non stationnaire**
M. Prévosto, A. Benveniste, B. Barnouin , 46 pages ; Mars 182

PI 164 **An enlarged definition and complete axiomatization of observational congruence of finite processes**
Ph. Darondeau , 45 pages ; Avril 1982

PI 165 **Accès vidéotex à une banque de données médicales**
A. Chauffaut, M. Dragone, R. Rivoire, J.M. Roger , 25 pages ; Mai 1982

PI 166 **Comparaison de groupes de variables définies sur le même ensemble d'individus**
B. Escofier, J. Pages , 115 pages ; Mai 1982

PI 167 **Transport en circuits virtuels internes sur réseau local et connexion Transpac**
M. Tournois, R. Trépos , 90 pages ; Mai 1982

PI 168 **Impact de l'intégration sur le traitement automatique de la parole**
P. Quinton , 14 pages ; Mai 1982

PI 169 **A systolic algorithm for connected word recognition**
J.P. Banâtre, P. Frison, P. Quinton , 13 pages ; Mai 1982

PI 170 **A network for the detection of words in continuous speech**
J.P. Banâtre, P. Frison, P. Quinton , 24 pages ; Mai 1982

PI 171 **Le langage ADA : Etude bibliographique**
J. André, Y. Jégou, M. Raynal , 12 pages ; Juin 1982

PI 172 **Comparaison de groupes de variables : 2ème partie : un exemple d'application**
B. Escofier, J. Pajès , 37 pages ; Juillet 1982

PI 173 **Unfold-fold program transformations**
L. Kott , 29pages ; Juillet 1982

PI 174 **Remarques sur les langages de parenthèses**
J.M. Autebert, J. Beauquier, L. Boasson, G. Senizergues , 20 pages ; Juillet 1982

PI 175 **Langages de parenthèses, langages N.T.S. et homomorphismes inverses**
J.M. Autebert, L. Boasson, G. Senizergues , 26 pages ; Juillet 1982

PI 176 **Tris pour machines synchrones ou Baudet Stevenson revisited**
R. Rannou , 26 pages ; Juillet 1982

PI 177 **Un nouvel algorithme de classification hiérarchique des éléments constitutifs de tableau de contingence basé sur la corrélation**
B. Tallur , Juillet 1982 ;

PI 178 **Programmes d'analyse des re'sultats d'une classification automatique**
I.C. Lerman et collaborateurs , 79 pages ; Septembre 1982

PI 179 **Attitude à l'égard des mathématiques des élèves de sixième**
J.Degouys, R. Gras, M. Postic , 29 pages ; Septembre 1982

PI 180 **Traitements de textes et manipulations de documents : bibliographie analytique**
J. André , 20 pages ; Septembre 1982

PI 181 **Algorithme assurant l'insertion dynamique d'un processeur autour d'un réseau à diffusion et garantissant la cohérence d'un système de numérotation des paquets global et réparti**
Annick Le Coz, Hervé Le Goff, Michel Ollivier , 31 pages ; Octobre 1982

PI 182 **Interprétation non linéaire d'un coefficient d'association entre modalités d'une juxtaposition de tables de contingence**
Israël César Lerman , 34 pages ; Novembre 1982

PI 183 **L'IRISA vu à travers les stages effectués par ses étudiants de DEA (1è année de thèse)**
Daniel Herman , 41 pages ; Novembre 1982

PI 184 **Commande non linéaire robuste des robots manipulateurs**
Claude Samson , 52 pages ; Janvier 1983

PI 185 **Dialogue et représentation des informations dans un système de messagerie intelligent**
Philippe Besnard, René Quiniou, Patrice Quinton, Patrick Saint-Dizier, Jacques Siroux, Laurent Trilling , 45 pages ; Janvier 1983

PI 186 **Analyse classificatoire d'une correspondance multiple ; typologie et régression**
I.C. Lerman , 54 pages ; Janvier 1983

PI 187 **Estimation de mouvement dans une sequence d'images de télévision en vue d'un codage avec compensation de mouvement**
Claude Labit , 132 pages ; Janvier 1983

PI 188 **Conception et réalisation d'un logiciel de saisie et restitution de cartes élémentaires**
Eric Sécher , 45 pages ; Janvier 1983

PI 189 **Etude comparative d'algorithmes pour l'amélioration de dessins au trait sur surfaces point par point**
M.A. ROY , 96 pages ; Janvier 1983

PI 190 **Généralisation de l'analyse des correspondances à la comparaison de tableaux de fréquence**
Brigitte Escofier , 35 pages ; Mars 1983

PI 191 **Association entre variables qualitatives ordinales «nettes» ou «floues»**
Israël-César Lerman , 42 pages ; Mars 1983

PI 192 **Un processeur intégré pour la reconnaissance de la parole**
Patrice Frison , 80 pages ; Mars 1983

PI 193

PI 194 **Régime stationnaire pour une file M/H/1 avec impatience**
Raymond Marie et Jean Pellaumail , 8 pages ; Mars 1983

PI 195 **SIGNAL : un langage pour le traitement du signal**
Paul Le Guernic, Albert Benveniste, Thierry Gautier , 49 pages ; Mars 1983

PI 196 **Algorithmes systoliques : de la théorie à la pratique**
Françoise André, Patrice Frison, Patrice Quinton , 19 pages ; Mars 1983

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique