Détection et correction des intersections entre courbes B-splines. Application a la généralisation cartographique.

Eric Guilbert

N° d'ordre : 3012

Thèse

présentée

devant l'Université de Rennes 1

pour obtenir

le grade de : **Docteur de l'Université de Rennes 1**

Mention *INFORMATIQUE*

par

Eric Guilbert

Équipe d'accueil :	*Institut de Recherche de l'Ecole Navale (IRENav, EA 3634)*
École doctorale :	*Mathématiques, Télécommunications, Informatique, Signal, Système et Electronique (MATISSE)*
Composante universitaire :	*IFSIC*

Détection et correction des intersections entre courbes B-splines Application à la généralisation cartographique

soutenue le 8 novembre 2004 devant la commission d'examen :

Composition du Jury

Président :	Christophe Claramunt, Professeur, Ecole Navale, Brest
Rapporteurs :	Christopher Gold, Professeur, University of Glamorgan Stefanie Hahmann, Maître de Conférences, INP Grenoble
Examinateurs :	Bruno Arnaldi, Professeur, INSA de Rennes Kadi Bouatouch, Professeur, Université de Rennes 1 Marc Daniel, Professeur, Université d'Aix-Marseille 2 Eric Saux, Maître de Conférences, Ecole Navale, Brest

Remerciements

Je tiens tout d'abord à remercier Christopher Gold (University of Glamorgan) et Stefanie Hahmann (INP Grenoble) pour l'intérêt qu'ils ont porté à mon travail en acceptant d'être les rapporteurs de cette thèse. Je remercie également Kadi Bouatouch et Christophe Claramunt pour avoir accepté de faire partie de ce jury de thèse.

Je souhaite également remercier Bruno Arnaldi (INSA Rennes) et Marc Daniel (Université d'Aix-Marseille) pour avoir accepter respectivement la direction et la co-direction de cette thèse et pour leur disponibilité. Les discussions que j'ai eues avec eux et les conseils qu'ils m'ont donnés furent très précieux pour améliorer la qualité de ce mémoire.

Je remercie Christophe Claramunt et l'ensemble du groupe Systèmes d'Information Géographique de l'IRENav pour m'avoir accueilli parmi eux. Leur motivation et leur enthousiasme ont contribué au bon déroulement de cette thèse. Merci également à l'ensemble des membres de l'IRENav, notamment le secrétariat et les services administratifs et informatiques pour leur aide. Je remercie aussi Michel Faramin et David Hélary de l'EPSHOM qui m'ont fourni les données nécessaires pour mes travaux. Ils ont de plus toujours été disponibles pour répondre à mes nombreuses questions sur la généralisation des cartes.

Je tiens surtout à remercier Eric Saux qui a encadré mon travail. Il m'a fait partager sa rigueur et ses compétences. Qu'il trouve ici l'expression de ma profonde reconnaissance.

Enfin, je remercie mes parents et mes frères ainsi qu'Antoine, Bertrand, Fabien, Franck, Ingrid, Jean-Christophe, Laure, Louis, Marc, Myriem, Nadège, Rodéric, Sébastien×3, Valérie, Yann et Youenn pour leur amitié et leur soutien.

Résumé

Cette thèse présente une méthode de détection et de correction des intersections entre courbes B-splines adaptée à la généralisation des lignes de profondeur (isobathes) pour les cartes marines. L'intérêt est de traiter les intersections visuelles (lorsque les courbes sont trop proches) et singulières (tangence, superposition de deux courbes) dans un grand ensemble de données. Ce mémoire est divisé en deux parties.

Dans une première partie, nous nous intéressons à la détection des intersections. La méthode proposée effectue d'abord un partitionnement du plan à l'aide d'un quadtree. La répartition des courbes dans les cellules se fait en comparant les positions des points de contrôle. Cette méthode est fiable et rapide puisque aucun point n'est calculé sur les courbes lors de la répartition, évitant ainsi les erreurs numériques et limitant les calculs. Ensuite, dans chaque cellule, les segments de courbe sont approchés par des lignes polygonales à l'aide de schémas de subdivision. Nous considérons que deux courbes sont en intersection si la distance entre les segments est inférieure à la distance de lisibilité. Les intersections sont repérées par les indices des points de contrôle des courbes en conflit. Des exemples sur des cartes marines sont présentés. Les résultats sont discutés en fonction de la profondeur du quadtree et de la précision des approximations.

La deuxième partie concerne la correction des conflits par déformation des courbes. Ces corrections doivent se faire en respectant les contraintes cartographiques. Les contraintes de sécurité et de lisibilité nous imposent un déplacement minimal pour assurer la validité de la correction. Le respect de la contrainte géomorphologique concernant la conservation des formes est évalué par des critères sur la norme du déplacement effectué et sur sa courbure. Nous présentons une première méthode de correction combinant des approches géométrique et mécanique. Une première solution est obtenue en calculant géométriquement un déplacement pour chaque point de contrôle puis le polygone de contrôle est assimilé à un réseau de barres et la courbe est déformée en modifiant les forces agissant aux points du réseau afin d'améliorer la solution précédente. La deuxième méthode est fondée sur les contours actifs (snakes). Le déplacement à effectuer est représenté par un contour actif soumis à des énergies internes tendant à conserver la forme de la courbe et des énergies externes fonction des conflits à corriger. Une solution est obtenue lorsque l'énergie du contour est minimale. Les paramètres de forme réglant le rapport entre les énergies sont propres à chaque courbe. Dans le but d'automati-

ser le processus de correction des conflits, ces paramètres sont réglés automatiquement. Les deux méthodes de correction sont ensuite comparées sur les conflits détectés dans la première partie. Enfin, la méthode des contours actifs est étendue pour corriger les auto-intersections.

Abstract

In this thesis, a method for detecting and correcting intersections between B-spline curves is introduced. The method is adapted to line generalisation for maritime charts (especially isobathymetric lines) and can deal with visual (i.e. proximity conflicts) and singular intersections (i.e. tangency and overlapping) in a large set of data. This report is divided into two parts.

First, we focus on intersection location. The map is segmented in a quadtree and the curves are shared in the different cells. This is done only by comparing the position of the control points. The curves are shared without inserting any point in order to avoid numerical errors and to reduce the computation time so that the method is fast and reliable. Possible intersections are then computed in each cell by approximating the curve segments with polygonal lines using subdivision schemes. Two curves intersect if the distance between the segments is less than the legibility distance. Intersections are defined with the indices of the control points of the conflicting segments. Some examples issued from maritime charts are given. Results are discussed for different quadtree depths and different values of approximation precision.

Secondly, conflicts are corrected by deforming the curves with respect to cartographic constraints. A minimal displacement is imposed by the legibility and security constraints. As a third constraint, the relief of the map has to be maintained. This constraint is evaluated by measuring the effective displacement and its curvature. A first correction method is presented, combining both geometrical and mechanical approaches. A first solution is obtained geometrically by computing a displacement for each control point, then, the control polygon is considered as a cable network and the curve is deformed by modifying forces applied on the points of the network in order to improve the previous solution. The second method is based on snakes (active contours). The curve displacement is considered as a snake submitted to internal energies preserving the shape of the curve and external energies according to the conflicts. A solution is reached when the energy is minimal. In order to automate conflict correction, parameters are fixed automatically. Both methods are compared on the conflicts detected in the first part of this thesis. The snake method is also extended to self-intersection correction.

Table des matières

Table des figures

Liste des tableaux

Préambule

Dans ce chapitre, nous rappelons les définitions et les propriétés essentielles des courbes B-splines. Nous nous restreignons ici aux courbes B-splines planes. Les notations introduites dans ce chapitre seront reprises dans la suite du mémoire. Pour plus d'informations, nous renvoyons le lecteur à [Farin, 1992].

Définitions

Une courbe B-spline f est une fonction paramétrique définie d'un intervalle $I = [a, b] \subset \mathbb{R}$ dans le plan $\mathbb{R}^2$. Elle est définie par

$$f(t) = \sum_{i=0}^{m} Q_i N_i^k(t) \tag{1}$$

Les valeurs Q_i sont des points de $\mathbb{R}^2$ appelés points de contrôle ou pôles. Les N_i^k sont les fonctions de base B-splines. Ce sont des fonctions polynomiales par morceaux définies de I dans $\mathbb{R}$. Elles sont de degré $k-1$. k est appelé l'ordre de la courbe B-spline. Les fonctions de base sont à support compact. Pour les définir, nous avons besoin d'une suite T de valeurs réelles ($t_0 = a \leq t_1 \leq \ldots \leq t_i \leq \ldots \leq t_{m+k} = b$) appelée le vecteur de nœuds ou la séquence nodale.

Les fonctions de base B-splines sont définies récursivement en fonction de k. Avec la convention $\frac{0}{0} = 0$, nous posons

$$\left\{ \begin{array}{ll} N_i^1(t) = & \left\{ \begin{array}{ll} 1 & \text{si } t_i \leq t \leq t_{i+1} \text{ et } t_i < t_{i+1} \\ 0 & \text{sinon} \end{array} \right. \\ N_i^j(t) = & \frac{t-t_i}{t_{i+j-1}-t_i} N_i^{j-1}(t) + \frac{t_{i+j}-t}{t_{i+j}-t_{i+1}} N_{i+1}^{j-1}(t) \text{ pour } 2 \leq j \leq k \end{array} \right. \tag{2}$$

La définition prise pour $N_i^1(t)$ permet d'avoir $N_i^k(b) = 1$ au lieu de $N_i^k(b) = 0$ dans le cas classique [Daniel, 1989].

Les fonctions de base B-splines N_i^k vérifient les propriétés suivantes :
- elles forment une partition de l'unité : $\sum_{i=0}^{m} N_i^k \equiv 1$;
- elles sont positives : $\forall t, \forall i, N_i^k(t) \geq 0$;
- elles sont à support local : $N_i^k(t) = 0 \quad \forall t \notin [t_i, t_{i+k}]$;

– elles sont de classe C^{k-2} sauf aux points correspondant à des nœuds confondus ($t_i = t_{i+1} = \ldots = t_{i+r-1}$) où la classe est C^{k-r-1}. r est l'ordre de multiplicité du nœud.

Propriétés

Les propriétés suivantes découlent directement de la définition des fonctions de base.

Propriété 1 *Une courbe B-spline est polynomiale par morceaux de degré $k-1$.*

Propriété 2 *Il y a au plus k fonctions de base N_j^k, $i-k+1 \leq j \leq i$ non nulles sur un intervalle $[t_i, t_{i+1}]$.*

Propriété 3 *Un point $f(t)$ est défini comme barycentre des points de contrôle Q_j, $i-k+1 \leq j \leq i$ pondérés par les coefficients $N_j^k(t)$ avec $t \in [t_i, ti+1]$.*

La dernière propriété signifie que tout point de la courbe peut s'écrire comme combinaison affine de k points de contrôle. Nous avons vu également que les fonctions N_i^k sont toujours positives et au plus k fonctions sont non nulles en un point. Par conséquent, tout point de la courbe s'écrit comme une combinaison convexe de k points de contrôle. Nous en déduisons deux propriétés importantes pour les courbes B-splines.

Propriété 4 *Le déplacement d'un point de contrôle entraine une modification locale de la courbe. Si un point de contrôle Q_i est déplacé, la courbe est modifiée sur l'intervalle $[t_i, t_{i+k}]$.*

Propriété 5 *Un arc de courbe défini sur un intervalle $[t_i, t_{i+1}]$ est toujours contenu dans l'enveloppe convexe des points de contrôle Q_{i-k+1}, ..., Q_i. A partir des enveloppes convexes locales de chaque intervalle, une enveloppe contenant la courbe B-spline peut être définie (figure 1).*

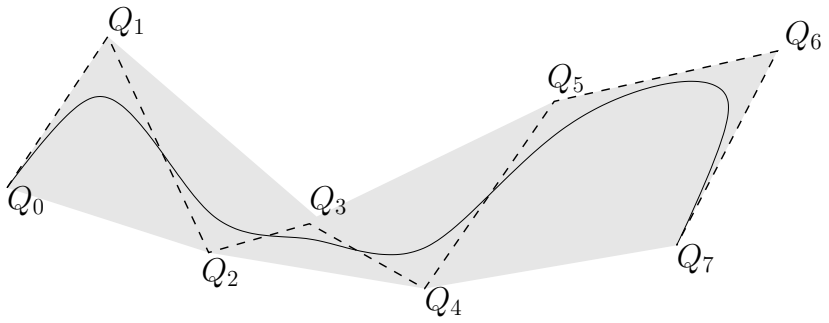

Fig. *1* — Polygone de contrôle (tirets) et enveloppe convexe d'une courbe B-spline (en gris).

Si les premier et dernier nœuds du vecteur T sont de multiplicité k alors la courbe B-spline passe par les points de contrôle extrêmes. Remarquons que ceci ne change en rien les propriétés de continuité et de partition de l'unité des fonctions de base.

Construction d'une courbe B-spline

Lors de la construction d'une courbe B-spline, l'utilisateur a plusieurs paramètres à définir. En général, la courbe est construite à partir d'un ensemble de points donnés $(P_i)_{i=0}^n$ par approximation (la courbe passe au plus près des points) ou par interpolation (la courbe passe par les points). En premier lieu, l'ordre de la spline est choisi suivant deux critères :

- la stabilité numérique : lorsque l'ordre augmente, les calculs sont plus importants et il y a plus de problèmes numériques ;
- le contrôle des dérivées : une courbe d'ordre k est de classe C^{k-2} par morceaux. Un ordre plus élevé permet une meilleure représentation des formes.

En pratique, les courbes cubiques ($k = 4$) sont les plus utilisées car elles permettent un contrôle de la courbure.

Nous donnons ici une méthode de construction calculant le vecteur de nœuds et les coordonnées des points de contrôle. Cette méthode n'est pas unique. Elle consiste à approcher ou à interpoler les points P_i par des points de la courbe pour des paramètres ζ_i choisis par l'utilisateur. La paramétrisation est généralement prise en fonction de la longueur et des angles entre les segments joignant les points [Lee, 1989].

Le vecteur de nœuds peut être uniforme, c'est-à-dire $t_i = i$ pour $i = 0$ à $m + k - 2$ ou adapté aux données représentées. Le traitement des courbes B-splines uniformes est plus rapide puisqu'il n'est pas nécessaire de stocker le vecteur de nœuds et la répartition uniforme permet de simplifier les calculs qui sont alors indépendants de l'intervalle paramétrique. Par contre, il peut être mal adapté lorsque la courbe est construite à partir de points dont la répartition est irrégulière.

Une fois que l'ordre et le vecteur de nœuds ont été choisis, nous avons défini les fonctions de base B-splines. Il reste maintenant à calculer les points de contrôle. Pour cela, nous cherchons à résoudre l'équation

$$\min \sum_{j=0}^{n} \|f(\zeta_j) - P_j\|^2 = \min \sum_{j=0}^{n} \| \sum_{i=0}^{m} Q_i N_i^k(\zeta_j) - P_j\|^2 \tag{3}$$

où $\|.\|$ est la norme euclidienne et les inconnues sont les coordonnées des points de contrôle. Si $m = n$, nous avons autant d'équations que d'inconnues et f interpolera les points P_j. Si $m < n$, (3) est résolu par moindres carrés et f approchera les points P_j.

Dérivée d'une courbe B-spline

Les fonctions dérivées par rapport à t d'une B-spline f sont obtenues à partir des dérivées des fonctions de base. En utilisant la propriété de support local des fonctions de base, la dérivée $p^{\text{ème}}$ au point t est calculée par

$$\frac{d^p f(t)}{dt^p} = \sum_{j=i-(k-p)+1}^{i} Q_j^p N_j^{k-p}(t) \tag{4}$$

où les $Q_j^p = (k-p)\frac{Q_j^{p-1} - Q_{j-1}^{p-1}}{t_{j+k-p} - t_j}$ sont calculés par récurrence. D'après cette formule, la dérivée d'ordre p d'une courbe B-spline d'ordre k est une courbe B-spline d'ordre $k-p$.

Définition 1 *La courbe associée à la dérivée première par rapport à t est appelée l'hodographe de f et est notée f'.*

Insertion de nœuds

Le processus d'insertion d'un nœud consiste à insérer un nœud dans le vecteur T sans modifier la courbe [Boehm, 1980]. Le nœud peut être nouveau ou correspondre à un nœud déjà existant de multiplicité $r < k$. Quand un nœud est inséré, nous remplaçons $k-1$ points de contrôle par k nouveaux points de contrôle.

Si un nœud $\bar{t}$ est inséré avec $t_i \leq \bar{t} < t_{i+1}$, l'algorithme d'insertion calculant les nouveaux points de contrôle est le suivant :

Résultat 1 *Algorithme de Boehm. Pour $j = i-k+2$ à i*

$$\bar{Q}_j = \frac{t_{j+k-1} - \bar{t}}{t_{j+k-1} - tj-1} Q_{j-1} + \frac{\bar{t} - t_{j-1}}{t_{j+k-1} - tj-1} Q_j \tag{5}$$

Lorsqu'un nœud doit être inséré plusieurs fois, le résultat 1 peut être appliqué jusqu'à atteindre la multiplicité voulue. Le même résultat peut être obtenu plus rapidement en appliquant la formule suivante :

Résultat 2 *Soit $\bar{t} \in [t_i, t_{i+1}[$ et r la multiplicité de $\bar{t}$, pour $j = 1, \ldots, k-1-r$ et $l = i-k+j+2, \ldots, i+1$, nous définissons*

$$Q_l^j = \frac{t_{l+k-1-j} - \bar{t}}{t_{l+k-1-j} - t_{l-1}} Q_{l-1} j-1(\bar{t}) + \frac{\bar{t} - t_{l-1}}{t_{l+k-1-j} - t_{l-1}} Q_l^{j-1}(\bar{t}) \tag{6}$$

alors, $f(u) = Q_{i+1}^{k-r-1}$.

Un autre algorithme d'insertion de nœuds est l'algorithme d'Oslo [Cohen et al., 1980] mais il est moins performant que l'algorithme de Boehm [Daniel, 1989].

En pratique, l'insertion de nœuds est souvent utilisée pour calculer une approximation de la courbe à partir du polygone de contrôle : en insérant de nouveaux nœuds, nous ajoutons de nouveaux points de contrôle et le polygone de contrôle est de plus en plus proche de la courbe. Elle est aussi utilisée pour transformer une courbe B-spline en courbe de Bézier. Si tous les nœuds existants sont amenés à une multiplicité égale à $k-1$ alors chaque segment défini sur un intervalle $[t_i, t_{i+1}]$ avec $t_i < t_{i+1}$ tous deux de multiplicité $k-1$ est une courbe de Bézier. L'ensemble de la courbe est définie par des courbes de Bézier raccordées à leurs extrémités.

Introduction

La carte marine est l'outil utilisé en navigation maritime pour se localiser et déterminer sa route. Elle fournit une représentation simplifiée du relief. Elle peut également contenir d'autres informations touristiques ou sur la nature des fonds. Elle est donc utilisée également pour la plaisance ou la pêche, par exemple. En France, la construction et la diffusion des cartes marines relèvent de la responsabilité juridique du Service Hydrographique et Océanographique de la Marine (SHOM) et doit respecter les standards définis par l'Organisation Hydrographique Internationale (OHI). La construction se fait en plusieurs étapes. D'abord, des mesures sont effectuées par des sondeurs lors de campagnes en mer. Ensuite, toutes les données relevées sont nettoyées (les données aberrantes ou redondantes sont supprimées) et assemblées dans la Base de Données Bathymétriques du SHOM (BDBS).

Les cartes marines sont construites à partir des données extraites de la BDBS. Les principaux objets constituant une carte sont les sondes (points de profondeur) et les isobathes (lignes de niveau). L'information contenue dans la BDBS est trop importante pour être restituée. Elle est donc simplifiée et schématisée pour être adaptée à l'échelle de la carte. L'étape de sélection et de schématisation s'appelle la généralisation. Deux sortes de généralisation sont effectuées : la généralisation objet qui consiste à sélectionner l'information suivant des critères sémantiques et la généralisation cartographique qui consiste à simplifier ou supprimer des objets suivant des contraintes géométriques ou spatiales. Elle est obtenue en appliquant des opérateurs de suppression, de déformation d'agrégation, etc.

Les contraintes cartographiques sont établies suivant les recommandations de l'OHI. Les principales contraintes sont la contrainte de sécurité (un objet ne doit jamais être à une profondeur inférieure à sa profondeur réelle), la contrainte de lisibilité (les objets doivent être suffisamment distincts pour que l'information soit lisible à l'échelle de la carte) et la contrainte géomorphologique (le relief des fonds doit être préservé). Les deux premières contraintes sont des contraintes fortes et doivent être impérativement respectées, la troisième contrainte est une contrainte faible. Les données contenues dans la BDBS ne pouvant pas être modifiées, la généralisation se fait sur les données extraites et non pas sur la BDBS.

Depuis quelques années, le métier de cartographe est en profonde évolution avec l'arrivée de nouvelles technologies. Notamment, les relevés bathymétriques sont maintenant effectués avec des sondeurs multi-faisceaux et la quantité de données est devenue trop importante

pour être traitée manuellement. Les cartographes doivent donc faire appel à des méthodes informatiques de plus en plus performantes. Des méthodes de construction automatiques, ou au moins semi-automatiques sont envisagées. De plus, la carte marine qui était traditionnellement reproduite sur un support en papier, est de plus en plus souvent traduite sous forme électronique. Il en résulte la généralisation des systèmes d'aide à la navigation comme l'ECDIS (*Electronic Chart Display and Information System*). La carte électronique doit alors être interactive et permettre des opérations de visualisation telles que l'agrandissement et le panning. De ce fait, il est nécessaire de développer des outils adaptés à la quantité de données et aux différents supports.

En cartographie routière et urbaine, des opérateurs de généralisation automatique ont été développés pour le traitement des routes et des bâtiments représentés par des listes de points. Ces opérateurs sont adaptés aux contraintes de généralisation spécifiques aux cartes routières et urbaines. En cartographie marine, des algorithmes de généralisation sont utilisés pour réduire le volume de données à traiter. Notamment, des algorithmes de sélection de sondes sont utilisés.

L'ensemble des sondes de la BDBS forme un modèle numérique de terrain. Les isobathes sont construites par interpolation des points du modèle ayant la même profondeur. Elles sont au départ définies par des listes de points de même profondeur. Cette représentation ne restitue pas le caractère lisse des isobathes. Les déformations sont appliquées en déplaçant chaque point. Pour conserver la forme de la courbe, il faut donc que les déplacements soient homogènes. Cette représentation est également mal adaptée à certains opérateurs de visualisation comme l'agrandissement qui accentue l'effet d'arcs brisés. Récemment, des travaux ont été réalisés pour modéliser les isobathes par des courbes B-splines. Les courbes B-splines sont alors utilisées pour approcher ces listes de points à l'aide de techniques de compression et de lissage. Lors de la construction de ces courbes, il peut apparaître des conflits spatiaux liés à des problèmes d'interpolation ou d'approximation des points. Les conflits sont également dus à des problèmes de lisibilité (courbes trop proches). Leur existence étant liée à l'échelle de la carte et la BDBS ne pouvant pas être modifiée, nous ne pouvons pas revenir aux données source pour les supprimer. Nous devons donc appliquer les opérateurs de généralisation aux courbes B-splines de la carte afin de corriger ces conflits.

Le but de cette thèse est de développer une méthode de détection et de correction des intersections entre courbes B-splines en tenant compte des contraintes spécifiques à la cartographie marine. L'objectif est de développer une méthode robuste (traitant tous les types de conflit) et rapide pouvant être utilisée pour la généralisation semi-automatique voire automatique des cartes marines. Les algorithmes développés sont testés sur des cartes marines fournies par l'Etablissement Principal du SHOM (EPSHOM)[1]. Ce mémoire est donc organisé en deux parties.

Dans la première partie, nous nous intéressons au problème de la détection des intersections entre isobathes représentées par des courbes B-splines. En cartographie, nous

[1]EPSHOM, 13 rue Chatellier, BP 30316, 29603 BREST CEDEX

considérons qu'il y a intersection entre deux courbes lorsque sur la carte, la distance entre les courbes est trop petite pour pouvoir les distinguer à l'œil ou lorsqu'il y a une intersection franche (c'est-à-dire, deux courbes qui se croisent en un point). Les courbes isobathes étant des courbes de niveaux, nous n'aurons que très peu d'intersections franches, ces dernières étant essentiellement dues à des problèmes de modélisation. La méthode mise en place doit donc être robuste pour traiter les problèmes de proximité (intersections visuelles) et les cas de tangence ou de recouvrement (intersections réelles) qui sont numériquement difficiles à caractériser. La méthode doit également être rapide étant donné qu'elle est appliquée à de grands volumes de données.

Dans le chapitre 1, nous nous intéressons aux méthodes de détection des intersections entre courbes B-splines. Nous distinguons les intersections réelles et les intersections visuelles. Ensuite, nous passons en revue les différentes méthodes de détection existantes (algébriques, non linéaires, géométriques). Nous nous attardons plus particulièrement sur les méthodes géométriques et comparons ces méthodes pour la détection de conflits entre isobathes.

Dans le chapitre 2, nous présentons une méthode de détection adaptée au processus de généralisation des cartes marines. Nous expliquons d'abord ce qu'est la généralisation et précisons quelles sont les contraintes spécifiques aux cartes marines (sécurité, lisibilité, conservation de la géomorphologie). La détection se fait en deux étapes. Dans un premier temps, la carte est segmentée par un quadtree et les courbes sont réparties dans chaque cellule. L'intérêt est de réduire les temps de calcul en effectuant les tests d'intersection uniquement pour les segments de courbe situés dans une même cellule. La segmentation des courbes se fait en étudiant le polygone de contrôle sans insérer de nouveaux points afin d'éviter les erreurs numériques. L'appartenance d'un segment à une cellule est définie à une tolérance près. Cela permet de traiter tous les types d'intersection et assure la robustesse de la méthode. Dans un deuxième temps, nous détectons les intersections à l'intérieur de chaque cellule. Pour cela, nous approchons les segments de courbe par des courbes polygonales à l'aide de techniques de subdivision puis calculons les intersections entre ces lignes. La méthode est appliquée à des cartes réelles. Les résultats sont discutés en fonction de la profondeur du quadtree et de la précision des approximations.

La deuxième partie de la thèse concerne la correction des intersections entre isobathes. En généralisation, un conflit peut être corrigé de plusieurs façons (suppression, déformation, agrégation). Dans le cadre de la correction automatique des conflits, le cartographe choisit l'opérateur de correction en fonction des objets et des contraintes. Ensuite, ces opérateurs sont appliqués automatiquement sans l'intervention de l'utilisateur. Pour notre problème, nous ne disposons que des isobathes et ne considérons pas les autres objets présents sur la carte tels que les sondes ou les traits de côtes. Nous nous intéressons à la correction des conflits entre deux isobathes par déplacement des courbes.

Dans le chapitre 3, nous appliquons différentes méthodes de déformation de courbes pour corriger les conflits détectés dans le chapitre précédent afin de satisfaire les contraintes cartographiques. Les contraintes sont exprimées sous forme géométrique. La contrainte de sécurité

impose que seule la courbe la plus profonde peut être déplacée. La contrainte de lisibilité nous donne le déplacement minimal à effectuer défini par une ligne polygonale. La contrainte géomorphologique est utilisée pour mesurer la qualité des résultats et est exprimée par des critères de forme.

Nous présentons deux méthodes de déformation. La première est une méthode géométrique où les courbes sont déformées en calculant un déplacement pour chaque point de contrôle. Nous calculons un premier déplacement à partir des distances entre les courbes en conflit puis proposons une méthode fondée sur une approche de réseaux de barres pour améliorer cette solution. Il s'agit de considérer les points de contrôle comme les sommets du réseau et de déformer la courbe en modifiant les forces aux sommets. Pour la deuxième méthode, nous utilisons des contours actifs. Il s'agit d'un modèle énergétique où la contrainte de lisibilité est exprimée sous la forme d'une énergie externe tendant à déformer la courbe alors que la contrainte géomorphologique est exprimée par une énergie interne limitant ces déformations. La correction s'effectue en minimisant l'énergie totale du système. Afin d'appliquer cette méthode à tous les types d'intersection énumérés dans le chapitre 1, nous cherchons à déterminer automatiquement les paramètres de forme du contour actif. Les méthodes mises en place sont appliquées à la correction des conflits détectés dans le chapitre 2. Nous comparons et discutons les résultats à l'aide des critères de forme que nous avons définis.

Enfin, nous concluons et donnons quelques perspectives pour améliorer et étendre les méthodes présentées à d'autres domaines ayant recours à des méthodes géométriques comme les systèmes d'information géographiques ou la CFAO.

CHAPITRE 1

Détection des intersections entre courbes paramétriques

1.1 Introduction

Dans ce premier chapitre, nous présentons une synthèse des différentes méthodes de détection des intersections entre courbes paramétriques dans le plan. Ces méthodes sont réparties en plusieurs grandes familles. Le choix d'une méthode de détection dépend du mode d'expression des courbes (courbes exprimées dans la base canonique, courbes de Bézier, courbes B-splines) et de plusieurs critères qui sont notamment :

- Le type d'intersection à rechercher ;
- La précision voulue sur le résultat ;
- La rapidité de la méthode.

Aucune méthode ne satisfait tous les critères à la fois. Le choix doit donc se faire en fonction de l'importance de chaque critère pour un problème donné. Nous présentons différentes méthodes et donnons une comparaison des résultats obtenus pour la détection des intersections entre courbes en fonction des critères précédents.

Dans le paragraphe 1.2, nous nous attachons à définir l'intersection de deux courbes et l'auto-intersection et introduisons la notion d'intersection visuelle par opposition aux intersections dites de modélisation.

Le paragraphe 1.3 est consacré à la présentation des différentes méthodes de détection réparties en plusieurs familles. Il y a d'abord les méthodes algébriques et non linéaires fondées sur la résolution d'un système d'équations puis les méthodes géométriques. Ces dernières fonctionnent en construisant des volumes englobant les courbes puis en réduisant les courbes en fonction des positions de ces volumes. Les méthodes géométriques varient en fonction du choix des volumes et des méthodes de réduction.

Dans le paragraphe 1.4, nous appliquons les méthodes de détection à des courbes B-splines. Dans la littérature, les résultats sont souvent présentés pour des courbes de Bézier avec des intersections franches. Nous donnons les temps obtenus pour calculer les intersections entre des

courbes B-splines composées d'un grand nombre de points pour des intersections singulières en fonction des tolérances utilisées. Les données sur lesquelles nous appliquons nos méthodes sont extraites de jeux de données bathymétriques fournis par le Service Hydrographique et Océanographique de la Marine (SHOM).

1.2 Définitions

1.2.1 Intersection réelle

L'intersection de deux courbes est l'ensemble des points appartenant aux deux courbes à la fois. Dans le cas des courbes paramétriques, les points sont définis par leur valeur paramétrique sur chacune des deux courbes. L'intersection est donc composée d'un ensemble de couples de valeurs paramétriques associées aux mêmes points pour chacune des courbes. Il existe plusieurs types d'intersection (figure 1.1). Pour le cas (d), nous avons une infinité de points d'intersection et l'intersection est définie par deux intervalles paramétriques regroupant tous ces points. Les intersections (a) et (b) sont des intersections franches facilement détectables alors que les cas (c) et (d) sont plus difficiles à caractériser, soulevant des problèmes numériques. Nous parlerons d'intersections *régulières* et *singulières* pour différencier les deux premiers cas des deux derniers.

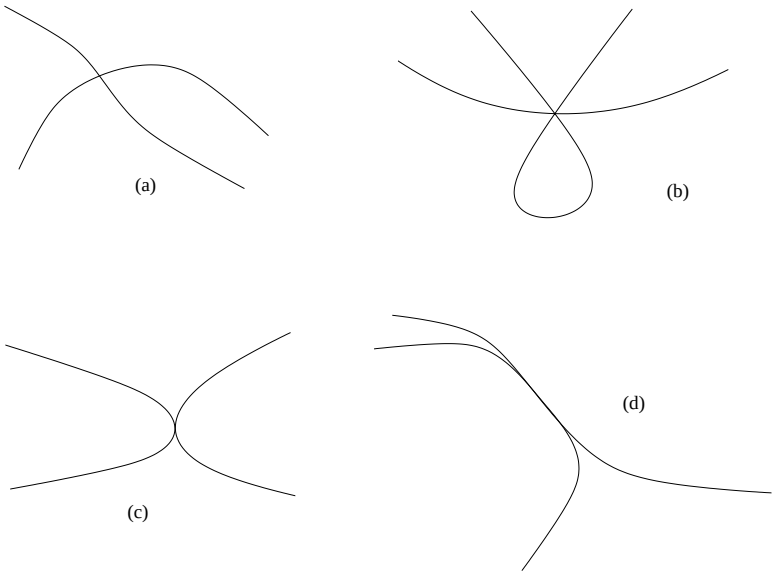

Fig. *1.1* — Les différents types d'intersection : (a) intersection ordinaire, (b) intersection multiple, (c) intersection tangentielle, (d) recouvrement ou superposition.

Mathématiquement, l'intersection de deux courbes paramétriques est définie comme suit :

Définition 2 *Soient f et g deux courbes paramétriques définies de $I \subset \mathbb{R}$ dans $\mathbb{R}^2$ et de*

$I' \subset \mathbb{R}$ *dans* $\mathbb{R}^2$. *L'intersection de* f *et de* g *est définie comme étant*

$$\{(t, t') \in I \times I', f(t) = g(t')\} \tag{1.1}$$

Une intersection pourra être simple (figure 2a) ou multiple (les autres cas). La multiplicité de l'intersection de deux courbes est définie géométriquement de la manière suivante [Manocha et Demmel, 1995] :

Définition 3 *Un point d'intersection* p *de deux courbes est de multiplicité* l *si une légère perturbation d'une ou des courbes donne* l *points d'intersection distincts dans le voisinage de* p.

Suivant cette définition, les intersections tangentielles sont considérées comme des intersections doubles. Les recouvrements peuvent être considérés comme une infinité d'intersections tangentielles.

1.2.2 Auto-intersection

L'auto-intersection d'une courbe est la réunion de l'ensemble des intersections de tous les couples possibles de segments de courbe disjoints.

Définition 4 *Soit* f *une courbe paramétrique définie de* $I \subset \mathbb{R}$ *dans* $\mathbb{R}^2$. *L'auto-intersection de* f *est définie comme étant*

$$\{(t, t') \in I \times I, \exists \eta > 0, |t - t'| > \eta, f(t) = f(t')\} \tag{1.2}$$

Remarquons que les deux variables t et t' jouent un rôle symétrique. Il n'est donc pas nécessaire de considérer leur ordre dans le couple (t, t'). Les différents types d'auto-intersection sont les mêmes que pour les intersections de deux courbes (figure 1.2). Elles peuvent par exemple être simples (a), tangentielles (b) ou comporter une infinité de points (c). Enfin, les points de rebroussement (*cusps*) (d) sont des cas limites d'auto-intersections. Ils correspondent au cas singulier $f'(t) = 0$. Ces singularités sont détectées avec les mêmes méthodes que pour les auto-intersections régulières.

1.2.3 Intersection visuelle

1.2.3.1 Précision des résultats

En calcul numérique, les résultats sont toujours donnés avec une précision finie. Au mieux, il s'agit de la précision de la machine. En général, cette précision est beaucoup plus large et dépend de la précision des valeurs en entrée du problème, des erreurs de calculs inévitables avec la représentation machine en virgule flottante et de la précision voulue sur les valeurs en sortie.

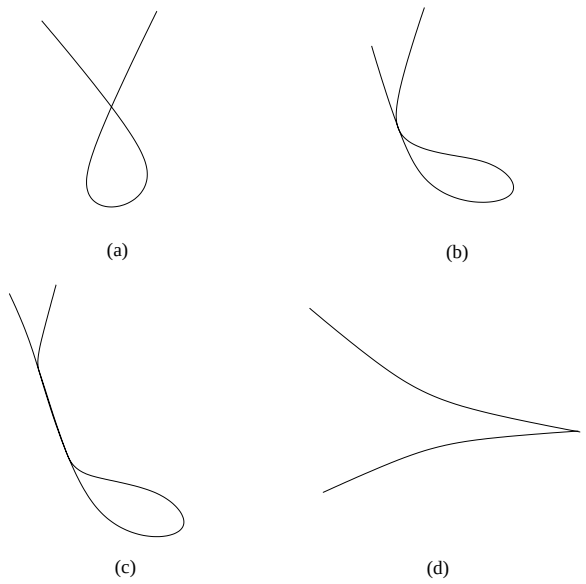

Fig. *1.2* — Les différents types d'auto-intersection : (a) simple, (b) tangentielle, (c) recouvrement, (d) point de rebroussement.

Par exemple, si les données du problème sont obtenues par des mesures expérimentales, elles comportent toujours une erreur due à la mesure elle-même. Cette erreur doit être prise en compte car elle peut être significative sur les résultats obtenus. Pour cela, il existe des méthodes permettant de prendre en compte ces erreurs et de les reporter au cours d'un processus afin de connaître l'erreur sur le résultat final. Pour la détection des intersections entre courbes, les principales méthodes sont l'arithmétique des intervalles [Hu et al., 1996] et l'arithmétique affine [Figueiredo, 1996] donnant un encadrement de la solution. Ces méthodes sont surtout adaptées pour les cas où l'erreur sur la valeur de départ est relativement petite puisque, au fur et à mesure que les calculs sont effectués, l'intervalle d'erreur se dilate et peut donner une solution inexploitable s'il devient trop grand.

Un autre problème est l'introduction d'erreurs numériques liées à la représentation des nombres en machine et aux erreurs d'arrondi. Pour remédier à cela, nous pouvons utiliser l'arithmétique exacte [Fortune et Van Wyk, 1993]. Il s'agit de conserver en mémoire toutes les opérations dans un arbre et d'effectuer les calculs uniquement lorsque cela est nécessaire pour éviter les erreurs de troncature. Cette méthode est extrêmement coûteuse aussi bien en temps qu'en place mémoire et n'a un intérêt que si l'on a besoin d'une très grande précision sur les résultats. En général, il n'est pas nécessaire d'avoir recours à des calculs aussi précis. Une solution est donc l'arithmétique paresseuse [Michelucci et Moreau, 1995]. Il s'agit de combiner l'arithmétique des intervalles et l'arithmétique exacte : les nombres sont représentés à la fois par un intervalle et une définition permettant de calculer la valeur exacte. Le nombre est évalué exactement lorsque l'intervalle ne permet pas de prendre une décision (par exemple, s'il faut évaluer le signe d'un nombre contenu dans un intervalle contenant 0).

Les méthodes citées ci-dessus sont utilisées pour gérer les erreurs antérieures ou les erreurs

de calcul de la méthode. Un autre problème est la précision du résultat final. Dans beaucoup de cas, l'objectif de l'utilisateur n'est pas de connaître exactement la solution mais simplement si elle existe. Dans le cas de la détection des intersections, il s'agit alors de définir une méthode robuste permettant de conclure à l'existence de cette intersection. La méthode la plus couramment employée est l'utilisation de tolérances [Daniel, 1989]. Dans ce cas, une valeur est considérée comme solution lorsqu'elle est à une distance inférieure à une tolérance fixée ϵ de la solution exacte. Cette méthode permet de traiter les cas singuliers comme la tangence de deux courbes qui sont numériquement difficiles à caractériser. Ainsi, nous disons que deux courbes se coupent à ϵ près si la distance entre les courbes est inférieure à ϵ. Le principal intérêt est que cela permet de traiter tous les types d'intersection sans distinction. Nous ne travaillons plus avec des points d'intersection ou de tangence mais avec des zones de contact.

Ce concept a été étendu avec le principe de la géométrie floue [Foufou, 1997] où une logique trivaluée est utilisée. Chaque entité est définie avec un flou topologique. Par exemple, les points sont des boules de rayon ϵ et les lignes disposent d'une épaisseur. Un test d'intersection peut avoir trois réponses : non intersection, intersection potentielle, intersection sûre.

1.2.3.2 Définition de l'intersection à une tolérance près

En pratique, nous ne calculons pas des intersections exactes. Les résultats sont donnés avec une certaine tolérance pour tenir compte des problèmes numériques liés à la représentation des nombres en machines et à la précision des données en entrée et en sortie. La détection des intersections doit se faire en fonction de ces précisions. Les définitions 2 et 4 nous donnent des intersections exactes et ne sont pas compatibles avec la notion de tolérance. Elles ne peuvent pas être utilisées telles quelles. En reprenant la notion de tolérance du paragraphe 1.2.3.1, nous donnons une définition de l'intersection définie à une tolérance près.

Définition 5 *Soient f et g deux courbes paramétriques définies de $I \subset \mathbb{R}$ dans $\mathbb{R}^2$ et de $I' \subset \mathbb{R}$ dans $\mathbb{R}^2$. L'intersection de f et de g à une tolérance ϵ près est définie comme étant*

$$\{(t,t') \in I \times I', \|f(t) - g(t')\| \leq \epsilon\} \tag{1.3}$$

Suivant cette définition, dès qu'un point de f et un point de g sont à une distance inférieure à ϵ, ils sont considérés en intersection. Cette définition peut également être étendue aux auto-intersections [Dubois, 2000]. Dans ce cas, il faut s'assurer que la courbe est bien revenue sur ses pas pour qu'il y ait un conflit. L'auto-intersection est alors l'ensemble des intersections à ϵ près des couples de segments disjoints séparés par un segment dont la tangente décrit un secteur angulaire supérieur à π.

Définition 6 *Soit f une courbe paramétrique définie de $I \subset \mathbb{R}$ dans $\mathbb{R}^2$. L'auto-intersection de f à ϵ près est définie comme étant*

$$\begin{aligned} \{(t,t') \in I \times I, \quad & \exists \eta > 0, |t - t'| > \eta, \\ & \forall u \in \mathbb{R}^2, \exists v \in f'([t,t']), u.v < 0, \|f(t) - f(t')\| \leq \epsilon\} \end{aligned} \tag{1.4}$$

L'intérêt de ces définitions est leur robustesse. Elles permettent de détecter tous les types d'intersection de la même façon et donnent toujours une réponse sur l'existence de l'intersection. Le seul problème est la valeur de la précision. Il faut choisir un ϵ adéquat au problème traité. S'il est trop grand, les résultats ne correspondront pas aux intérêts de l'utilisateur. S'il est trop petit, il peut persister des problèmes numériques et les critères d'intersection peuvent ne pas être satisfaits.

1.2.3.3 Intersection de visualisation et intersection de modélisation

Dans la conception d'un système de CAO, deux types d'intersection sont distinguées : les "intersections de modélisation" et les "intersections de visualisation". Dans le premier cas, il s'agit de définir l'intersection à la précision du modèle en coordonnées cartésiennes (position de l'intersection dans l'espace) et en coordonnées paramétriques. La tolérance notée ϵ_{mod} est donc faible afin d'avoir une modélisation mathématique du résultat. Ceci est nécessaire par exemple pour la modélisation d'un objet en CFAO.

Le deuxième type d'intersection est lié aux problèmes de rendu et de visualisation d'une scène sur un support (écran ou carte papier par exemple). La précision de l'intersection est donc liée à la précision du support et est notée ϵ_{vis}. Dans le cadre de la généralisation cartographique, un des objectifs est d'obtenir une carte lisible. Nous disons donc qu'il y a intersection visuelle entre deux segments de courbe si un critère de lisibilité n'est pas satisfait. Ce critère est lié à l'échelle de la carte. Les intersections sont donc bien définies à une tolérance près mais cette tolérance est beaucoup plus grande que la précision requise pour les intersections de modélisation. Nous appelons cette tolérance la distance de lisibilité. Elle dépend de deux choses : l'épaisseur du trait de crayon utilisé pour le tracé et la distance nécessaire pour que l'observateur puisse percevoir deux courbes distinctes sur la carte [Ruas et Bianchin, 2002]. Sur les figures 1.3 et 1.4, nous avons deux exemples d'auto-intersection visuelle. Sur la première figure à gauche, il n'y a pas d'auto-intersection mais à droite, le trait est plus épais et il y a auto-intersection visuelle. Sur la figure 1.4, le trait est le même que sur la courbe de gauche mais il y a auto-intersection parce que l'échelle est plus petite. Sur les deux figures, il y a un conflit qui doit être corrigé pour que l'information soit lisible.

1.3 Les méthodes de détection

Les différentes méthodes de détection des intersections entre courbes paramétriques peuvent être réparties en trois catégories. Nous présentons dans un premier temps les méthodes algébriques et les méthodes non linéaires fondées sur des méthodes numériques. Ensuite, nous expliquons le principe des méthodes géométriques. Ces méthodes sont itératives. A chaque étape, nous testons s'il n'y a pas intersection et nous segmentons les courbes. Nous nous intéressons à deux points : les volumes englobants utilisés pour tester les intersections et

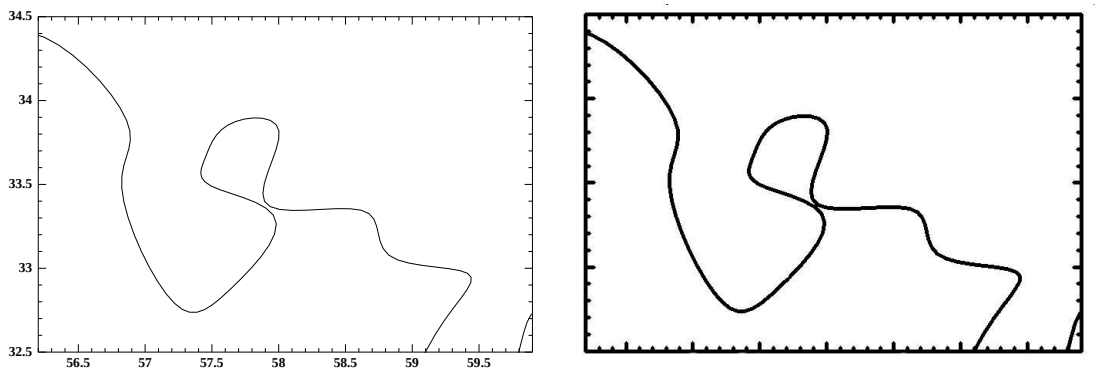

Fig. *1.3* — Auto-intersection visuelle due à l'épaisseur du trait.

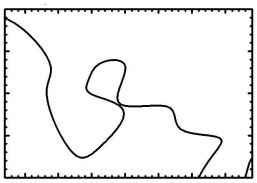

Fig. *1.4* — Auto-intersection visuelle due au changement d'échelle.

les méthodes de segmentation des courbes. Enfin, nous abordons le cas des auto-intersections.

1.3.1 Les méthodes algébriques et non linéaires

Ces méthodes transforment le problème d'intersection en la résolution d'un système d'équations. Ces méthodes peuvent être directes ou itératives. Dans la littérature, elles sont présentées pour des courbes de Bézier. Lorsque les courbes sont représentées sous forme B-spline, il faut les transformer en insérant des nœuds dans le vecteur nodal afin qu'ils soient tous de multiplicité k égale à l'ordre de la spline (résultat 1).

1.3.1.1 Les méthodes algébriques

La mise en équation du problème d'intersection impose pour ces méthodes [Sederberg et Parry, 1986, Manocha et Demmel, 1994] la connaissance d'une entité sous sa forme implicite (c'est-à-dire, sous la forme d'une équation $f(x, y) = 0$). Les points d'intersection entre deux courbes f et g où f est une fonction implicite de $\mathbb{R}^2$ dans $\mathbb{R}$ et g est une fonction paramétrique de $I \subset \mathbb{R}$ dans $\mathbb{R}^2$ sont les solutions de $f(g(t)) = 0$.

Les démarches élaborées sont issues de la géométrie algébrique et font appel à la notion de résultant. Le résultant est un polynôme à deux inconnues calculé à partir de deux polynômes f_1 et f_2 à valeurs réelles. Il est défini de manière à s'annuler lorsque les deux polynômes ont une racine commune. Sa définition n'est pas unique. Nous présentons ici celle utilisée dans

[Manocha et Demmel, 1994].

Définition 7 *Soient deux polynômes f_1 et f_2 définis de $\mathbb{R}$ dans $\mathbb{R}$, le résultant R de f_1 et f_2 est défini par :*

$$R(t,s) = \frac{f_1(t)f_2(s) - f_1(s)f_2(t)}{t-s} \tag{1.5}$$

Le résultant est utilisé pour calculer la forme implicite d'une courbe de Bézier. Soit $f(t)$ une courbe de Bézier plane de degré d. Ses composantes sont notées $f_x(t)$ et $f_y(t)$. Nous considérons le système suivant :

$$\begin{cases} f_1(t): & x - f_x(t) = 0 \\ f_2(t): & y - f_y(t) = 0 \end{cases} \tag{1.6}$$

où nous avons introduit les variables réelles x et y. Le résultant de f_1 et f_2 peut s'écrire sous la forme d'un système matriciel

$$R(t,s) = \begin{pmatrix} 1 & s & \cdots & s^{d-1} \end{pmatrix} (xM_1 + yM_2 + M_3) \begin{pmatrix} 1 & t & \cdots & t^{d-1} \end{pmatrix}^t \tag{1.7}$$

où M_1, M_2 et M_3 sont des matrices constantes de taille $d \times d$. En posant $M = xM_1 + yM_2 + M_3$, nous avons le résultat suivant :

Résultat 3 *La représentation implicite de f est donnée par le déterminant de M.*

En effet, le déterminant de M est un polynôme. Si ce déterminant est nul pour un couple (x_0, y_0), la matrice M est singulière et il existe un t_0 non nul tel que le résultant s'annule. Ce t_0 est le paramètre donnant $f(t_0) = (x_0, y_0)$ solution de (1.8).

$$M \begin{pmatrix} 1 & t & \cdots & t^{d-1} \end{pmatrix}^t = 0 \tag{1.8}$$

Soient deux courbes de Bézier $f(t)$ et $g(u)$ avec f de degré d et g de degré inférieur. f admet une forme implicite $\det M(x,y) = 0$. Nous cherchons les intersections entre ces deux courbes. Un point p du plan est intersection de f et g s'il appartient à la fois à f et à g, c'est-à-dire, s'il existe un u_0 tel que $p = g(u_0)$ et $\det M(p) = 0$. Les points d'intersection de f et g sont donc les solutions de $\det M(g(u)) = 0 \quad$ avec $u \in I$. La différence entre les méthodes algébriques de Manocha et Sederberg citées précédemment est la méthode utilisée pour résoudre cette équation. Sederberg calcule le déterminant alors que Manocha ramène son problème à un calcul de valeurs propres.

1.3.1.2 Les méthodes non linéaires

Soient deux courbes paramétriques f et g. Les méthodes non linéaires [Léon, 1991] consistent à résoudre le système d'équations suivant :

$$f(u) - g(v) = 0, \quad u \in I \subset \mathbb{R}, v \in I' \subset \mathbb{R}. \tag{1.9}$$

Il est possible de résoudre ce système à l'aide de méthodes de type Newton ou des méthodes de descente. Les techniques de ce type ont d'ailleurs été les premières à avoir été utilisées en vue de résoudre des problèmes d'intersection. Le problème de ces méthodes est qu'elles demandent la connaissance d'un point initial proche de la solution or on ne sait généralement pas où elle se situe ni même si elle existe.

Une autre approche pour résoudre (1.9) est la méthode présentée dans [Maekawa et Patrikalakis, 1993]. Cette méthode, présentée pour les courbes de Bézier, consiste à chercher les solutions des équations $f_x(u) - g_x(v) = 0$ et $f_y(u) - g_y(v) = 0$ en projetant chaque équation suivant l'axe x et l'axe y et en recherchant les racines de chaque projeté. Ces racines sont les intersections avec l'axe t. Les solutions du système sont les solutions communes à toutes ces équations. L'intérêt de cet algorithme est que l'on reste toujours dans la base de Bernstein et qu'il n'est pas nécessaire de savoir s'il existe une ou plusieurs solutions.

Sur la figure 1.5, nous illustrons le principe de recherche de racine d'un polynôme de Bézier à une inconnue. La courbe est définie par trois points de contrôle Q_0, Q_1, Q_2 et on cherche les zéros de la fonction. Pour cela, nous recherchons les intersections entre la courbe et l'axe t. Nous construisons l'enveloppe convexe du polygone de contrôle. Nous calculons d'abord les intersections entre l'enveloppe et l'axe t. Il s'agit simplement de calculer l'intersection entre les segments de l'enveloppe et l'axe t. Sur la figure 1.5, nous avons deux intersections aux paramètres t_i et t_j. S'il n'y a pas d'intersection alors la courbe n'a pas de racine. Sinon, la courbe est coupée aux points $(t, f(t))$ correspondants et les segments ne coupant pas l'axe sont éliminés. Une courbe plus petite est obtenue. Une nouvelle enveloppe est construite (en gris sur la figure) et la méthode est relancée jusqu'à avoir une solution suffisamment précise. Dans le cas où il y a plusieurs racines, la courbe est divisée en deux et l'algorithme est relancé sur chaque partie.

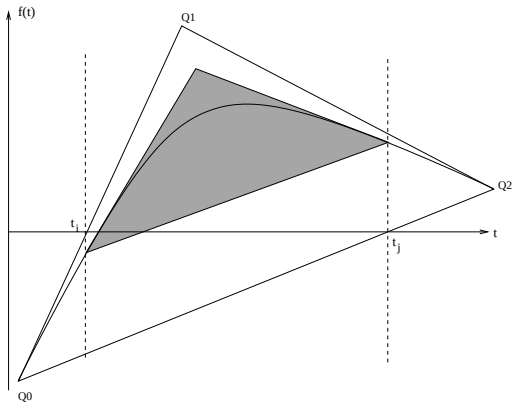

Fig. *1.5* — Calcul du zéro d'une courbe de Bézier quadratique : les segments à gauche de $t - i$ et à droite de t_j sont éliminés. La partie contenue dans l'enveloppe grise est conservée pour l'itération suivante.

1.3.1.3 Limites de ces méthodes

Le principal avantage des méthodes algébriques (pour des ordres inférieurs à cinq) et des méthodes non linéaires (hormis la méthode de projection) est leur rapidité. En effet, étant donné qu'elles font appel à des méthodes numériques, elles sont extrêmement rapides, particulièrement lorsque le degré des courbes est faible.

Les méthodes de descente et de Newton peuvent être utilisées pour calculer une intersection si celle-ci a déjà été localisée mais en général, ces méthodes ne sont pas adaptées pour les problèmes de détection. En effet, même en partant d'un point proche de la solution, les méthodes peuvent ne pas converger. Or, le plus souvent, nous ne savons pas si une intersection existe ni même si elle est unique. De plus, ces méthodes sont fondées sur la résolution numérique d'un système d'équations. Des problèmes numérique peuvent donc apparaître, rendant la méthode peu robuste. La méthode de Maekawa et Patrikalakis est plus robuste car elle est fondée des critères géométriques mais elle est plus lente puisqu'il faut calculer l'enveloppe convexe du polygone de contrôle et les intersections avec les axes à chaque itération, ce qui pose des problèmes de précision et rend la méthode peu adaptée pour traiter des intersections singulières.

Les méthodes algébriques sont également sujettes à ce problème de robustesse puisqu'il s'agit aussi de résoudre des systèmes matriciels. De plus, les méthodes de résolution utilisées passent par un calcul de déterminant ou de valeurs propres. Il peut donc y avoir des difficultés numériques si le déterminant est proche de zéro ou si les valeurs propres sont très proches.

Un problème commun à toutes ces méthodes est qu'elles ne peuvent pas traiter les auto-intersections. En effet, si l'on applique les équations (1.5) et (1.9) à deux fonctions f et g identiques, alors, tous les points de la courbe sont solutions.

Enfin, ces méthodes sont faites pour calculer des solutions ponctuelles. Elles ne sont pas adaptées pour calculer des solutions à une précision donnée. En particulier, elles ne peuvent pas être utilisées pour calculer des intersections singulières (figure 1.1 (c) et (d)) ou des intersections visuelles (figures 1.3 et 1.4).

1.3.2 Les méthodes géométriques

1.3.2.1 Principe général

Les méthodes géométriques sont fondées sur le principe "diviser pour régner". Pour chaque courbe, il s'agit de construire une enveloppe englobant la courbe. Si les deux enveloppes ne se coupent pas alors les courbes ne peuvent pas se couper. Sinon, les courbes sont divisées et l'algorithme est relancé pour les segments restants jusqu'à ce que l'on puisse conclure. Ces méthodes sont particulièrement adaptées aux courbes à pôles puisqu'elles utilisent principalement la propriété d'enveloppe convexe pour construire les volumes englobants. Nous traitons les problèmes d'intersection dans le plan mais les méthodes présentées peuvent être étendues

à l'espace.

Les différentes méthodes géométriques varient suivant deux critères : le type de volume englobant les courbes et la façon dont sont coupées les courbes à chaque itération. L'efficacité de la méthode est liée à ces deux critères puisqu'un volume grossier demandera à faire plus d'itérations alors qu'un volume beaucoup plus fin demandera plus d'opérations lors de la construction et des tests d'intersection. De même, les méthodes de segmentation peuvent être plus ou moins fines pour rechercher le point de séparation. En fonction du problème d'intersection, un compromis doit être fait sur le choix des englobants et des méthodes de réduction. Les méthodes les plus grossières sont sensées être plus lentes mais plus robustes que des méthodes plus fines où des problèmes numériques peuvent apparaître.

Les méthodes géométriques sont des méthodes récursives. A chaque étape, les segments de courbes sont de plus en plus petits. Nous nous arrêtons lorsque toutes les intersections entre les englobants sont vides (dans ce cas, il n'y a pas d'intersection entre les courbes) ou lorsqu'un critère d'arrêt est atteint, signifiant qu'il y a intersection. Ce critère peut être de plusieurs sortes :

- un certain niveau de récursivité est atteint [Aziz et al., 1990] ;
- l'intervalle paramétrique sur lequel est défini le segment de courbe est de longueur inférieure à une longueur fixée ;
- les englobants sont de taille inférieure à une taille fixée [Dubois, 2000] ;
- les segments de courbe sont assimilables à des segments de droite [Daniel, 1989, Lasser, 1989].

Seuls les deux derniers critères sont purement géométriques. L'inconvénient des deux autres critères est que leur vérification ne garantit pas la proximité des points.

Pour calculer une intersection, nous avons besoin de deux précisions. D'abord, il y a la précision définissant le critère d'arrêt. Elle définit par exemple la taille maximale des englobants. Cette précision doit être relativement petite et correspond à la précision numérique souhaitée. Elle est notée ϵ_{num} et est de l'ordre d'ϵ_{mod}. La deuxième précision est utilisée pour tester l'appartenance d'un point à un englobant. Sa grandeur dépend du type d'intersection recherché. Si nous cherchons des intersections de modélisation, elle sera du même ordre qu'ϵ_{mod}. Si nous recherchons des intersections visuelles, il y a intersection lorsque la distance de lisibilité n'est pas respectée et la précision sera prise égale à cette distance qui est notée ϵ_{vis}. Pour des intersections visuelles, nous avons $\epsilon_{vis} \gg \epsilon_{num}$.

Les méthodes géométriques fournissent une solution sous forme d'intervalle : nous obtenons à la fin un intervalle paramétrique pour chaque courbe dans lequel se situe l'intersection. A partir de cette solution, il est possible de calculer une valeur approchée du point d'intersection. Pour le troisième critère, le point d'intersection est contenu dans l'intersection des deux englobants. Pour le dernier critère, nous considérons qu'une courbe est assimilable à un segment de droite si tous les points de contrôle sont proches de ce segment. Le point d'intersection peut être estimé comme l'intersection des segments de droite. Si un segment de droite est défini par deux points $P_0 = f(t_0)$ et $P_1 = f(t_1)$, le point d'intersection P_2 peut

s'écrire $P_2 = \lambda P_0 + (1 - \lambda)P_1$. Le paramètre t du point d'intersection peut être obtenu en considérant une paramétrisation linéaire du segment : $t = \lambda t_0 + (1 - \lambda)t_1$.

1.3.2.2 Les volumes englobants

Plusieurs types de volumes englobant la courbe peuvent être considérés. Si les englobants sont trop grossiers, ils ont plus de risques de se couper. Il faudra donc plus d'itérations pour localiser précisément cette intersection. Si l'englobant est trop complexe à calculer, les itérations seront coûteuses. Dans le cas des courbes de Bézier ou des B-splines, un englobant suffisamment fin est l'enveloppe convexe du polygone de contrôle. Cependant, la détermination de celle-ci est en $O(n \log n)$ et l'étude de l'intersection de deux enveloppes est une étape longue et délicate [Preparata et Shamos, 1985]. Pour cela, nous préférons utiliser des englobants plus simples, construits à partir de cette enveloppe convexe.

L'englobant le plus simple à construire est la boîte minmax ou AABB (*Axis Aligned Bounding Box*). Il s'agit de construire une boîte alignée avec les axes contenant l'enveloppe convexe de la courbe. Pour cela, nous prenons comme coin inférieur gauche le point composé avec la plus petite abscisse et la plus petite ordonnée et comme coin supérieur droit le point composé de la plus grande abscisse et de la plus grande ordonnée (figure 1.6). Ces boîtes sont grossières mais leur coût de construction est très faible ($4n$ comparaisons où n est le nombre de points de contrôle). Le principal problème est que lorsque la courbe est orientée suivant une diagonale aux axes, la boîte minmax est très large.

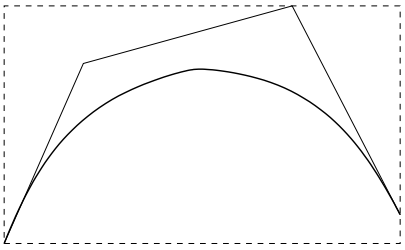

Fig. *1.6* — Courbe B-spline avec son polygone de contrôle et sa boîte minmax.

Un autre type de boîte est la boîte inclinée ou OBB (*Oriented Bounding Box*). Ces boîtes sont plus petites que les boîtes minmax, notamment quand les polygones sont inclinés (figure 1.7) mais leur coût de calcul est plus important ($O(n^3)$ pour une boîte inclinée optimale) [Gottschalk et al., 1996].

Enfin, Sederberg utilise dans [Sederberg et Nishita, 1990] une bande (*fat line*) contenant la courbe. Il s'agit de deux lignes parallèles contenant le polygone de contrôle (figure 1.8). Comme pour les boîtes alignées, l'épaisseur de la ligne dépendra de l'orientation choisie. Les auteurs privilégient la direction donnée par le segment reliant les points extrêmes. En effet, lorsque l'on avance dans le découpage des courbes, les segments sont de plus en plus plats

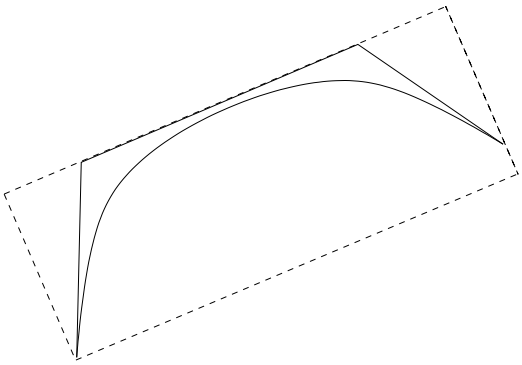

Fig. *1.7* — Boite inclinée englobant une courbe B-spline.

donc de plus en plus alignés avec le segment joignant les extrémités. Ce choix est donc plus intéressant que de chercher une direction optimale qui est une opération coûteuse. Le calcul de la bande est une opération assez simple à réaliser dans le plan : il suffit de calculer par projection orthogonale la distance signée entre chaque point de contrôle et la droite donnant l'orientation de la bande. La complexité, en $O(n)$, est inférieure à celle des boîtes inclinées mais supérieure aux boîtes minmax. La bande peut être calculée pour des problèmes dans le plan ou dans l'espace. Il s'agit dans ce cas de définir un tube contenant la courbe. Une méthode spécifique est présentée dans [Hlúšek, 2000].

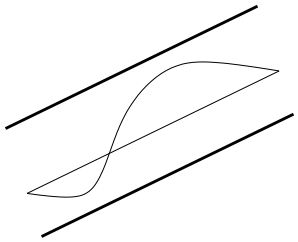

Fig. *1.8* — Bande contenant une courbe B-spline.

D'autres englobants tels des cercles ou des ellipses peuvent être utilisés. Sur le même principe que les bandes, Sederberg a défini les arcs épais (*fat arcs*) dans [Sederberg et al., 1989]. Il s'agit de construire deux cercles concentriques et de considérer l'espace inclus entre les deux cercles. Ensuite, un secteur de sommet confondu avec le centre des cercles est construit. L'arc épais est l'intersection du secteur et de l'espace entre les deux cercles (figure 1.9). Pour déterminer l'enveloppe, les paramètres suivants sont nécessaires :

- c le centre des cercles,
- r_{min} et r_{max} les rayons,
- α l'angle de l'arc.

L'équation d'un cercle de centre c et de rayon r est donnée par $(x - c_x)^2 + (y - c_y)^2 - r^2 = 0$. Le centre du cercle peut être calculé en construisant un système de moindres carrés à partir des points de contrôle. Cette méthode est coûteuse puisque la résolution est en

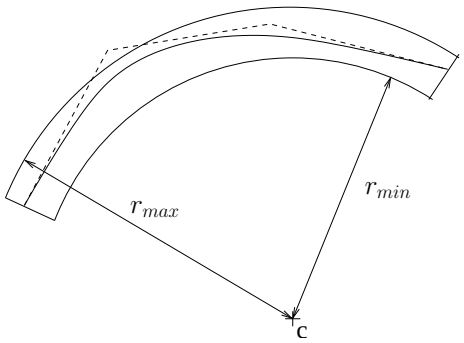

Fig. *1.9* — Arc épais d'une courbe B-spline.

$O(n^2)$ [Krishnan et al., 1998]. Une autre méthode est de choisir trois points de la courbe (par exemple, les points aux extrémités et le point au paramètre milieu) et de construire le cercle interpolant ces points. Cela est beaucoup plus rapide mais donnera une moins bonne approximation. Pour le calcul des rayons r_{min} et r_{max}, la distance entre un point $f(t)$ et le centre c est notée $d(t)$. Le carré de la distance est donné par la fonction $d^2(t)$. Si $f(t)$ est une courbe B-spline d'ordre k alors d^2 est un polynôme de degré $2(k-1)$ qui s'écrit dans la base B-spline sous la forme

$$d^2(t) = \|f(t) - c\|^2 = \sum_i d_i^2 N_i^{2k-1}(t) \tag{1.10}$$

Il suffit de prendre $r_{min}^2 = \min(d_i^2)$ et $r_{max}^2 = \max(d_i^2)$.

1.3.2.3 Réduction de l'intervalle paramétrique

Le fait que les volumes englobants se coupent n'est pas suffisant pour conclure à une intersection. Il faut donc décomposer le problème. C'est-à-dire que l'on va chercher les intervalles paramétriques de la courbe où il n'y a pas intersection, les éliminer et relancer la méthode avec des segments plus petits. Il existe plusieurs manières de couper les courbes, plus ou moins grossières.

Subdivision au paramètre milieu La méthode la plus ancienne et la plus simple [Lane et Riesenfield, 1980] est de couper les courbes en deux par subdivision au paramètre milieu. Lorsque les volumes englobants se coupent, les courbes sont simplement coupées en deux en insérant le point de la courbe correspondant au milieu de l'intervalle paramétrique de chaque courbe. Sur la figure 1.10, nous recherchons l'intersection entre deux courbes de Bézier. Sur le dessin en haut à gauche, les deux boîtes minmax initiales en pointillés sont en intersection. Les courbes sont donc coupées et les boîtes relatives à chaque segment (en couleur sur la figure) sont construites. Seules deux boîtes se coupent. Le processus est relancé pour les segments inclus dans ces boîtes (dessin en haut à droite). Les autres segments sont retirés de l'étude. L'algorithme est reproduit tant que le critère d'arrêt n'est pas satisfait (dessin du bas).

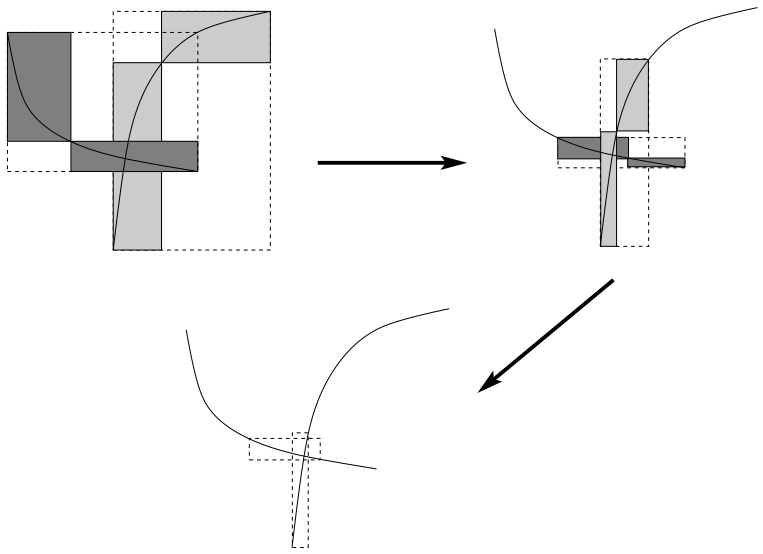

Fig. *1.10* — Détection de l'intersection par subdivision au paramètre milieu : à chaque itération, les courbes sont divisées en deux. Les segments dont les boîtes ne sont pas en intersection avec d'autres boîtes sont supprimés.

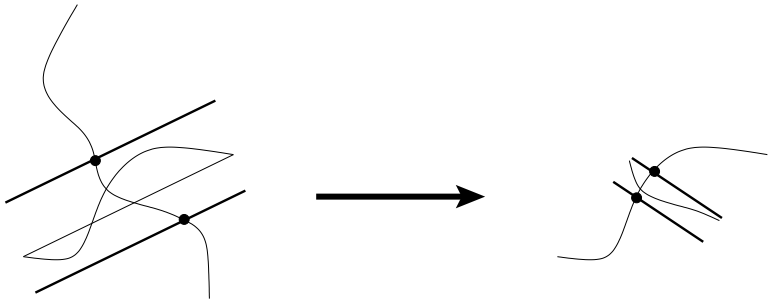

Fig. *1.11* — Détection de l'intersection par élagage de la courbe : à chaque étape, les segments à l'extérieur de la bande sont retirés.

Méthode d'élagage La méthode d'élagage (*clipping*) est présentée dans [Sederberg et Nishita, 1990]. Le principe est de construire un volume contenant une des deux courbes (les auteurs utilisent une bande). Ensuite, les intervalles paramétriques de la deuxième courbe qui ne sont pas dans cette bande sont retirés car il ne peut pas y avoir d'intersection sur ces segments de la courbe. Le paramètre réalisant l'intersection entre la courbe et le bord de la bande est obtenu par dichotomie. La méthode est ensuite répétée en inversant le rôle des courbes (figure 1.11). Dans certains cas, il n'est plus possible de réduire les courbes de cette manière (par exemple, lorsqu'une courbe est entièrement contenue dans la bande de l'autre courbe ou lorsque les courbes se coupent plusieurs fois). Dans ce cas, une des deux courbes est coupée en deux comme pour la méthode précédente et l'algorithme est relancé pour chaque segment.

Une variante utilisant les arcs épais est donnée dans [Sederberg et al., 1989]. Si deux courbes se coupent, leurs arcs se coupent aussi. Au lieu de construire une bande à partir du polygone de contrôle, la bande est construite à partir de l'intersection des deux arcs épais.

Segmentation suivant des points caractéristiques Il existe également d'autres méthodes de découpage où les courbes sont segmentées suivant des points caractéristiques. Ainsi, la méthode de Koparkar [Koparkar et Mudur, 1983] est une variante de la méthode de Lane. Avant d'appliquer la méthode de subdivision, les auteurs recherchent les tangentes horizontales et verticales de la courbe ainsi que ses points d'inflexion. La courbe est ensuite divisée en ces points. Les boîtes englobantes sont définies à partir de ces points et non pas à partir du polygone de contrôle (figure 1.12). Il est également possible d'utiliser des triangles afin d'avoir des englobants plus petits. Le reste de l'algorithme est identique à l'algorithme de Lane.

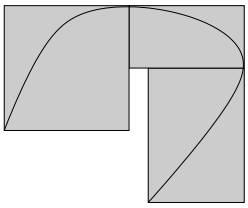

Fig. *1.12* — Méthode de Koparkar : segmentation de la courbe aux points caractéristiques.

Enfin, une dernière méthode est "l'algorithme cocktail" [Kim et al., 1998]. Elle consiste également à segmenter une courbe suivant des points caractéristiques. Ensuite, les arcs de courbes sont approchées par des Bézier rationnelles quadratiques. Si les boîtes des segments de courbes se coupent alors les intersections sont calculées par une méthode algébrique.

Le problème de ces méthodes est que le calcul des points particuliers est coûteux. Nous discutons des intérêts des méthodes de détection dans le paragraphe 1.4.1.

1.3.2.4 Réduction par l'étude du polygone de contrôle

Les méthodes présentées ci-dessus ont été développées pour détecter les intersections entre courbes de Bézier. Nous allons maintenant présenter une méthode spécifique aux courbes B-splines [Daniel, 1989, Daniel, 1992]. Elle est fondée sur le caractère local et la propriété d'enveloppe convexe de ces courbes (propriétés 4 et 5). Si k points de contrôle Q_{i-k+1} à Q_i sont situés du même côté d'une droite dans le plan (ou d'un plan dans l'espace), alors l'arc de courbe défini sur l'intervalle paramétrique élémentaire $[t_i, t_{i+1}]$ ne traverse pas la droite. Cette étude peut être menée tout le long de la courbe afin de ne conserver que les intervalles intéressants [Nicolas, 1995]. Cette méthode ne calcule pas l'intersection exacte de deux courbes mais permet de localiser l'intervalle paramétrique où elle se situe. Il s'agit en

fait d'une méthode appliquée en pré-traitement afin d'appliquer les méthodes précédentes sur des intervalles plus restreints.

Soient deux courbes B-splines $f^0(t)$ et $f^1(t)$. Nous voulons retirer les intervalles paramétriques élémentaires de f^0 ne coupant pas f^1. Nous construisons la boîte minmax de f^1 et nous considérons les quatre droites délimitant la boite. Elles forment une partition du plan en neuf régions (figure 1.13). A chaque région, nous associons un code à quatre bits correspondant à sa position par rapport aux quatre droites. Pour chaque point de contrôle Q_i, nous lui affectons le code z_i de la région où il se situe. Nous voyons que pour que deux points de contrôle soient du même côté de la boite, il faut que le résultat de l'opération logique *et* des bits des codes des deux points soit égale à 1 pour au moins un bit.

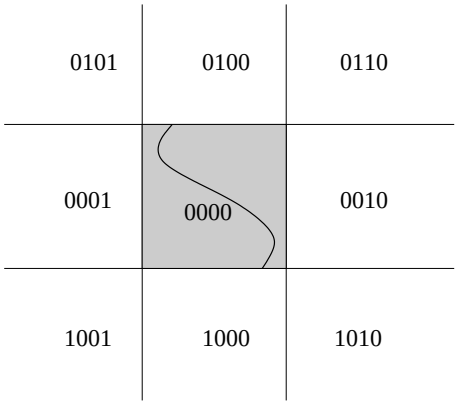

Fig. *1.13* — Partition d'un plan à partir d'une boîte minmax : un masque est affecté à chaque zone.

La méthode consiste donc à partir du premier point de contrôle Q_0 et à calculer pour chaque ensemble de k points consécutifs $z = z_i \wedge \ldots \wedge z_{i+k-1}$. Si z n'est pas nul (c'est-à-dire, au moins un des bits est non nul), l'intervalle paramétrique $[t_i, t_{i+1}]$ est à l'extérieur de la boite et peut être retiré. Si z vaut 0, il y a intersection potentielle et l'intervalle doit être conservé. Lorsque nous passons d'un intervalle $[t_{i-1}, t_i]$ à l'extérieur à un intervalle $[t_i, t_{i+1}]$ coupant la boite, nous disons que t_i est un paramètre entrant. De même, lorsque nous passons d'un intervalle en intersection potentielle à un intervalle à l'extérieur, t_i est appelé un paramètre sortant. Lorsque nous avons traité tous les points de contrôle, la courbe f^0 passant dans la boite de f^1 est alors réduite à une liste d'intervalles $[t_i, t_j]$ où les t_i sont des paramètres entrants et les t_j des paramètres sortants. Ce principe est similaire à l'algorithme de Cohen Sutherland [Foley et al., 1995] où les segments passant dans une fenêtre sont délimités par des points entrants et des points sortants.

Par exemple, sur la figure 1.14, nous avons représenté le polygone de contrôle d'une B-spline cubique et la boîte minmax d'une deuxième courbe. Les quatre premiers points de contrôle Q_0 à Q_3 sont situés au-dessus de la boîte donc le segment défini par ces points est à l'extérieur de la boite. Q_4 est dans la boîte donc le segment défini avec les points Q_1, Q_2, Q_3, Q_4 peut couper la boite. Q_1, Q_2, Q_3, Q_4 doivent être conservés dans l'intervalle d'étude.

Par contre, Q_0 peut être retiré. De même, les deux derniers points Q_{11} et Q_{12} peuvent être retirés. Sur la figure, les points noirs sont retirés alors que les blancs sont conservés. L'intervalle paramétrique est réduit à $[t_4, t_{11}]$.

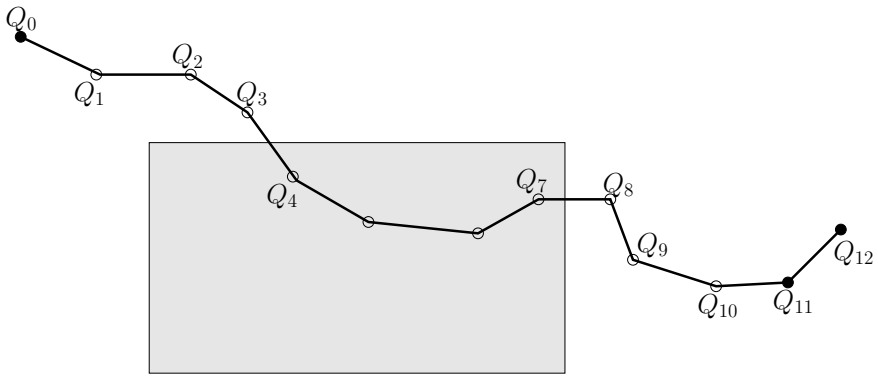

Fig. *1.14* — Réduction de l'intervalle d'étude : les points de contrôle Q_0, Q_{11} et Q_{12} peuvent être retirés.

Ceci constitue la première étape de la méthode puisque nous pouvons répéter l'algorithme en inversant le rôle des courbes. Nous construisons les boîtes minmax de tous les segments restants de f^0 et nous réduisons l'intervalle paramétrique de f^1 par rapport à chaque boite. L'algorithme est répété en alternant à chaque fois le rôle des courbes jusqu'à ce que les intervalles soient réduits au maximum.

Cette méthode ne permet pas de calculer précisément une intersection mais elle peut être utilisée pour éliminer rapidement des portions de courbe où il ne peut pas y avoir intersection. Nous l'utiliserons donc comme un pré-traitement avant d'appliquer les méthodes vues précédemment sur les intervalles restants.

1.3.3 Traitement des auto-intersections

Il n'est pas possible de calculer directement les auto-intersections d'une courbe de Bézier ou d'une courbe B-spline avec les méthodes présentées. La solution de ce problème est donc dans un premier temps de couper la courbe en plusieurs segments sur lesquels nous sommes sûrs qu'il ne peut pas y avoir d'auto-intersection puis d'appliquer les méthodes de détection sur chaque segment. Nous donnons ici un critère issu de [Andersson et al., 1998] garantissant la non-existence d'une auto-intersection sur un segment de courbe. Ce critère se base sur le résultat suivant [Ho et Cohen, 2000] :

Résultat 4 *Le plus petit cône contenant les tangentes d'une courbe possédant une auto-intersection a un angle supérieur à π.*

Les tangentes d'une courbe paramétrique sont données par sa courbe hodographe (définition 1). Le polygone de contrôle de l'hodographe est donné par les branches $dQ_i =$

$Q_{i+1} - Q_i$ du polygone de contrôle de la courbe. A partir du résultat 4, une condition suffisante pour qu'il n'y ait pas auto-intersection est que les branches du polygone de contrôle soient contenues dans un cône d'angle strictement inférieur à π. C'est-à-dire, il existe une droite passant par l'origine telle que les points de contrôle de l'hodographe soient tous situés du même côté de la droite (figure 1.15).

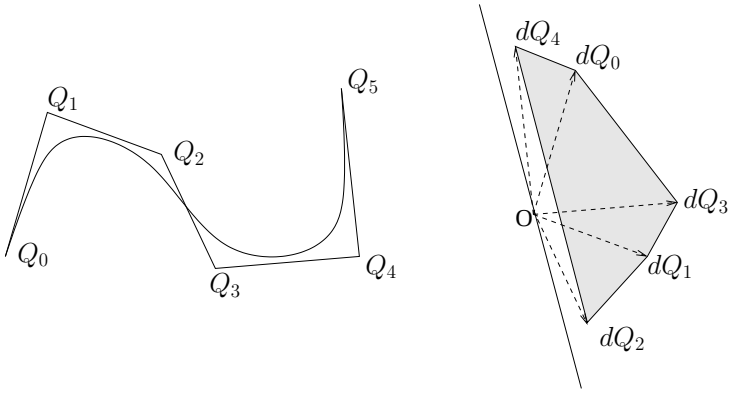

Fig. *1.15* — Points de contrôle d'une courbe et enveloppe convexe du polygone de contrôle de son hodographe : la courbe n'admet pas d'auto-intersection car l'hodographe est toujous du même côté par rapport à la droite passant par l'origine.

Soient f une courbe B-spline de polygone de contrôle $(Q_i)_{i=0}^{i=m}$ et dQ_i pour $i = 0$ à $m - 1$. Le critère de non auto-intersection est défini sous la forme suivante équivalente [Andersson et al., 1998] :

Critère 1 *f n'admet pas d'auto-intersection si* $\exists u \in \mathbb{R}^2$ *tel que* $\min_{0 \leq i \leq m-1} (dQ_i.u) > 0$.

Ce critère donne une condition suffisante mais non nécessaire de non auto-intersection. Nous utiliserons ce critère pour segmenter la courbe en morceaux sur lesquels il n'y a pas d'auto-intersection. Nous partons du premier point de contrôle de la courbe et construisons le cône contenant les branches du polygone de contrôle. Si l'ouverture du cône devient supérieure à π en ajoutant la branche dQ_i alors, la courbe est segmentée par insertion du nœud t_{i-1} pour qu'il soit de multiplicité k (résultat 1). Nous avons alors deux courbes. La première vérifie le critère de non auto-intersection. Nous relançons la méthode sur la deuxième courbe. A la fin, la courbe initiale est segmentée en morceaux vérifiant tous le critère de non auto-intersection.

Les méthodes géométriques peuvent alors être uniquement appliquées aux segments de courbe disjoints. En revanche, si deux segments ont une extrémité commune, nous ne pouvons pas appliquer ces méthodes qui détecteraient cette extrémité comme une intersection. Pour cela, nous appliquons un deuxième critère présenté dans [Andersson et al., 1998] garantissant la non-existence d'intersections en dehors de leur extrémité commune.

Nous notons f^0 et f^1 deux courbes B-splines ayant une extrémité commune $f^0(0) = f^1(0)$. Les courbes passent par les points de contrôle aux extrémités donc $Q_0^1 = Q_0^0$. Le critère d'Andersson donne une condition suffisante pour que la courbe f^1 se trouve toujours du même

côté par rapport à f^0. Nous notons $Q' = \{Q_j^1 - Q_i^0, \quad 0 \leq i \leq m^0, \quad 0 \leq j \leq m^1\} - \{Q_0^1 - Q_0^0\}$ l'ensemble des vecteurs reliant les points de contrôle de f^0 aux points de contrôle de f^1. La différence $\{Q_0^1 - Q_0^0\}$ n'est pas prise en compte sinon nous trouverions systématiquement une intersection en Q_0^0. Nous supposons également que $f^{0\prime}(t) \times f^{1\prime}(t) \neq 0$ ou $f^{0\prime}(t).f^{1\prime}(t) < 0$. Ceci correspond au cas où les deux courbes sont tangentes en Q_0^0 créant un point de rebroussement.

Une condition suffisante pour que f^1 soit toujours du même côté de f^0 est que l'enveloppe convexe du polygone de contrôle de f^1 soit toujours du même côté de l'enveloppe convexe du polygone de contrôle de f^0. Ceci est vérifié s'il existe une droite passant par l'origine ne coupant pas l'enveloppe convexe de Q' (figure 1.16).

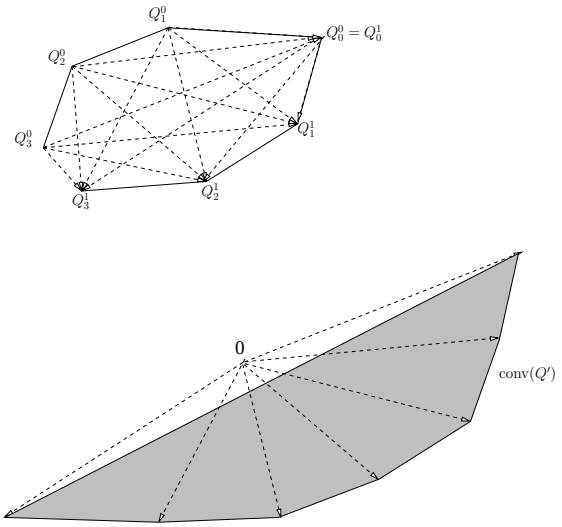

Fig. *1.16* — Non intersection de courbes ayant une extrémité commune : l'enveloppe des vecteurs de Q' construits à partir des points de contrôle ne contient pas l'origine.

Le critère de non intersection peut s'écrire sous la forme suivante :

Critère 2 *f^0 et f^1 n'admettent pas d'intersection si* $\exists u \in \mathbb{R}^2$ *tel que* $\min_{q \in Q'}(q.u) > 0$.

Nous appliquons ce critère pour les segments de courbe ayant une extrémité commune obtenus après application du critère précédent. Nous partons des points de contrôle Q_0^0 et Q_0^1 puis pour un point Q_i^0, nous parcourons tous les Q_j^1. Si le critère 2 n'est pas vérifié, f^1 est segmentée. Si le critère est vérifié pour tous les Q_j^1, nous passons au point Q_i^0 suivant et recommençons jusqu'à $Q_{m^0}^0$. A la fin, le critère est vérifié pour tous les segments de courbe adjacents. Les tests d'intersection peuvent être appliqués à tous les segments disjoints obtenus lors de cette opération.

Dans le cas où $f^{0\prime}(t).f^{1\prime}(t) > 0$ et $f^{0\prime}(t) \times f^{1\prime}(t) = 0$, cela signifie que les courbes sont tangentes et que Q_0^0 est un point de rebroussement. Les trois points Q_0^0, Q_1^0 et Q_1^1 sont alors alignés avec Q_1^0 et Q_1^1 du même côté par rapport à Q_0^0 et Q_0^0 doit être considéré comme un point d'intersection.

1.4 Evaluation des méthodes

1.4.1 Résultats existants pour les courbes de Bézier

Nous avons déjà présenté les limites des méthodes numériques de détection dans le paragraphe 1.3.1.3. Notamment, elles ne sont pas adaptées au traitement des intersections singulières et visuelles et au traitement des auto-intersections. De leur côté, les méthodes géométriques sont des méthodes itératives fondées sur l'étude du polygone de contrôle. Elles ne calculent pas des solutions exactes mais des intersections à une tolérance près. La récursivité est arrêtée lorsque cette tolérance est atteinte. La méthode est donc beaucoup plus robuste puisque l'on considère qu'il y a intersection dès que la distance entre les courbes est inférieure à la tolérance. Cela permet de traiter tous les types d'intersection de la même façon et donc de prendre en compte les intersections singulières et visuelles. Les méthodes géométriques permettent également de traiter les auto-intersections en appliquant les critères présentés dans le paragraphe 1.3.3.

En ce qui concerne la rapidité des méthodes, une comparaison a été faite dans [Sederberg et Parry, 1986] et [Sederberg et Nishita, 1990] pour les courbes de Bézier entre une méthode algébrique, les méthodes de Lane et de Koparkar et la méthode d'élagage. Il apparaît que les méthodes algébriques sont plus rapides pour les degrés inférieurs à quatre. Au-delà, les méthodes géométriques deviennent plus performantes. Cela vient du fait que la complexité en fonction du degré des méthodes algébriques est quadratique alors qu'elle est linéaire pour les méthodes géométriques. D'après Sederberg, la méthode de Koparkar donne de meilleurs résultats que la méthode de subdivision de Lane. Mais cette méthode requiert le calcul de points particuliers de la courbe. Cela demande un temps de calcul non négligeable et pose le problème de la méthode utilisée pour obtenir ces points particuliers.

La méthode géométrique la plus rapide est la méthode d'élagage. Les bandes sont des englobants plus coûteux à calculer mais plus fins que les boîtes minmax. Le découpage se fait aussi plus finement. On effectue donc beaucoup moins d'itérations et de tests d'intersections qu'avec les autres méthodes. Les coûts en nombre d'opérations pour construire et comparer des englobants sont donnés dans [Sederberg et al., 1989] pour des courbes de Bézier cubiques. Il apparaît que le volume le plus facile à construire et à manipuler est la boîte minmax (les seules opérations effectuées sont des comparaisons). Des autres volumes présentés, le plus cher en nombre d'opérations est l'arc épais (106 opérations pour la construction contre 12 pour les boîtes minmax pour des courbes de Bézier cubiques) [Sederberg et al., 1989]. Les auteurs comparent l'efficacité des englobants en donnant des vitesses de convergence. Ces vitesses correspondent à la vitesse à laquelle l'aire des englobants tend vers 0. La boîte minmax a la convergence la plus lente ($O(h)$), la bande et l'enveloppe convexe sont en $O(h^2)$ et l'arc épais est en $O(h^3)$.

1.4.2 Résultats obtenus pour les courbes B-splines

Dans les articles cités précédemment, les méthodes sont appliquées à des courbes de Bézier. Les auteurs font varier le degré des courbes mais elles sont toujours définies avec un petit nombre de points. De plus, les intersections détectées sont des intersections franches. Les cas de tangence et de recouvrement sont peu traités.

Le premier problème est le choix d'un englobant. Avec les courbes de Bézier, il est possible de calculer des englobants fins car les courbes sont définies avec peu de points. Pour les courbes B-splines définies avec un grand nombre de points, des englobants comme des boîtes inclinées où l'on cherche une direction optimale ou des arcs épais sont trop coûteux. En effet, pour le premier, l'algorithme est de complexité $O(n^3)$ et pour le deuxième, le calcul des rayons se fait en $O(n^2)$. Nous avons évalué dans le tableau 1.1 le nombre d'opérations à effectuer en fonction du nombre de points de contrôle pour calculer différents englobants et évaluer la position d'un point par rapport à un englobant. $\oplus$ représente le nombre d'additions et de soustractions, $\otimes$ le nombre de multiplications et $\lessgtr$ le nombre de comparaisons. Les temps d'exécution de chaque opération sont les mêmes. Nous voyons que les englobants les moins coûteux sont la boîte minmax et la bande aussi bien pour la construction (qui est en $O(n)$) que pour la comparaison. Pour cette raison, nous testerons uniquement ces deux types d'englobants.

	Construction du volume	Classification d'un point
boîte minmax	$4n\lessgtr$	$4\lessgtr$
Bande	$2n+4\oplus$, $2n+2\otimes$, $2\lessgtr$	$2\oplus$, $2\otimes$, $2\lessgtr$
boîte optimale	$O(n^3)$	$2\oplus$, $4\otimes$, $4\lessgtr$
Arc épais	$O(n^2)$	$5\oplus$, $6\otimes$, $4\lessgtr$

Tab. *1.1* — Nombre d'opérations effectuées pour différents volumes englobants.

Nous présentons des résultats obtenus pour des courbes B-splines cubiques définies avec un grand nombre de points (plusieurs centaines de points). Les tests ont été faits pour les méthodes de Lane et de Sederberg avec des boîtes minmax et des bandes orientées suivant l'axe joignant les extrémités. Des tests ont également été effectués avec la méthode de Koparkar mais la recherche des points particuliers est beaucoup trop coûteuse et demande beaucoup plus de temps que la recherche des points d'intersection. Parmi les critères d'arrêt présentés au paragraphe 1.3.2.1, nous choisissons d'arrêter l'algorithme lorsque la courbe est assimilable à un segment de droite. Ce critère est plus performant que celui sur la taille des englobants car il permet de faire moins d'itérations. En plus, le point d'intersection peut être approché par l'intersection des deux segments. La précision ϵ_{num} pour le critère d'arrêt est liée à la précision des données et est fixée à $\epsilon_{num} = 10^{-3}$. Il s'agit de la précision absolue que nous avons sur les données en entrée.

Dans un premier temps, nous calculons des intersections à la précision des données, c'est-à-dire à 10^{-3} près. Les résultats du tableau 1.2 sont obtenus pour des courbes B-splines d'une

centaine de points de contrôle. Les intersections à détecter sont des intersections régulières. Nous avons testé les méthodes avec et sans le pré-traitement présenté au paragraphe 1.3.2.4. Les temps sont donnés par rapport au meilleur temps. Nous constatons que la méthode de Lane donne les moins bons résultats. La méthode de Sederberg est plus efficace avec les bandes. Nous obtenons le même classement que [Sederberg et Nishita, 1990] mais les écarts sont moins importants. Dans tous les cas, le pré-traitement améliore les résultats en réduisant les temps de calcul mais ne change pas les conclusions.

Méthode	Sans pré-traitement	Avec pré-traitement
Lane	1.78	1.32
Sederberg (boite)	1.55	1.24
Sederberg (bande)	1.11	1

Tab. ***1.2*** — Temps de calcul relatifs associés à la détection d'intersections régulières de modélisation.

Nous avons également fait les tests pour des intersections singulières (tangence et recouvrement de courbes). Nous présentons en fait des résultats obtenus pour des courbes de niveau extraites de données bathymétriques. Nous avons utilisé des jeux de 200 à 400 courbes où les intersections à détecter sont surtout des intersections singulières. Nous avons deux problèmes qui n'apparaissaient pas dans le paragraphe 1.4.1. Premièrement, avoir une courbe définie avec beaucoup de points fait que le calcul d'englobants est coûteux. Deuxièmement, la détection d'une intersection singulière pose des problèmes au niveau de la segmentation des courbes car lorsque deux courbes se recouvrent ou sont très proches, la première courbe est souvent incluse dans le volume englobant la deuxième courbe. Par conséquent, il faut souvent faire beaucoup d'itérations avant de pouvoir conclure.

Nous donnons les temps de calcul et le nombre d'itérations dans le tableau 1.3. Pour les trois méthodes, l'utilisation du pré-traitement améliore les temps. Le nombre d'itérations effectuées est beaucoup plus petit car dans la majeure partie des cas où il n'y a pas d'intersection, seul le pré-traitement est effectué. Nous remarquons que la méthode d'élagage avec bande donne les moins bons temps. Les différences avec les deux autres méthodes sont accentuées lorsque nous effectuons un pré-traitement. Cela signifie que la détection des intersections singulières est moins rapide avec des bandes qu'avec des boites. En effet, le calcul des bandes est plus coûteux que le calcul des boites minmax mais ne permet pas de segmenter plus finement les courbes quand elles sont tangentes. Les courbes ne peuvent pas être segmentées par élagage et doivent être subdivisées. C'est pour cela que les résultats entre la méthode de subdivision et la méthode d'élagage avec des boites donnent des résultats très proches en temps.

Enfin, nous avons appliqué les trois méthodes de détection aux auto-intersections. Les courbes sont segmentées à l'aide des critères 1 et 2 puis les intersections sont détectées entre les segments. Les résultats sont assez proches (tableau 1.4) mais les boîtes minmax sont

	Sans pré-traitement		Avec pré-traitement	
Méthode	Temps de calcul	Itérations	Temps de calcul	Itérations
Lane	1.19	41	1.03	1.89
Sederberg (boite)	1.16	40	1	1
Sederberg (bande)	1.47	43	1.33	4.22

Tab. *1.3* — Temps de calcul relatifs associés à la détection d'intersections singulières de modélisation.

toujours plus efficaces que les bandes. Les raisons sont les mêmes que précédemment même si les écarts sont peu importants. En fait, les temps sont équivalents pour toutes les méthodes car la plus grande partie du calcul est passée à appliquer le critère de non auto-intersection. La détection est faite sur de petits segments, donc relativement rapide.

Méthode	Temps
Lane	1
Sederberg (boite)	1.11
Sederberg (bande)	1.23

Tab. *1.4* — Temps de calcul relatifs associés à la détection des auto-intersections de modélisation.

Pour finir, nous avons calculé les intersections et auto-intersections visuelles sur les mêmes échantillons que pour les tableaux 1.3 et 1.4. La distance de lisibilité est fixée à $\epsilon_{vis} = 2.10^{-2}$, correspondant à l'épaisseur d'un trait de crayon. Pour les auto-intersections, nous avons obtenu les mêmes résultats que dans le tableau 1.4. Etant donné qu'il y a plus d'intersections visuelles que d'intersections de modélisation, il faut effectuer plus d'itérations et les temps de détection sont donc légèrement plus élevés. Nous comparons les différentes méthodes dans le tableau 1.5. Les méthodes utilisant des boites donnent des résultats très proches. Par contre, les différences avec les bandes sont accentuées : le rapport de temps est beaucoup plus important. Le meilleur temps est obtenu avec la méthode de subdivision et le pré-traitement.

	Sans pré-traitement		Avec pré-traitement	
Méthode	Temps de calcul	Itérations	Temps de calcul	Itérations
Lane	1.16	26	1	1.86
Sederberg (boite)	1.12	25	1.03	1
Sederberg (bande)	3.87	30	2.27	8

Tab. *1.5* — Temps de calcul relatifs associés à la détection d'intersections visuelles.

Pour tous les tests effectués, nous constatons que l'utilisation de boites minmax donnent

de meilleurs résultats pour la détection d'intersections singulières. Les boites minmax sont des volumes englobants grossiers mais robustes et rapides à calculer par rapport aux bandes. Dans tous les cas de figure, la technique de pré-traitement permet d'améliorer les temps de calcul. Pour les deux méthodes de segmentation de courbes utilisées, nous voyons que nous avons des résultats semblables. Compte tenu du type d'intersection à détecter, la recherche d'un point de segmentation ne réduit que très peu les temps de détection car chaque courbe est souvent incluse dans le volume englobant l'autre courbe et la segmentation se fait par subdivision.

1.5 Conclusion

Dans ce chapitre, nous nous sommes intéressés aux méthodes de détection des intersections entre courbes paramétriques, plus particulièrement, entre courbes B-splines. Les intersections peuvent être de plusieurs sortes. Il peut apparaître des intersections franches ou des intersections singulières (tangence, recouvrement de segments de courbe). Suivant les applications considérées, nous pouvons traiter des intersections de modélisation ou des intersections visuelles. Le traitement des intersections franches entre courbes de Bézier est un problème largement abordé dans la littérature. Nous nous sommes attardés au cas des intersections singulières entre courbes B-splines et aux auto-intersections. Les contraintes, dans ce cas, ne sont pas les mêmes puisque les intersections singulières et les auto-intersections posent des problèmes de détection et demandent des méthodes robustes.

Nous avons présenté les principales méthodes de détection existantes. Elles sont réparties en deux catégories. Les premières sont fondées sur l'utilisation de méthodes numériques et la résolution de systèmes d'équations. Ces méthodes sont rapides et permettent de calculer des solutions très précises. Malheureusement, elles sont peu robustes et ne permettent pas de traiter les intersections singulières et les auto-intersections. La méthode de projection permet de résoudre certaines de ces difficultés mais elle est très lente puisqu'il faut résoudre deux équations et calculer des enveloppes convexes à chaque étape.

La deuxième catégorie regroupe les méthodes géométriques. Ces méthodes sont particulièrement adaptées aux courbes B-splines. Elles utilisent les propriétés géométriques des courbes et permettent de traiter les intersections singulières et les intersections visuelles en tenant compte de la distance de lisibilité. Nous avons également présenté une méthode spécifique aux courbes B-splines permettant de définir un intervalle paramétrique réduit sur lequel il y a intersection potentielle. Cette méthode utilise la propriété d'enveloppe convexe locale et est appliquée en pré-traitement avant d'utiliser les méthodes précédentes. Enfin, nous avons présenté des critères pour traiter les auto-intersections.

Dans la dernière partie, nous avons appliqué les méthodes géométriques à des courbes B-splines définies avec un grand nombre de points. Nous avons tout d'abord appliqué ces méthodes pour détecter des intersections régulières. Dans ce cas, les conclusions sont les mêmes que pour les courbes de Bézier. Les meilleurs temps sont obtenus avec les englobants

les plus fins et avec la méthode d'élagage réduisant les courbes plus finement que la méthode de subdivision. Nous avons ensuite appliqué les mêmes méthodes à la détection des intersections singulières. Les meilleurs résultats sont obtenus avec les boîtes minmax. Ce sont des englobants plus rapides à calculer et plus robustes que les bandes. Les conclusions sont les mêmes pour les intersections de modélisation et de visualisation. Nous avons également vu que l'application de la méthode de pré-traitement permet de réduire les temps de calcul en conservant les mêmes résultats.

La détection d'intersections singulières dans un grand ensemble de courbes demande donc des méthodes simples et robustes. Ces résultats peuvent être appliqués à la détection des intersections pour le traitement des cartes marines. En effet, les courbes de profondeur sont représentées par des courbes B-splines définies avec un grand nombre de points et les intersections à détecter sont surtout des intersections visuelles et singulières. Malgré tout, ces méthodes ne sont pas suffisantes pour le traitement d'une carte complète.

Sur une carte marine, nous avons un ensemble de plusieurs centaines de courbes. Il est donc important de minimiser les calculs afin d'accélérer la détection. Pour cela, il est nécessaire de segmenter la carte en différentes zones afin de limiter les tests d'intersections. Cette méthode sera fondée sur le même principe que la méthode de pré-traitement utilisant des boites minmax afin d'éliminer le plus rapidement possible des zones où il ne peut pas y avoir de conflit. Dans le chapitre suivant, nous présentons une méthode de détection adaptée au traitement des grands ensembles de courbes.

CHAPITRE 2

Application à la généralisation cartographique des cartes marines

2.1 Introduction

La carte marine permet aux navigateurs de se localiser et de déterminer leur route tout en assurant la sécurité de la navigation. Elle retranscrit les caractéristiques principales du fond marin ainsi que les objets et les informations nécessaires à la navigation (bouées, câbles, rails de navigation). Les fonds marins sont définis notamment par l'intermédiaire :

- des sondes, points de profondeur identifiés par des coordonnées x et y dans un plan et par une profondeur $z > 0$;
- des isobathes, lignes de niveau reliant les points de même profondeur ;
- des traits de côte, lignes obtenues par l'intersection de la topographie terrestre avec le niveau des plus hautes mers.

En France, le Service Hydrographique et Océanographique de la Marine (SHOM) est chargé de la construction, de la gestion et de la diffusion des cartes marines aussi bien sous la forme d'une carte papier traditionnelle que sous la forme de supports électroniques. Il est responsable de l'information contenue sur les cartes et de leur mise à jour. La première étape de la construction est l'acquisition des données. Actuellement, celle-ci se fait principalement avec des sondeurs multifaisceaux (SMF). Ce sont des appareils permettant de mesurer la profondeur des fonds sur toute une fauchée perpendiculaire à l'axe du navire. La largeur de cette fauchée est de 2 à 7 fois la profondeur. Les sondes relevées par les SMF fournissent une couverture bathymétrique totale du fond. La quantité de données relevées est très importante : un SMF peut relever environ un million de sondes au kilomètre carré. Pour pouvoir être utilisées, les données sont corrigées (les positions et les profondeurs sont corrigées en tenant compte entre autres des lacets et des roulis du bateau, de la marée) puis nettoyées (les données aberrantes ou redondantes sont supprimées). Par conséquent, nous supposons

que les données sont exactes. Les isobathes sont ensuite construites à partir des sondes par triangulation et interpolation. Le volume de données étant très dense, elles ne peuvent pas toutes être reproduites sur la carte. Des modifications et des suppressions sont nécessaires afin de sélectionner les éléments intéressants et de fournir une information lisible et pertinente.

La deuxième étape consistant à choisir les sondes et les isobathes est la généralisation cartographique. A la fin du traitement, les informations sont recueillies dans une base de données bathymétriques afin d'archiver les données numériques et de les mettre à la disposition des utilisateurs. Au niveau informatique, une carte marine est représentée par un ensemble de fichiers de sondes et de lignes polygonales, chaque ligne étant munie d'un identificateur spécifiant le type de ligne représentée (isobathe, câble, trait de côte ...) et d'un identificateur spécifiant si la ligne est ouverte ou fermée.

C'est lors de ces étapes de construction et de généralisation des isobathes qu'il peut apparaître des conflits. La suppression de ces conflits se fait en deux étapes. Dans un premier temps, il s'agit de détecter et localiser l'ensemble des conflits sur la carte. Ensuite, ces conflits doivent être corrigés en respectant des contraintes de déplacement. Dans ce chapitre, nous présentons une méthode de détection des conflits adaptée à la généralisation des cartes marines. L'objectif de cette méthode est, à partir d'un ensemble de courbes B-splines représentant les isobathes, d'identifier les zones de conflit où les contraintes ne sont pas respectées afin de les corriger dans un deuxième temps.

Dans le paragraphe 2.2, nous expliquons le principe de la généralisation cartographique et plus précisément la généralisation des isobathes. Nous définissons les contraintes à respecter et les types d'intersection à détecter. En particulier, la méthode de détection doit être adaptée au traitement d'un grand nombre de données, une carte pouvant contenir, suivant l'échelle, plusieurs centaines, voire milliers de courbes. La méthode doit également être robuste car il s'agit de détecter principalement des intersections visuelles ou des recouvrements de courbes.

Dans le paragraphe 2.3, nous détaillons la méthode mise en place. La détection se fait en deux parties. D'abord, nous segmentons la carte à l'aide d'un quadtree afin de localiser les intersections possibles. Ensuite, nous détectons ces intersections proprement dites dans chaque cellule en approchant les segments de courbe par des lignes polygonales à l'aide de schémas de subdivision. Nous présentons des résultats obtenus sur des ensembles d'isobathes extraits de cartes marines. Ces résultats sont discutés en fonction des paramètres du quadtree et de la précision de l'approximation.

2.2 La généralisation cartographique des cartes marines

2.2.1 La généralisation cartographique

Une base de données géographiques est une base dont les données possèdent des coordonnées géographiques permettant de les localiser. Il existe trois grands types de bases de données géographiques [Ruas et Libourel, 2002] :

- *les bases de données thématiques* décrivant un ensemble de ressources ou d'activités insérées dans des zones géographiques. Elles sont utilisées, par exemple, pour des études économiques ou sociologiques ;
- *les bases de données topographiques* décrivant la localisation et la nature d'entités liées à l'activité humaine ;
- *les bases de données environnementales* qui décrivent les ressources naturelles (géologie, végétation. . .).

La généralisation consiste à extraire l'information importante dans les différentes bases de données en fonction d'un besoin. Par exemple, une information sans intérêt pour l'application pressentie est omise. Les informations moins importantes sont réduites ou simplifiées alors que les informations plus importantes sont mises en valeur en les amplifiant ou en les caricaturant. La généralisation d'informations géographiques est composée de deux grands thèmes : la généralisation modèle et la généralisation cartographique. La généralisation modèle peut être vue comme le processus d'interprétation conduisant à une vision d'un phénomène à un plus haut niveau – c'est-à-dire à une plus petite échelle. Ce paradigme est toujours le premier utilisé, que ce soit en généralisation de données spatiales ou statistiques. En second lieu, la généralisation cartographique peut être vue comme une série de transformations de l'information spatiale sous forme de représentation graphique, afin de facilité la lisibilité et la compréhension des données en tenant compte de l'utilisation finale du produit [Müller et al., 1995].

En cartographie, la généralisation est divisée en généralisation objet et en généralisation cartographique. La généralisation objet fait partie de la généralisation modèle. Elle consiste à adapter les objets en fonction de la quantité d'information retenue tout en conservant l'information sémantique. Les opérations se font sous des contraintes logiques. Des structures hiérarchiques sont souvent utilisées pour fusionner ou subdiviser les données. Les principaux opérateurs de généralisation objet sont l'élimination, l'agrégation et le regroupement de classes. La généralisation objet est souvent utilisée comme une étape de pré-traitement pour la généralisation cartographique. Elle relève plutôt de la gestion de la base de données car les contraintes visuelles et esthétiques ne sont pas prises en compte. Malgré tout, elle a quand même des conséquences sur le résultat visuel final puisqu'elle a une influence sur la généralisation cartographique. Les différentes étapes de la généralisation sont résumées figure 2.1. Dans notre problème, nous ne sommes pas concernés par ce type de généralisation puisque nous nous attachons surtout à résoudre des problèmes de visualisation.

Le deuxième thème est la généralisation cartographique. La généralisation cartographique consiste à réduire la complexité d'une carte lorsque l'on passe à une échelle plus petite en mettant en valeur l'information essentielle et en supprimant ce qui n'est pas important tout en gardant les liens et les relations entre les différents objets et en préservant l'esthétisme de la carte [Weibel et Dutton, 1999]. Les objets sont retenus en fonction de l'échelle de la carte. Leurs caractéristiques géométriques sont lissées ou amplifiées en fonction de leur pertinence. Les différents opérateurs de généralisation cartographique sont réparties en sept parties

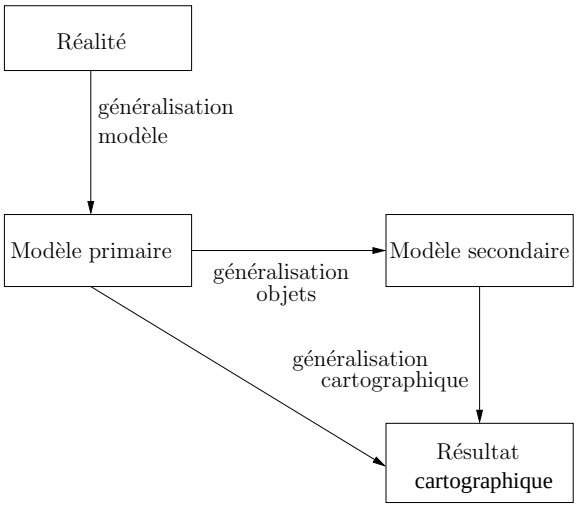

Fig. *2.1* — Les différentes étapes de la généralisation d'après [Weibel et Dutton, 1999].

[Fei, 2002] :

- simplification (lissage) ;
- agrandissement (élargissement) ;
- déplacement ;
- agrégation ;
- sélection (et suppression) ;
- classification (changement de symboles) ;
- caricature (exagération, atténuation).

Les trois premiers opérateurs sont purement géométriques tandis que les quatre autres sont à la fois des opérateurs de généralisation objet et cartographique. Il est à noter que les résultats obtenus dépendent de l'ordre d'utilisation des opérateurs [Huet, 1996].

2.2.2 Contraintes spécifiques à la cartographie marine

En cartographie marine, la généralisation consiste à modifier et à supprimer les courbes et les sondes afin de mettre en évidence le relief sous-marin et ses dangers [Saux et al., 2002]. Sur la figure 2.2, nous avons à gauche les données initiales (sondes et isobathes) où l'on peut voir les fauchées du SMF et à droite le résultat après généralisation manuelle.

La carte marine devant assurer juridiquement la sécurité de l'utilisateur, la généralisation doit se faire en vérifiant certaines contraintes pour assurer la lisibilité de la carte et la sécurité de la navigation imposées par l'Organisation Hydrographique Internationale [OHI, 1988]. Elles dépendent directement de la quantité d'information et de l'échelle de la carte. En reprenant la classification proposée par Beard [Beard, 1991] (contraintes applicatives, graphiques, structurales et procédurales), nous identifions :

- *la contrainte applicative de sécurité* : la représentation issue du processus de généralisation doit fournir une enveloppe haute de la représentation initiale. En conséquence,

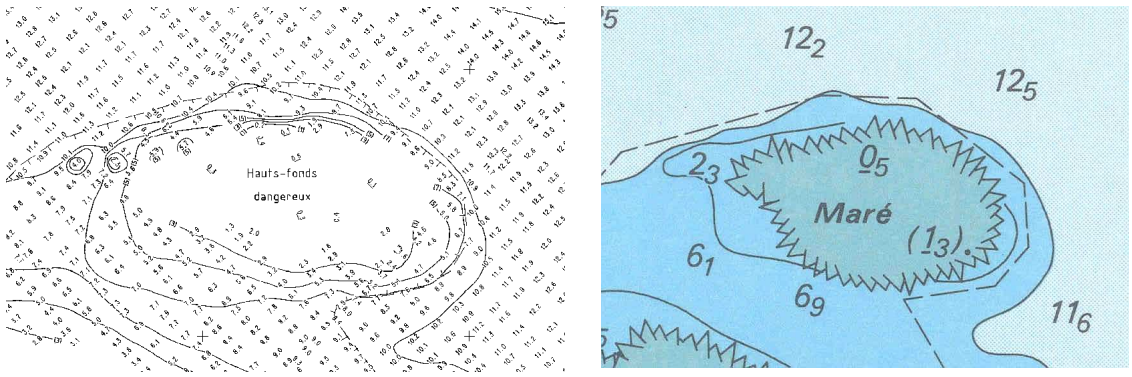

Fig. *2.2* — Exemple de généralisation de la bathymétrie.

les profondeurs sur la carte ne doivent jamais être supérieures aux profondeurs réelles afin d'assurer la sécurité de la navigation (figure 2.3) ;

- *la contrainte graphique de lisibilité* : la représentation généralisée doit respecter les règles cartographiques de lisibilité sur les sondes et les isobathes énumérées dans les normes définies par le SHOM [SHOM, 1984]. Ces normes concernent la densité de sondes sur la carte et la distance minimale entre les différents objets, dont les isobathes, et les types de traits représentant les objets afin de faciliter la lecture et d'éviter toute ambiguïté ;
- *la contrainte structurale géomorphologique* : l'utilisation de la carte marine impose que le caractère géomorphologique des fonds (pente, rugosité) soit au mieux maintenu. Dans le même temps, les éléments caractéristiques du relief (bosses, talwegs, chenaux) doivent être conservés et mis en évidence ;
- *la contrainte procédurale de cohérence* : lors des procédures à effectuer, des impératifs concernant les objets doivent être respectés. Par exemple, lors de l'agrégation de deux isobathes, l'une au moins doit être fermée.

En général, une généralisation ne respecte pas l'ensemble des contraintes. Un ordre de priorité a été établi dans [Creac'h et al., 2000]. Les contraintes de sécurité et de lisibilité sont les contraintes les plus fortes et doivent être absolument respectées.

2.2.3 Généralisation des isobathes

Les isobathes sont construites à partir des sondes extraites de la BDBS. Lorsque la construction se fait automatiquement, toutes les sondes de la zone à cartographier sont utilisées. Lorsque les isobathes sont construites manuellement, la quantité de données est trop importantes et seules les sondes les plus caractéristiques sont prises en compte. Une triangulation de Delaunay est d'abord effectuée sur l'ensemble des sondes. Pour construire les isobathes à une profondeur donnée, le cartographe calcule les points d'intersection entre les segments des triangles et un plan horizontal correspondant à la profondeur demandée. Cela nous fournit une liste de points de même profondeur. Les isobathes sont alors construites en parcourant les triangles et en reliant les points des triangles adjacents [Creach et Le Gac, 1995]. Les iso-

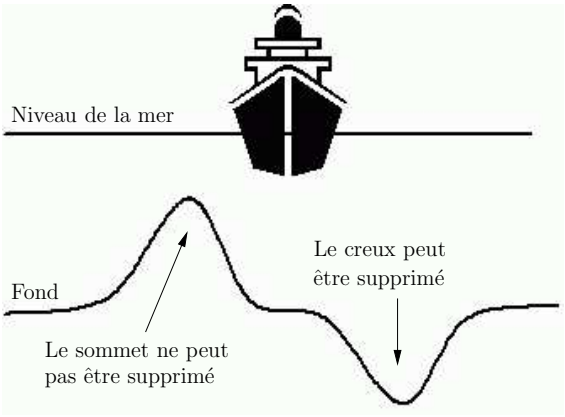

Fig. *2.3* — Illustration de la contrainte de sécurité.

bathes sont donc définies à partir de points interpolés. Etant donné la quantité de sondes relevées, l'erreur d'interpolation est relativement faible.

Les isobathes sont ensuite compressées afin de réduire le volume de données [Saux et Daniel, 1999]. Le principe est de rechercher la courbe B-spline définie avec le moins de points de contrôle approchant une polyligne à une tolérance donnée. Cette tolérance est relativement petite pour que la compression ne soit pas visible. La recherche du nombre de points de contrôle se fait par dichotomie. L'erreur entre la courbe B-spline et la polyligne est mesurée en calculant une approximation polygonale de la courbe par insertion de nœuds et en mesurant la distance de Hausdorff entre les deux polylignes. La solution est valide si cette erreur est inférieure à la tolérance donnée. Le processus dichotomique consiste à rechercher la courbe valide définie avec le moins de points contrôle.

Enfin, elles sont généralisées de sorte que les courbes affichées soient cohérentes et lisibles (figure 2.4).

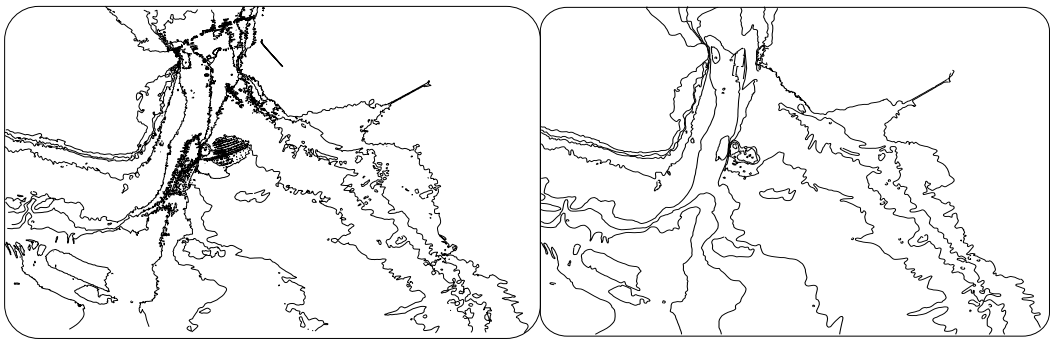

Fig. *2.4* — Isobathes avant (à gauche) et après généralisation manuelle (données fournies par le SHOM).

L'intérêt de modéliser des isobathes avec des courbes B-splines a été montré dans [Saux, 1999]. Les isobathes sont des courbes "lisses". Les courbes B-splines sont des courbes

paramétriques continues sur lesquelles nous disposons d'un contrôle local. Elles sont donc particulièrement bien adaptées pour la visualisation des isobathes et pour les opérations de changement d'échelle puisque, contrairement aux lignes polygonales où les isobathes peuvent être représentées par des lignes brisées, la courbe reste toujours lisse lorsque l'on augmente l'échelle de la carte. Ceci constitue un besoin important dans le cadre de systèmes embarqués où des agrandissements peuvent être demandés par l'opérateur travaillant sur son écran.

La généralisation des isobathes doit se faire en fonction des contraintes citées au paragraphe 2.2.2. Les différents opérateurs applicables sont :

- le lissage qui doit conserver les points caractéristiques de la courbe ;
- le déplacement de tout ou partie de la courbe ;
- la caricature visant à simplifier les détails et à amplifier les formes remarquables de la courbe ;
- l'agrégation consistant à définir une nouvelle isobathe enveloppant deux ou plusieurs courbes de même profondeur ;
- la suppression de tout ou partie d'une courbe.

Dans la suite de ce chapitre, les courbes B-splines sont construites en approchant des listes de points par lissage et par compression [Saux, 1999]. Elles sont construites avec une précision de 0,2 mm correspondant à l'épaisseur du trait de crayon pour que la compression ne soit pas visible à l'œil. Les isobathes étant des courbes de niveau, elles ne peuvent pas se couper par définition. Il y a donc très peu d'intersections franches. Néanmoins, il peut y avoir des intersections ou des auto-intersections réelles (figure 2.5). Elles sont dues à des problèmes numériques liés aux choix de paramétrisation et de vecteurs de nœuds ou au nombre de points de contrôle. Il peut également y avoir des tangences ou des recouvrements. En plus, les courbes étant définies avec beaucoup de points (jusqu'à 4000), la compression des courbes peut poser des problèmes numériques dus à un mauvais conditionnement.

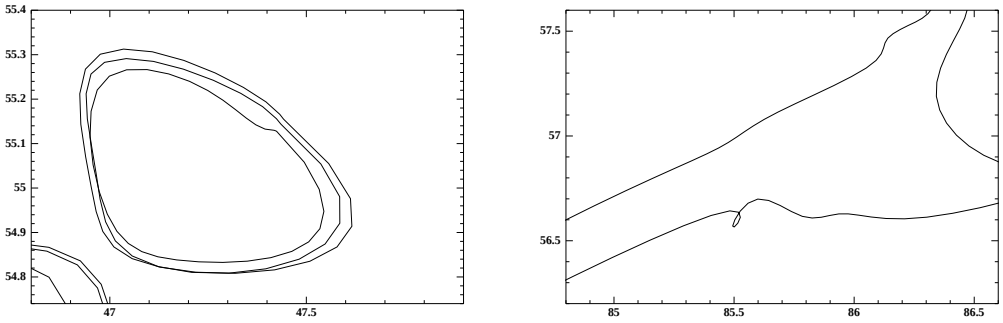

Fig. *2.5* — Intersections réelles d'isobathes : recouvrement de courbes et auto-intersection.

Enfin, la majeure partie des intersections existantes sont des intersections visuelles. Elles correspondent aux cas où, les courbes étant trop rapprochées, la contrainte de lisibilité n'est pas vérifiée. Sur la carte, la distance de lisibilité correspond à l'épaisseur du trait de crayon et est de 0,2 mm. Lorsque la distance entre deux courbes est inférieure, la contrainte n'est

pas respectée et il y a une intersection de visualisation. Par exemple, dans les zones de forte pente, il est possible que les courbes soient très rapprochées (figure 2.5 à gauche).

En fonction de l'échelle de la carte, les informations à représenter ne sont pas les mêmes. Plus l'échelle est petite, plus l'information est synthétisée. La représentation des mêmes données cartographiques à des échelles différentes donne lieu à des conflits et des choix de généralisation différents. Lors de la construction d'une carte, la correction des conflits se fait donc en modifiant les données de la carte et non en modifiant les données initiales de la base de données bathymétrique. Par conséquent, nous n'avons pas accès aux données initiales et la détection et la correction des conflits se fait sur les courbes B-splines représentant les isobathes.

2.3 Méthode de détection des intersections

2.3.1 Principe général

La méthode de détection doit être capable de traiter tous les types d'intersection mentionnés précédemment. Il faut donc une méthode robuste permettant de traiter aussi bien les intersections visuelles que réelles. Les méthodes géométriques sont les mieux adaptées car elles détectent tous ces conflits sans distinction. La méthode doit en plus être appliquée à un grand nombre de données. La méthode doit donc être également rapide. Etant donnée la quantité de courbes, il n'est pas possible de tester les intersections entre les courbes deux par deux en appliquant directement les méthodes du chapitre 1.

La détection se fait en deux phases : une première phase, rapide, dans laquelle nous cherchons à minimiser les calculs et dont l'objectif est de localiser les zones où il y a des intersections potentielles et une deuxième phase dans laquelle nous calculons les intersections avec des méthodes plus précises [Guilbert et al., 2003]. Ce principe est comparable aux techniques utilisées en animation pour la détection de collisions où le nombre de données à traiter est important. La première étape est une phase accélératrice grossière (*broad phase*). Il s'agit de déterminer rapidement les cas de non-intersection. Pour cela, nous avons recours à deux types de stratégies fondées sur :

- le découpage de l'espace : deux objets n'étant pas dans une même région ne peuvent pas être en intersection ;
- la topologie de l'espace : les objets sont regroupés en fonction de leur position les uns par rapport aux autres.

Ces deux stratégies font appel à différentes structures spatiales [Samet, 1990]. Dans la deuxième stratégie, la structure (R-tree, R^+-tree) est construite en fonction de la distance ou des recouvrements entre les objets. Ainsi, les cellules à un niveau sont formées de polygones englobant une ou plusieurs courbes. Une cellule mère est construite en calculant un polygone englobant toutes ses cellules filles. Les cellules sont donc construites par regroupement des englobants des courbes ou des segments de courbes alors que dans les méthodes de découpage

de l'espace, les courbes sont segmentées pour être réparties dans les différentes cellules. La deuxième approche convient mieux à notre problème de détection des intersections où le nombre de courbes est important rendant difficile et coûteux de définir un critère topologique permettant une classification efficace des courbes et une répartition homogène dans les cellules.

Le découpage spatial peut être absolu (quadtree) ou adapté à l'environnement (BSP, k-d tree). Les méthodes de partitionnement sont hiérarchiques. Cela permet d'adapter la taille des cellules à la densité d'information : lorsqu'une cellule est trop grande ou contient trop d'informations, elle est divisée en plusieurs cellules filles. Comme nous avons vu dans le chapitre précédent, la division des cellules doit se faire avec des méthodes rapides et robustes. Pour cela, la structure la plus intéressante est le quadtree : les calculs sont très rapides puisque les cellules sont construites en divisant la cellule mère en quatre cellules de même taille et la répartition des courbes revient à comparer la position des courbes par rapport à des boîtes.

Le principe de la décomposition est de répartir les courbes dans différentes cellules. Les courbes ne peuvent alors se couper que si elles sont situées dans la même cellule. Le partitionnement est également intéressant pour détecter les auto-intersections. Si deux segments distincts d'une courbe sont dans la même cellule, ce cas est traité comme une intersection. Le critère 1 de non auto-intersection n'est appliqué que sur chaque segment et non sur les courbes complètes. Un autre intérêt du quadtree est que, si des modifications sont effectuées comme l'insertion, la suppression ou le déplacement d'une courbe, il peut être mis à jour rapidement. Par exemple, l'insertion se fait en plaçant la nouvelle courbe à la racine puis en la segmentant dans les cellules filles jusqu'à arriver aux feuilles. Si nécessaire, de nouvelles feuilles sont créées.

La deuxième phase est une phase plus précise (*narrow phase*) dans laquelle sont calculées les intersections. Pour cela, des englobants relativement fins sont utilisés. Lorsque la cellule contient deux courbes, les méthodes du premier chapitre peuvent être utilisées. S'il y a plus de courbes, les tests d'intersection sont appliquées aux courbes deux à deux. A chaque fois, les courbes sont segmentées. Pour limiter les calculs, nous définissons des enveloppes fines pour chaque courbe. Les enveloppes représentent des approximations des courbes à une précision donnée. Les intersections sont alors définies comme les intersections entre les enveloppes.

La détection des intersections se fait donc en deux étapes. Nous présentons d'abord la méthode mise en place pour construire le quadtree et répartir les courbes dans les cellules puis, nous expliquons comment sont approchées les courbes et sont calculées les intersections.

2.3.2 Décomposition du plan

2.3.2.1 Hiérarchisation quadtree

Nous considérons le plan contenant toutes les courbes B-splines comme la racine du quadtree. A chaque étape de la décomposition, nous divisons les cellules en quatre cellules filles

suivant le critère suivant : si une cellule contient au plus une courbe, il ne peut pas y avoir d'intersection et la segmentation est arrêtée. Sinon, il y a intersection potentielle et la cellule est divisée (figure 2.6).

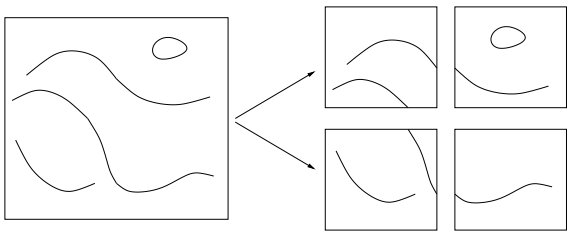

Fig. *2.6* — Division d'une cellule et répartition des courbes.

Deux segments situés dans deux cellules voisines peuvent être en intersection s'ils sont à une distance inférieure à ϵ_{vis}. De ce point de vue, un segment est en intersection potentielle avec les segments de la même cellule et avec ceux des cellules voisines. Pour simplifier notre problème, nous définissons l'appartenance d'un point ou d'un segment à une cellule à ϵ_{vis} près. C'est-à-dire, nous considérons qu'un point ou un segment appartiennent à une cellule s'ils sont à une distance inférieure à ϵ_{vis} du bord de la cellule. Ainsi, deux segments de courbes en intersection sont toujours dans la même cellule et il n'est pas nécessaire de détecter les conflits entre une courbe et les courbes des cellules voisines. Deux cellules voisines ont donc une frontière commune et les segments situés dans cette bande appartiennent aux deux cellules. Par conséquent, les conflits apparaissant dans ces zones sont détectés deux fois. Pour réduire le nombre d'intersections détectées en double, nous tenons compte du côté où se situe un point par rapport à la cellule. Par exemple, si le point se trouve au dessus ou à gauche de la cellule, l'appartenance à cette cellule n'est pas définie à une tolérance près. Par contre, s'il se trouve à droite ou en dessous, le point appartient à la cellule s'il est à ϵ_{vis} du bord (figure 2.7). Cela permet de limiter la largeur de la bande commune aux deux cellules.

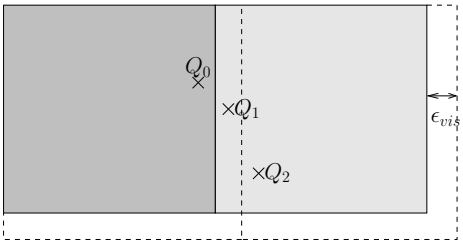

Fig. *2.7* — Appartenance d'un point à une cellule : le point Q_0 appartient à la cellule de gauche. Q_1 appartient aux deux cellules. Q_2 est dans la cellule de droite.

Le partitionnement du plan est également arrêté si :

– une taille minimale de cellule est atteinte. Plus cette taille minimale est petite et plus la localisation du conflit est précise ;

– les segments de courbe sont définis avec k points de contrôle car un segment ne peut pas être défini avec moins de points (voir propriété 3) du préambule.

Nous donnons sur la figure 2.8 un exemple de structure quadtree obtenu pour la carte de la figure 2.4. Il apparaît clairement que la décomposition est liée à la densité de courbes dans une région. C'est dans les zones où il y a le plus de courbes qu'il y a le plus de cellules car le risque d'intersection est plus grand.

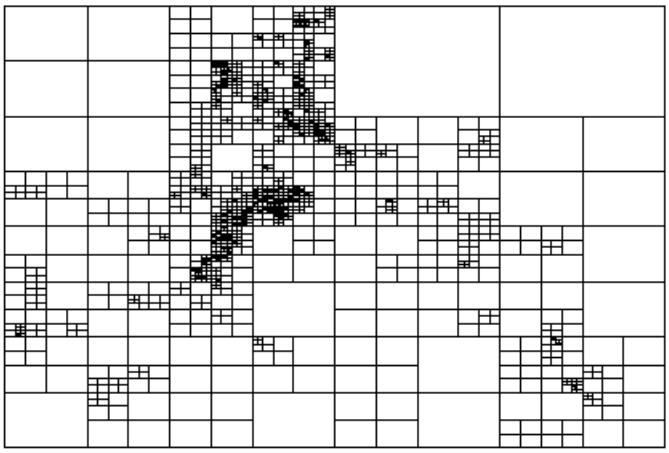

Fig. *2.8* — Exemple de structure quadtree d'une carte.

2.3.2.2 Segmentation des courbes

Lorsqu'une cellule est divisée, les segments de courbe de la cellule sont également répartis dans les cellules filles. Dans [Brunet et al., 1993], les auteurs utilisent la méthode d'élagage (paragraphe 1.3.2.3) pour couper les courbes. La largeur de la bande sert alors de critère d'arrêt pour le partitionnement. Cette méthode ne nous convient pas pour deux raisons. D'abord, la méthode d'élagage peut poser des problèmes de robustesse si la courbe est tangente au bord de la cellule. Ensuite et surtout, nous cherchons à avoir une méthode rapide en limitant le nombre d'opérations.

Pour ces raisons, la méthode fondée sur l'étude du polygone de contrôle présentée au paragraphe 1.3.2.4 est intéressante à deux niveaux. D'abord, nous effectuons peu de calculs puisque la méthode n'insère pas de nouveaux points sur la courbe. Ensuite, les courbes originelles ne sont pas modifiées. Il suffit de travailler avec des listes d'indices délimitant les segments de courbes. Cela permet d'économiser de la place mémoire tout en gardant un accès à l'ensemble des courbes.

Cette méthode travaille sur des segments délimités à partir des points de contrôle et des nœuds déjà existants. Pour identifier un segment, nous n'avons donc besoin que des indices du premier et du dernier nœuds de l'intervalle paramétrique définissant le segment et d'un lien avec la courbe B-spline à laquelle il appartient. En terme de mémoire, un segment est donc défini par deux entiers et un pointeur (un entier long) vers la courbe. En comparaison,

la méthode d'élagage nécessite l'insertion de $2k$ points de contrôle à deux coordonnées et $2k$ nœuds soient $6k$ réels lorsque l'on coupe une courbe. De plus, l'étude du polygone de contrôle est intéressante en terme de coût de calcul puisqu'elle est fondée sur une comparaison des coordonnées des points avec les coordonnées de la cellule.

Comme les segments sont définis uniquement à partir des points de contrôle existants, il y a intersection potentielle entre un segment de courbe et une cellule si k points de contrôle consécutifs du segment ne sont pas du même côté de la cellule. Cela ne signifie pas pour autant que le segment est dans la cellule. Par exemple, nous présentons figure 2.9 le cas particulier où le segment de courbe défini par les quatre points de contrôle appartient à la cellule parce qu'ils ne sont pas tous dans le même demi-plan au-dessus ou à droite de la cellule. Ce cas est relativement rare car les segments sont définis avec un grand nombre de points et seuls les points extrêmes sont à l'extérieur de la cellule. Cependant, un segment tel que celui-ci doit être pris en compte et appartient à quatre cellules.

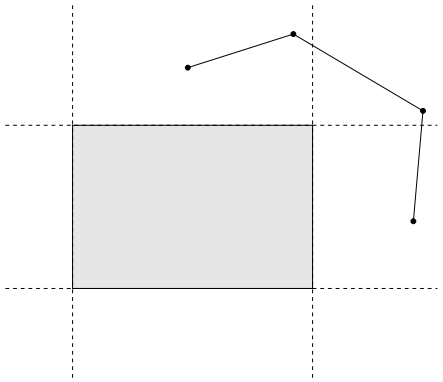

Fig. *2.9* — Le segment de courbe appartient à la cellule. Il y a intersection potentielle entre la courbe et la cellule car les points de contrôle ne sont pas tous du même côté.

Ainsi, si nous disposons d'un ensemble de $N+1$ courbes B-splines f^i, $0 \leq i \leq N$, nous notons $I \subset \{0, \ldots, N\}$ l'ensemble des indices i des f^i passant dans une cellule c. A chaque indice i de I, nous associons une liste de couples d'indices (i_{min}, i_{max}). Ce couple désigne le plus petit intervalle paramétrique $[t_{i_{min}}, t_{i_{max}}]$ contenant l'intersection de f^i et de la cellule c. Cet intervalle est calculé par la méthode de pré-traitement présentée au paragraphe 1.3.2.4.

Nous illustrons le principe de la méthode sur la figure 2.10. La courbe f^0 est une B-spline cubique définie sur l'intervalle $[t_0^0, t_{24}^0]$ (nous considérons que les nœuds extrêmes sont de multiplicité $k = 4$). Nous regardons les suites de k points de contrôle d'un même côté de la cellule. Ainsi, les quatre premiers points de contrôle Q_0^0 à Q_3^0 sont au-dessus de la cellule. L'intervalle $[t_3^0, t_4^0]$ est donc retiré de l'intervalle d'étude. De même, les points Q_8^0 à Q_{12}^0 sont à droite de la cellule et les points Q_{11}^0 à Q_{14}^0 sont sous la cellule. Nous pouvons retiré l'intervalle $[t_{11}^0, t_{15}^0]$. Enfin, les points de contrôle Q_{16}^0 à Q_{19}^0 sont en dessous de la cellule et les points Q_{17}^0 à Q_{20}^0 sont à gauche. L'intervalle $[t_{19}^0, t_{21}^0]$ est donc supprimé. Par conséquent, la restriction de f^0 à la cellule est constituée de deux segments de courbe définis sur les intervalles $[t_4^0, t_{11}^0]$

et $[t^0_{15}, t^0_{19}]$. Pour la cellule c de la figure 2.10, nous avons donc $0 \in I$ et 0 est lié à la liste $\{(4, 11), (15, 19)\}$.

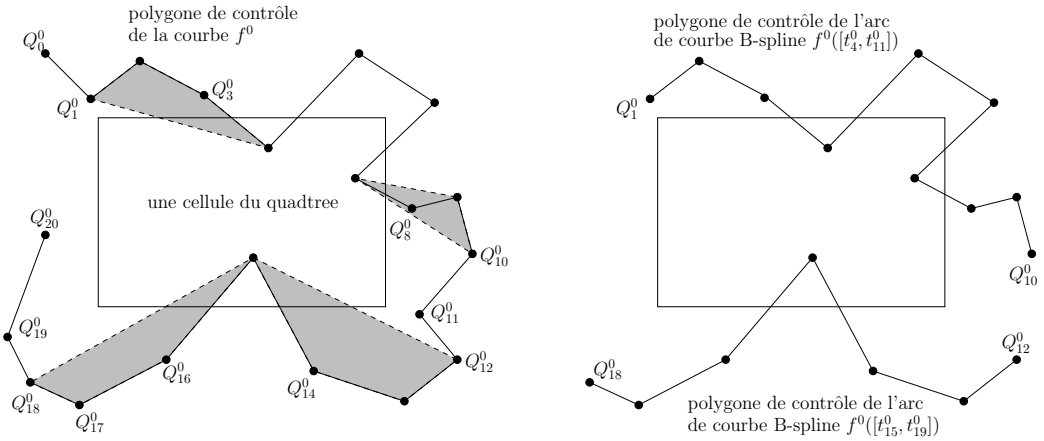

Fig. *2.10* — Réduction d'une courbe dans une cellule. A gauche, les enveloppes convexes grises représentent les extrémités des segments en intersection potentielle avec la cellule. A droite, polygones de contrôle et intervalles paramétriques des deux segments de courbe.

Notons que ce découpage spatial permet de détecter les auto-intersections puisqu'il n'est plus nécessaire d'appliquer le critère de non auto-intersection sur toute la courbe. Il suffit de l'appliquer sur chaque segment. Par exemple, sur la figure 2.10, nous appliquons le critère sur chacun des deux segments. S'il y a une auto-intersection entre les segments définis sur $[t^0_4, t^0_{11}]$ et $[t^0_{15}, t^0_{19}]$, elle est détectée avec la même méthode que pour les intersections.

Au début de la décomposition, la racine de l'arbre correspond à la carte complète avec toutes les courbes. Nous avons $I = \{0, \ldots, N\}$ contenant les indices des courbes f^i et chaque i est lié au couple $(k-1, m^i+1)$ avec m^i+1 le nombre de points de contrôle de f^i. Ensuite, nous appliquons la méthode de partitionnement présentée en 2.3.2.1 couplée avec cette méthode de découpage.

A la fin du processus, nous avons défini un quadtree où chaque cellule est vide ou contient une liste de segments de courbes. Lorsqu'une courbe est segmentée dans plusieurs cellules, certaines parties de la courbe sont communes à plusieurs cellules étant donné que l'appartenance est définie à une tolérance ϵ_{vis} près et que la répartition se fait à partir du polygone de contrôle. S'il y a intersection sur ces parties, l'intersection sera détectée plusieurs fois puisqu'elle est présente dans plusieurs cellules. Sur la figure 2.11, nous voyons que les boites minmax des segments de courbe (en traits forts) se recoupent au bord des cellules (en traits pointillés). De ce fait, la méthode de décomposition est robuste puisque les segments de courbe sont toujours plus larges. Une intersection peut être localisée dans plusieurs cellules mais elle ne peut pas être omise.

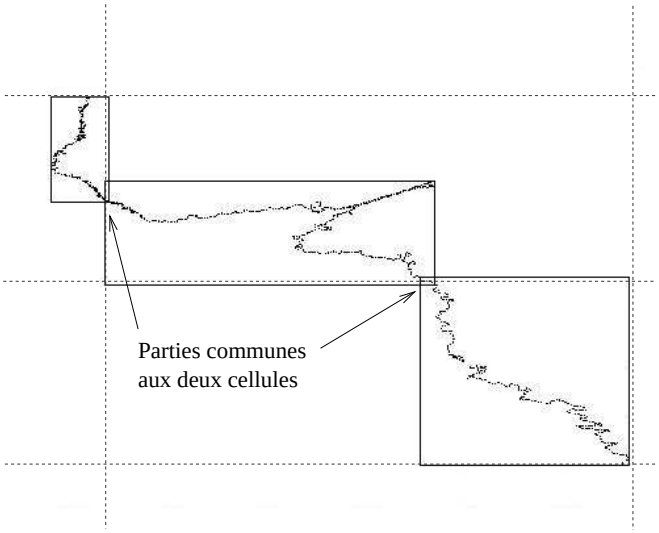

Fig. *2.11* — Répartition d'une courbe dans plusieurs cellules : certains points de contrôle appartiennent à plusieurs segments (extrait de [Guilbert, 2002]).

2.3.3 Détection des intersections par des méthodes de subdivision

L'objectif de la méthode présentée dans ce paragraphe est de calculer les auto-intersections et les intersections entre des segments de courbe dans une même cellule. Une méthode pour réduire les calculs est de définir des volumes englobants hiérarchiques [Gottschalk et al., 1996, Krishnan et al., 1998]. Une suite d'englobants est calculée pour chaque objet. Si deux englobants se recouvrent, la détection des intersections se fait alors en utilisant les englobants au niveau hiérarchique suivant. Nous n'utilisons pas d'englobants hiérarchiques car les risques d'intersection sont relativement grands avec des englobants trop grossiers et ils ne permettent pas de conclure rapidement à une non intersection ni de calculer cette intersection. Pour ce faire, nous calculons une enveloppe fine contenant la courbe. Cette enveloppe est définie à partir du polygone de contrôle. Elle doit être suffisamment fine pour constituer une approximation de la courbe. Nous subdivisons donc le polygone de contrôle pour réduire la largeur de l'enveloppe jusqu'à ce que la précision voulue soit atteinte. Le schéma de subdivision appliqué dépend des caractéristiques de la courbe [Sabin, 2000].

Dans [Neagu et Lacolle, 1999], les auteurs présentent une méthode de détection des intersections entre courbes de Bézier convexes où des enveloppes sont définies à l'aide de schémas de subdivision. A partir de ces enveloppes, une condition nécessaire et suffisante d'intersection est donnée. Cependant, le critère établi ne concerne que les intersections franches. De plus, son application nécessite d'exprimer les courbes B-splines sous forme de courbes de Bézier convexes raccordées par morceaux.

Dans le cas des courbes B-splines quelconques, le schéma de subdivision dépend du vecteur de nœuds et se fait en insérant de nouveaux nœuds sur ce vecteur. Dans le cas où le vecteur de nœuds est uniforme, l'insertion se fait de façon régulière et des simplifications peuvent être

faites réduisant les calculs. Les deux cas sont donc traités séparemment. Nous présentons d'abord l'algorithme utilisé pour des courbes B-splines quelconques puis nous traitons le cas des courbes B-splines uniformes. Enfin, nous expliquons comment sont calculées les auto-intersections et les intersections.

2.3.3.1 Approximation des courbes B-splines non uniformes

Construction d'une enveloppe fine Nous présentons ici une méthode permettant de calculer une enveloppe contenant une courbe B-spline. L'enveloppe définie de cette façon est plus fine que l'enveloppe convexe du polygone de contrôle.

Soit f une fonction B-spline. Nous notons ζ_i les abscisses de Greville de la fonction f définies par $\zeta_i = \frac{1}{k-1}\sum_{j=1}^{k-1} t_{i+j}$. Soit $l(\zeta)$ la courbe polygonale interpolant les points de contrôle Q_i aux abscisses de Greville (c'est-à-dire, $l(\zeta_i) = Q_i$). Nous notons également $\Delta_2 Q_i = \frac{Q_{i+1}-Q_i}{\zeta_{i+1}-\zeta_i} - \frac{Q_i-Q_{i-1}}{\zeta_i-\zeta_{i-1}}$.

Pour un intervalle $[\zeta_i, \zeta_{i+1}]$, nous notons $\overline{i}$ et $\underline{i}$ les indices de la première et de la dernière fonction de base B-spline non nulle sur cet intervalle. Enfin, nous définissons les fonctions β_{ij} par

$$\beta_{ij} = \begin{cases} \sum_{h=j}^{\overline{i}} (\zeta_h - \zeta_j) N_h^k & j > i \\ \sum_{h=\underline{i}}^{j} (\zeta_j - \zeta_h) N_h^k & j \leq i \end{cases} \tag{2.1}$$

Ces fonctions sont positives et convexes par morceaux [Lutterkort et Peters, 1999].

Lutterkort a montré dans [Lutterkort et Peters, 1999] que sur un intervalle $[\zeta_i, \zeta_{i+1}]$, la différence entre $l(\zeta)$ et $f(\zeta)$ est donnée par

$$f(\zeta) - l(\zeta) = \sum_{j=\underline{i}}^{\overline{i}} \Delta_2 Q_j \beta_{ij}(\zeta) \tag{2.2}$$

A partir de l'équation (2.2), nous pouvons déduire une enveloppe fine contenant la courbe. Cette enveloppe est définie par les points $l(\zeta_i)$ qui sont les points de contrôle et les points $f(\zeta_i)$ qui sont des points de la courbe (figure 2.12).

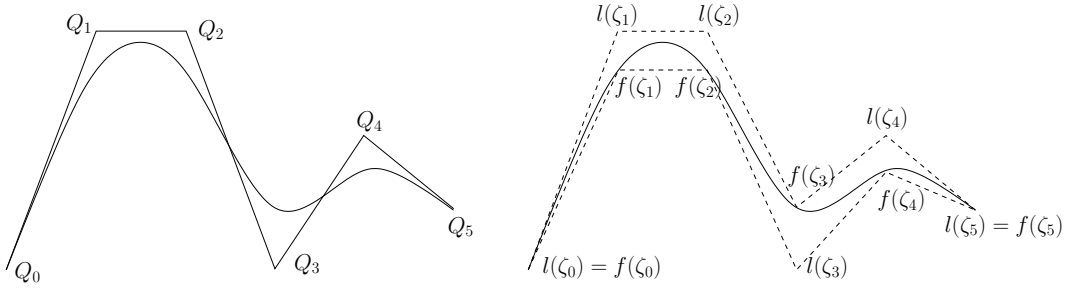

Fig. *2.12* — Enveloppe fine d'une courbe B-spline.

(2.2) permet de calculer rapidement les $f(\zeta_i)$ et nous donne la distance entre les $f(\zeta_i)$ et les $l(\zeta_i)$. La largeur de l'enveloppe est majorée par la plus grande distance $\|f(\zeta_i)-l(\zeta_i)\|$. Nous notons ϵ_{num} la demi-épaisseur de l'enveloppe. Celle-ci constitue une approximation à ϵ_{num} près de la courbe. Nous dirons qu'il y a intersection entre deux courbes si la distance entre leurs enveloppes est inférieure à ϵ_{vis} (figure 2.13). Nous voyons donc que plus les enveloppes sont larges, plus les approximations sont mauvaises et plus il y aura d'intersections.

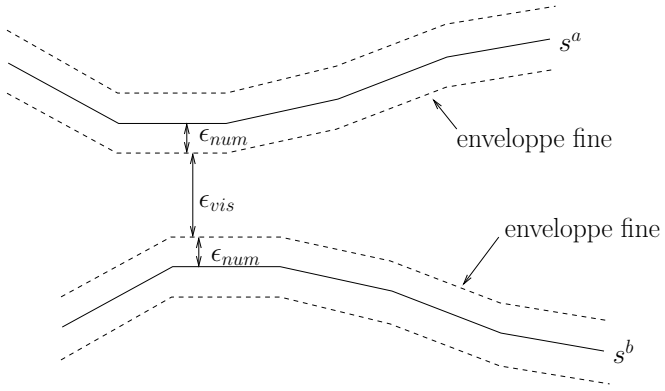

Fig. *2.13* — Distance entre deux segments.

Pour calculer une enveloppe d'une largeur donnée, nous utilisons un schéma de subdivision pour que le polygone de contrôle soit plus près de la courbe : tant que la largeur voulue n'est pas atteinte, le polygone de contrôle est subdivisé et l'enveloppe est construite à partir du nouveau polygone.

Schéma de subdivision L'approximation des courbes B-splines se fait en insérant de nouveaux nœuds dans le vecteur nodal. D'une façon générale, à chaque fois qu'un nouveau nœud est inséré, nous remplaçons $k-1$ points de contrôle par k nouveaux points de contrôle (voir résultat 1). Le nouveau polygone de contrôle est alors plus proche de la courbe qu'il définit. Si nous insérons un nœud au centre de chaque intervalle paramétrique $[t_i, t_{i+1}]$, nous passons de m à $2m+1$ points de contrôle et nous avons un polygone plus proche de la courbe. Cela revient à appliquer le schéma suivant :

$$\begin{cases} t_{2i}^{(n+1)} & = t_i^{(n)} \\ t_{2i+1}^{(n+1)} & = \frac{1}{2}(t_i^{(n)} + t_{i+1}^{(n)}) \end{cases} \tag{2.3}$$

où n désigne la $n^{\text{ème}}$ étape de subdivision. Ce schéma peut être appliqué plusieurs fois (figure 2.14).

La suite de polygones $(Q_i^{(n)})_{i=0}^{m^{(n)}}$ converge uniformément vers la fonction B-spline cubique définie par le polygone de contrôle initial $(Q_i^{(0)})_{i=0}^{m^{(0)}}$ [Farin, 1992]. Le polygone de contrôle à l'étape n peut être considérée comme une approximation de la courbe à une précision donnée par $\max_{i=0}^{m^{(n)}} \|Q_{i+1}^{(n)} - Q_i^{(n)}\|$. Etant donné que le polygone est de plus en plus proche de la

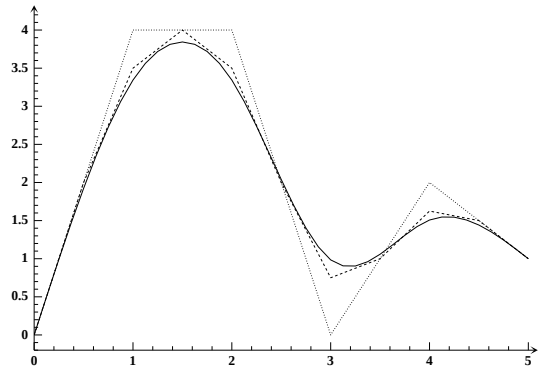

Fig. *2.14* — Subdivision d'un polygone de contrôle : polygone initial (en pointillés) et polygones obtenus après une (en tirets) et trois subdivisions (en trait plein).

courbe, l'enveloppe construite à partir de ce polygone est de plus en plus fine et a pour épaisseur maximale $\max \|f(\zeta_i^{(n)}) - Q_i^{(n)}\|$.

Calcul d'une ligne médiane Le calcul de l'enveloppe puis des distances entre deux enveloppes est une opération complexe. Pour limiter les calculs, nous définissons une courbe polygonale $s(\zeta)$ médiane à l'enveloppe (*midpath*). s relie les points $s(\zeta_i)$ définis par [Peters et Wu, 2001]

$$s(\zeta_i) = \frac{1}{2}(f(\zeta_i) + l(\zeta_i)) \tag{2.4}$$

Cette ligne est entièrement contenue dans l'enveloppe et constitue une approximation de la courbe f à ϵ_{num} près (figure 2.15). Pour savoir s'il y a intersection entre les courbes B-splines, nous calculerons les intersections entre les lignes médianes.

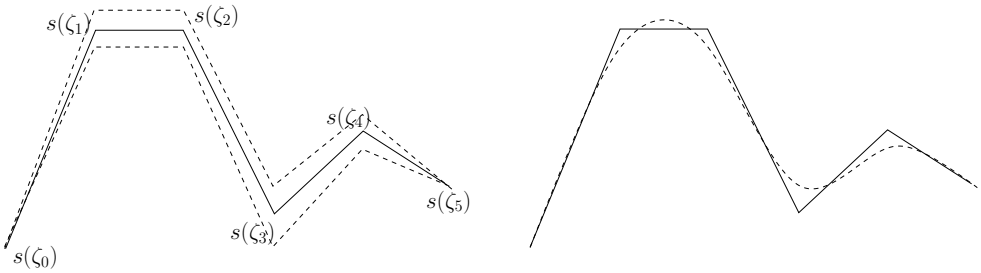

Fig. *2.15* — Ligne médiane d'une enveloppe.

2.3.3.2 Approximation des courbes B-splines uniformes

Dans le cas où le vecteur nodal est uniforme, les expressions données dans le paragraphe précédent sont simplifiées puisqu'elles sont indépendantes de l'intervale paramétrique considéré. Dans ce paragraphe, les schémas sont donnés pour des courbes B-splines cubiques uniformes ($k = 4$) mais la méthode peut facilement être étendue à d'autres degrés.

Schéma de subdivision Ainsi, le schéma de subdivision est défini uniquement à partir des points de contrôle. Soit une courbe B-spline uniforme $f(t) = \sum_{i\in\mathbb{Z}} Q_i^{(n)} N_i^k(t)$ où n indique la $n^{\text{ème}}$ étape de subdivision, nous cherchons à écrire f à l'étape $n+1$ sous la forme

$$f(t) = \sum_{i\in\mathbb{Z}} Q_i^{(n+1)} N_i^k(2t) \tag{2.5}$$

Le nouveau polygone donné par les $Q_i^{(n+1)}$ est un polygone de contrôle de f défini avec deux fois plus de points et est plus proche de la courbe. Le calcul des $Q^{(n+1)}$ dépend de l'ordre k. Dans le cas des splines cubiques, nous obtenons un schéma de la forme

$$\left\{ \begin{array}{lcl} Q_{2i}^{(n+1)} & = & \frac{1}{2}Q_i^{(n)} + \frac{1}{2}Q_{i+1}^{(n)} \\ Q_{2i+1}^{(n+1)} & = & \frac{1}{8}Q_i^{(n)} + \frac{3}{4}Q_{i+1}^{(n)} + \frac{1}{8}Q_{i+2}^{(n)} \end{array} \right. \tag{2.6}$$

Cette suite de polygones converge uniformément vers la fonction B-spline cubique uniforme définie par le polygone de contrôle initial et sa vitesse de convergence est connue :

$$\max_{i\in\mathbb{Z}} \|Q_{i+1}^{(n+1)} - Q_i^{(n+1)}\| \leq \frac{1}{2} \max_{i\in\mathbb{Z}} \|Q_{i+1}^{(n)} - Q_i^{(n)}\| \tag{2.7}$$

Construction d'une enveloppe fine Dans le cas des splines uniformes, les abscisses de Greville sont définies par $\zeta_i = i + \frac{k}{2}$ et nous avons $\Delta_2 Q_i = Q_{i-1} - 2Q_i + Q_{i+1}$. L'équation (2.2) nous donne l'inégalité suivante [Lutterkort et Peters, 2000] :

$$\|f - l\| \leq \frac{k}{24} \max_{i-\frac{k-1}{2}+1\leq j\leq j+\frac{k-1}{2}-1} \|\Delta_2 Q_j\|_2 \tag{2.8}$$

A partir de l'équation (2.8), il est possible de construire une enveloppe contenant la courbe. Dans le cas des courbes cubiques, nous avons l'égalité (2.9) aux abscisses de Greville. Nous pouvons donc, comme pour (2.2), calculer la largeur de l'enveloppe et les positions des points $f(\zeta_i)$.

$$f(\zeta_i) = l(\zeta_i) + \frac{1}{6}\Delta_2 Q_i \tag{2.9}$$

Si nous subdivisons le polygone de contrôle avec le schéma (2.6), nous pouvons majorer la largeur de l'enveloppe du polygone subdivisé à partir de l'enveloppe du polygone initial : à partir de (2.7) et (2.9), nous majorons $\Delta_2 Q_i^{(n+1)}$ en fonction de $\Delta_2 Q_i^{(n)}$. Le nouveau majorant est donné par (2.10) et est meilleur que (2.7).

$$\max_i \|\Delta_2 Q_i^{(n+1)}\| = \frac{1}{4} \max_i \|\Delta_2 Q_i^{(n)}\| \tag{2.10}$$

Par conséquent, il est possible de calculer *a priori* le nombre de subdivisions n pour amener le polygone de contrôle à une distance ϵ de la courbe.

$$n = \log_4 \frac{\max_i \|\Delta_2 Q_i\|_2}{6\epsilon} \tag{2.11}$$

Nous remarquons que le calcul de l'enveloppe se fait sans manipuler le vecteur de nœuds. En plus, le nombre de subdivisions est connu dès le départ. Il n'est pas nécessaire de calculer la largeur de l'enveloppe à chaque étape. L'approximation des courbes B-splines uniformes se fait donc beaucoup plus rapidement. Ensuite, le calcul de la courbe médiane à l'enveloppe s se fait de la même façon dans les deux cas.

2.3.3.3 Calcul des intersections

L'objectif de la méthode est de détecter et localiser les intersections pour les corriger dans un second temps. Comme les corrections se feront essentiellement par le déplacement de points de contrôle, nous n'avons pas besoin de connaître avec précision le paramètre ou l'intervalle paramétrique où a lieu cette intersection. Nous avons surtout besoin de connaître sur quel intervalle $[t_i, t_{i+1}]$ a lieu l'intersection pour identifier les points de contrôle à déplacer. Cependant, les calculs doivent être faits avec une précision suffisante pour que les solutions soient valides. Si nous choisissons un ϵ_{num} trop grand, nous obtiendrons trop d'intersections.

Calcul des intersections Pour localiser les intersections dans une cellule, nous calculons d'abord les enveloppes fines de tous les segments de courbe à une demi-largeur ϵ_{num}. La ligne médiane s de chaque enveloppe est ensuite calculée. Pour deux courbes s^a et s^b, le calcul des intersections se fait en calculant la distance entre tous les segments de s^a avec tous les segments de s^b. Pour optimiser les calculs, nous pouvons utiliser au préalable le même principe que pour la méthode présentée en 1.3.2.4. Pour chaque courbe, nous excluons tous les segments qui sont à une distance supérieure à $\epsilon = \epsilon_{vis} + 2\epsilon_{num}$ de la boite minmax de la deuxième courbe. En appliquant ce principe à chaque courbe successivement, nous éliminons une grande partie des points.

Ensuite, nous calculons les distances entre les segments restants. Nous considérons qu'il y a intersection entre deux segments si la distance est inférieure à ϵ (figure 2.13). La distance entre deux segments $[s^a(\zeta_p^a)s^a(\zeta_{p+1}^a)]$ et $[s^b(\zeta_q^b)s^b(\zeta_{q+1}^b)]$ est définie comme étant la plus petite distance entre les points des segments. Une intersection est donnée par les intervalles paramétriques de chaque segment. C'est un couple de la forme $([\zeta_p^a, \zeta_{p+1}^a], [\zeta_q^b, \zeta_{q+1}^b])$. Lorsque plusieurs segments consécutifs sont en intersection avec un autre segment, nous pouvons regrouper les segments pour ne définir qu'une seule intersection. En regroupant les intersections consécutives, nous pouvons définir une intersection par un couple $([\zeta_{p_{min}}^a, \zeta_{p_{max}}^a], [\zeta_{q_{min}}^b, \zeta_{q_{max}}^b])$. Pour corriger les conflits, nous n'avons pas besoin de connaître les intervalles paramétriques à modifier avec une grande précision puisque les corrections seront effectuées en modifiant les points de contrôle. Par conséquent, nous cherchons ensuite les nœuds encadrant ces intervalles $t_{i_{min}}^a \leq \zeta_{p_{min}}^a < \zeta_{p_{max}}^a \leq t_{i_{max}}^a$ et $t_{j_{min}}^b \leq \zeta_{q_{min}}^b < \zeta_{q_{max}}^b \leq t_{j_{max}}^b$. Une intersection entre deux segments dans une cellule est donc repérée par une liste de couples d'indices $((i_{min}, i_{max}), (j_{min}, j_{max}))$ des nœuds définissant les parties en conflit.

Calcul des auto-intersections En plus de calculer les intersections, pour chaque courbe, il faut détecter les auto-intersections. Nous avons vu dans le paragraphe 2.3.2.2 qu'il n'était pas nécessaire d'appliquer le critère 1 de non auto-intersection sur toute la courbe mais seulement sur chaque segment. En effet, un conflit entre deux segments disjoints d'une même courbe est détecté comme une intersection et non comme une auto-intersection.

Comme nous disposons d'une approximation s de chaque segment de courbe, nous pouvons détecter les auto-intersections sur ces approximations. s admet une auto-intersection si elle revient en arrière sur elle-même (c'est une condition nécessaire mais non suffisante). En reprenant le critère 1, nous pouvons segmenter s en segments n'admettant pas d'auto-intersection. Nous pouvons appliqué le critère 2 entre les segments adjacents. Après application de ces deux critères, les seules auto-intersections possibles sont des intersections entre des segments disjoints. Nous nous ramenons alors à la méthode de calcul précédente.

Regroupement des intervalles A cet instant, nous avons calculé les intersections à l'intérieur de chaque cellule mais un même conflit peut être calculé plusieurs fois s'il se situe à l'intersection de deux cellules. De plus, pour les cas d'intersection visuelle ou de superposition de courbes, les segments en conflit peuvent être grands et l'intersection est alors répartie sur plusieurs cellules. Il est donc nécessaire de regrouper les segments répartis dans plusieurs cellules afin de ne définir qu'un seul conflit à corriger et de supprimer les conflits détectés plusieurs fois.

Une dernière étape de regroupement des intervalles doit donc être appliquée avant de corriger les conflits. Il s'agit de passer en revue la liste des intersections et des auto-intersections et de regrouper les conflits ayant un intervalle commun ou adjacent. Chaque intersection est repérée par les indices a et b des deux courbes f^a et f^b en conflit et par les indices $((i_{min}, i_{max}), (j_{min}, j_{max}))$ des nœuds délimitant les intervalles en conflit. En parcourant la liste des conflits, nous pouvons trouver plusieurs conflits entre f^a et f^b. Si les intervalles paramétriques se recouvrent alors cela signifie que nous avons en fait une seule intersection segmentée dans plusieurs cellules. Nous regroupons donc les intersections en une seule définie par le plus petit et le plus grand indices de chaque courbe. Après ce traitement, nous obtenons une nouvelle liste de conflits toujours définis de la même manière (les indices des courbes et les indices des segments en intersection) mais ces conflits ne tiennent plus compte du partitionnement de la carte. C'est à partir de cette liste que les intersections seront corrigées. Le regroupement est une opération très rapide puisqu'elle se fait uniquement en comparant les indices des segments en conflit.

2.3.4 Schéma général de la méthode

Le principe de la méthode est résumé par l'algorithme 1. Les parties en italique correspondent aux étapes d'approximation et de calcul des intersections présentées en 2.3.3.

Algorithme 1 *Détection des intersections dans un ensemble d'isobathes.*

```
Fonction détection
Données en entrée : cellule c et liste des segments contenus dans la cellule
      Si aucun critère d'arrêt n'est vérifié
            Division de c en quatre cellules
            Répartition des segments dans les cellules filles
            Pour chaque cellule fille f_i
                  Appel détection(f_i)
            Fin pour
      Sinon
            Si la cellule est vide
                  Fin détection
            Fin si
            Si la courbe contient un seul segment
                  Approximation du segment
                  Détection des auto-intersections
                  Ajout à la liste
            Sinon
                  Approximation des segments
                  Détection des intersections et des auto-intersections
                  Ajout à la liste
            Fin si
      Fin si
Données en sortie : liste des intersections dans la cellule
Fin détection

Début
Données en entrée : liste des isobathes
      Initialisation
          Construction de la cellule mère contenant toutes les isobathes
      Appel détection(cellule mère)
      Regroupement des conflits
Données en sortie : liste des conflits
Fin
```

2.4 Résultats

La méthode présentée a été testée sur plusieurs cartes fournies par le SHOM. Le programme a été écrit en C++. Les tests ont été faits sur un Athlon Thunderbird 1.1GHz sous Linux. Nous présentons les résultats obtenus pour les deux cartes de la figure 2.16. La figure de gauche est la même que la figure 2.4. Elle contient 900 courbes et 30 000 points. La fi-

gure de droite contient 2 300 courbes et 80 000 points. Ces cartes sont intéressantes car la morphologie des fonds est assez complexe, ce qui fait que nous avons une grande densité de courbes et que nous pouvons trouver toutes sortes de conflits. Notons que la méthode a été testée sur d'autres cartes et qu'elle a donné des résultats similaires mais avec des différences moins marquées que pour ceux présentés ici car la nature des fonds sur ces cartes était plus homogène. Pour toutes les cartes, la précision des données qui nous ont été fournies est de $\epsilon_{mod} = 10^{-3}$.

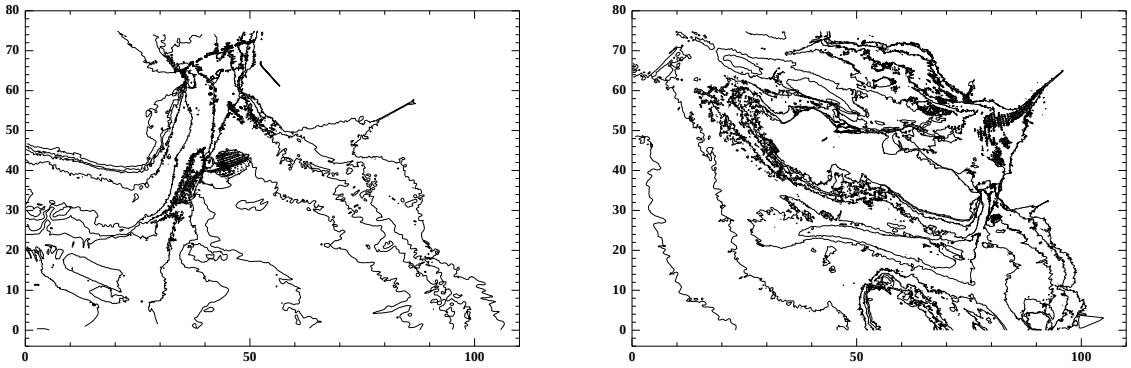

Fig. *2.16* — Jeux de données utilisés pour les tests : cartes n ° 7413 et 7404 fournies par le SHOM.

Les données fournies par le SHOM sont des listes de points représentant les isobathes avec deux indicateurs donnant la profondeur de l'isobathe et précisant si la courbe est ouverte ou fermée. Une première étape fut donc de construire les courbes B-splines par compression des données. La même méthode que celle présentée dans [Saux et Daniel, 1999] a été utilisée. C'est-à-dire, nous avons recherché par dichotomie à réduire le nombre de points de contrôle de la courbe. La paramétrisation utilisée est une paramétrisation centripète [Lee, 1989]. Cette paramétrisation donne de bons résultats avec un conditionnement faible et est beaucoup plus rapide que la paramétrisation de Hoschek améliorée. L'algorithme de construction des courbes est résumé ci-dessous par l'algorithme 2.

Pour chaque carte, nous avons construit un jeu de courbes B-splines avec un vecteur de nœuds de De Boor et un jeu de courbes B-splines avec un vecteur de nœuds uniforme afin de tester les méthodes avec les deux types de schémas de subdivision.

Dans chaque cas, nous avons calculé le nombre d'intersections en fonction de la profondeur du quadtree et de la précision ϵ_{num}. Cette précision est utilisée pour l'approximation des B-splines par des polylignes. Nous donnons le temps de calcul total ainsi que le pourcentage de temps passé à calculer les intersections avec la méthode présentée en 2.3.3. Ceci permet de connaître le temps passé au calcul des intersections par rapport au temps de hiérarchisation de la carte. Dans tous les exemples traités, nous avons pris $\epsilon_{vis} = 2.10^{-2}$ correspondant à l'épaisseur du trait sur les cartes marines.

Algorithme 2 *Construction des courbes B-splines par compression.*

```
Début
Données en entrée : polyligne l définie par n points
      Paramétrisation centripète de la courbe B-spline
      m_inf = 0, m_sup = n
      Tant que m_sup − m_inf > 1
            m = (m_inf + m_sup)/2
            Calcul de la B-spline f approchant l
            (utilisation d'une méthode des moindres carrés)
            Mesure de l'erreur entre f et l
            Si erreur>tolerance
                  m_inf = m
            Sinon
                  m_sup = m
            Fin si
      Fin tant que
Données en sortie : courbe B-spline approchant l
Fin
```

2.4.1 Courbes B-splines non uniformes

2.4.1.1 Résultats en fonction de la profondeur de l'arbre

Dans les tableaux 2.1 et 2.2, nous donnons les temps et le nombre d'intersections trouvées en fonction de la profondeur de l'arbre. La précision ϵ_{num} est fixée à 2.10^{-3}. Nous donnons dans la quatrième colonne le nombre total d'intersections détectées dans chaque cellule et dans la cinquième colonne le nombre d'intersections après regroupement des intervalles.

Nous voyons que le nombre d'intersections détectées augmente avec la profondeur. Cela vient du fait que les courbes sont segmentées en plus de morceaux. Nous constatons également que le pourcentage de temps passé à calculer les intersections diminue. En effet, comme les cellules sont plus petites, les segments de courbes sont plus petits et définis avec moins de points. L'approximation des courbes et le calcul des distances entre les segments prendra donc moins de temps et la part de temps passé à la décomposition du plan sera plus importante.

Les meilleurs temps sont obtenus pour des arbres de profondeur 5 ou 6 suivant la carte. Pour les arbres de profondeur inférieure, les segments de courbe sont plus grands et l'on passe beaucoup plus de temps à calculer les distances entre les courbes. Pour les arbres de profondeur supérieure, le temps de calcul augmente parce que les calculs sont redondants. En effet, un segment de courbe doit être défini avec au moins k points de contrôle. Si les cellules sont trop petites, un segment peut être défini dans plusieurs cellules et une même intersection pourra être calculée plusieurs fois. C'est pour cette raison que le nombre d'intersections

détectées augmente avec la profondeur.

Les intersections sont calculées à partir des approximations des courbes dans chaque cellule. En fonction de la profondeur, les segments de courbes sont plus ou moins grands. Les approximations ne sont donc pas toujours les mêmes. Par conséquent, nous obtenons de légères différences dans la définition des segments. Par exemple, à la profondeur 6, nous trouvons une intersection entre les segments de courbe $f^0([t^0_4, t^0_{33}])$ et $f^1([t^1_5, t^1_{39}])$ et à la profondeur 7, nous trouvons deux intersections entre les segments $f^0([t^0_4, t^0_{12}])$ et $f^1([t^1_5, t^1_{13}])$ et $f^0([t^0_{11}, t^0_{33}])$ et $f^1([t^1_{14}, t^1_{39}])$. Ceci explique les variations dans le nombre d'intersections après assemblage pour le dernier niveau de profondeur. Lors de l'étape de correction, suivant la profondeur de l'arbre, l'utilisateur pourra avoir un ou deux conflits à corriger. Dans le deuxième cas, il pourra donc choisir de grouper les deux intersections pour n'en corriger qu'une seule ou corriger les deux intersections séparément. Cela dépendra du type de correction effectué.

2.4.1.2 Résultats en fonction de la précision numérique

Nous donnons maintenant dans les tableaux 2.3 et 2.4 les résultats obtenus en fonction de la précision ϵ_{num}. Les calculs sont faits pour un niveau de profondeur égal à 5. Il s'agit du niveau donnant les meilleurs temps dans la plupart des cas traités. Nous constatons que plus ϵ_{num} est grand, plus les calculs sont rapides. Lorsque l'approximation est grossière, il faut moins de points pour définir les polylignes. Il faudra donc moins de temps pour calculer l'approximation de la courbe et par conséquent pour calculer les distances entre les segments. Pour la même raison, nous détectons moins de segments en conflit.

Après regroupement, les conflits ne sont plus les mêmes. Comme ϵ_{num} est plus grand, les enveloppes contenant les approximations des courbes sont plus larges et nous devons détecter plus de conflits. Sur le tableau 2.3, le nombre d'intersections varie très peu. En fait, dans certaines zones où nous détections deux intersections à une certaine précision, nous n'en avons plus qu'une seule sur un intervalle plus large. C'est pour cela que nous obtenons moins d'intersections pour $\epsilon_{num} = 5.10^{-3}$. Sur le tableau 2.4, le nombre d'intersections après assemblage augmente régulièrement avec ϵ_{num}. Cette augmentation est liée au fait que les approximations sont plus grossières et qu'il y a alors plus de conflits. Nous aurons donc plus de conflits à corriger par la suite, ce qui augmentera le temps de correction. Compte tenu des temps et du nombre de conflits corrigés, le meilleur choix est de prendre ϵ_{num} compris entre 2.10^{-3} et 5.10^{-3} : en dessous, nous avons des résultats relativement proches pour un temps de calcul beaucoup plus important. Au dessus, le nombre de conflits est trop important pour que le gain de temps soit intéressant.

2.4.2 Courbes B-splines uniformes

Nous présentons maintenant les résultats obtenus avec un vecteur de nœuds uniforme. Les courbes ont été construites à partir des mêmes ensembles de points mais l'approximation des données initiales est le plus souvent moins bonne. L'intérêt de distinguer les deux types

	Schéma non uniforme		Intersections	
Profondeur	Temps (s)	Rapport	détectées	assemblées
3	19.32	99%	2739	393
4	17.81	98%	2770	393
5	17.73	98%	2845	393
6	16.41	97%	2990	393
7	18.89	96%	3494	395

Tab. *2.1* — Résultats pour la carte 1 en fonction de la profondeur de l'arbre (vecteur de nœuds non uniforme).

	Schéma non uniforme		Intersections	
Profondeur	Temps (s)	Rapport	détectées	assemblées
3	19.13	97%	2121	399
4	15.61	95%	2122	399
5	14.84	94%	2136	399
6	16.74	93%	2224	399
7	20.78	93%	2362	400

Tab. *2.2* — Résultats pour la carte 2 en fonction de la profondeur de l'arbre (vecteur de nœuds non uniforme).

	Schéma non uniforme		Intersections	
ϵ_{num}	Temps (s)	Rapport	détectées	assemblées
10^{-3}	55	99%	3216	392
2.10^{-3}	17.73	98%	2845	393
5.10^{-3}	7.76	95%	2697	391
10^{-2}	6.09	93%	2591	392

Tab. *2.3* — Résultats pour la carte 1 en fonction de la précision numérique ϵ_{num} (vecteur de nœuds non uniforme).

	Schéma non uniforme		Intersections	
ϵ_{num}	Temps (s)	Rapport	détectées	assemblées
10^{-3}	26.67	96%	2690	396
2.10^{-3}	14.84	94%	2136	399
5.10^{-3}	7.72	89%	1683	402
10^{-2}	6.27	87%	1448	416

Tab. *2.4* — Résultats pour la carte 2 en fonction de la précision numérique ϵ_{num} (vecteur de nœuds non uniforme).

de schémas est que, lorsque nous utilisons un vecteur non uniforme, nous avons une meilleure compression des courbes B-splines lors de l'étape de construction mais l'application du schéma de subdivision est plus coûteux lors de la détection des intersections alors que lorsque le vecteur est uniforme, l'erreur lors de la compression est plus importante mais la subdivision se fait beaucoup plus rapidement.

Nous comparons les temps obtenus en fonction de la profondeur de l'arbre et de la précision en utilisant les schémas de subdivision uniforme et non uniforme. Etant donné que les courbes ne sont pas les mêmes que dans le paragraphe précédent, cela permet de connaître le gain de temps réalisé par l'utilisation d'un schéma uniforme.

2.4.2.1 Résultats en fonction de la profondeur de l'arbre

Les temps sont donnés dans les tableaux 2.5 et 2.6 pour la carte 1 et dans les tableaux 2.7 et 2.8 pour la carte 2. Les meilleurs temps sont obtenus pour les niveaux 5 et 6. Nous voyons que pour une profondeur égale à 3, les temps sont très élevés dans tous les cas. Cela est dû au fait que les segments de courbe sont grands et que nous effectuons beaucoup de tests d'intersections entre les segments. Le schéma utilisé pour l'approximation a moins d'influence sur le temps de calcul.

Pour les autres niveaux de profondeur, les schémas uniformes sont plus rapides. La subdivision des courbes se fait plus rapidement puisque nous connaissons le nombre de subdivisions. Le gain de temps se situe donc au niveau de l'approximation des courbes. C'est pour cette raison que le pourcentage de temps passé à subdiviser les courbes et calculer les intersections est plus petit pour les schémas uniformes.

Le nombre d'intersections détectées est toujours supérieur pour les schémas non uniformes. Comme nous ne connaissons pas *a priori* le nombre de subdivisions à effectuer, nous subdivisons les segments sur des intervalles plus grands et nous détectons donc plus d'intersections. De ce fait, après assemblage, le nombre d'intersections peut être inférieur puisque certains intervalles adjacents peuvent être regroupés.

Nous remarquons enfin que le nombre d'intersections assemblées est plus grand que dans les tableaux 2.1 et 2.2. Cela est dû à l'approximation qui est moins bonne. Le système résolu lors de la construction est souvent moins bien conditionné avec un vecteur de nœuds uniforme. Les temps sont également beaucoup plus importants puisque nous devons faire beaucoup plus de calculs d'intersections entre les segments.

2.4.2.2 Résultats en fonction de la précision numérique

Les résultats sont donnés dans les tableaux 2.9 et 2.10 pour la carte 1 et dans les tableaux 2.11 et 2.12 pour la carte 2. Les schémas uniformes sont toujours plus rapides. Les différences sont surtout visibles lorsque la précision est élevée puisque c'est dans ces cas que l'on fait le plus de subdivisions. Nous détectons toujours plus d'intersections avec le schéma non uniforme

	Schéma non uniforme		Intersections	
Profondeur	Temps (s)	Rapport	détectées	assemblées
3	53.71	99%	1792	558
4	29.43	98%	1827	555
5	24.89	96%	1873	559
6	30.61	96%	2096	589
7	46.79	95%	2451	585

Tab. *2.5* — Résultats pour la carte 1 en fonction de la profondeur de l'arbre (vecteur de nœuds uniforme, schéma non uniforme).

	Schéma uniforme		Intersections	
Profondeur	Temps (s)	Rapport	détectées	assemblées
3	65.02	99%	1689	559
4	34.78	98%	1725	556
5	18.83	95%	1773	560
6	16.56	92%	2015	590
7	17.34	88%	2426	586

Tab. *2.6* — Résultats pour la carte 1 en fonction de la profondeur de l'arbre (vecteur de nœuds uniforme, schéma uniforme).

	Schéma non uniforme		Intersections	
Profondeur	Temps (s)	Rapport	détectées	assemblées
3	1687.14	100%	2169	505
4	56.78	97%	2312	505
5	48.93	95%	2334	508
6	69.99	93%	2442	514

Tab. *2.7* — Résultats pour la carte 2 en fonction de la profondeur de l'arbre (vecteur de nœuds uniforme, schéma non uniforme).

	Schéma uniforme		Intersections	
Profondeur	Temps (s)	Rapport	détectées	assemblées
3	1675	100%	1969	505
4	38.62	96%	1978	505
5	23.79	90%	2003	508
6	19.45	81%	2106	514

Tab. *2.8* — Résultats pour la carte 2 en fonction de la profondeur de l'arbre (vecteur de nœuds uniforme, schéma uniforme).

où nous avons des segments plus larges.

2.5 Conclusion

Dans ce chapitre, nous avons présenté une méthode de détection des intersections entre courbes B-splines. Cette méthode est robuste et rapide. Elle est donc adaptée aux grands ensembles de données et permet de détecter les intersections visuelles et les intersections singulières. Elle est appliquée à la détection des conflits en généralisation cartographique.

Dans une première partie, nous avons défini le problème complexe de la généralisation cartographique. Cela nous a permis de définir les intersections à détecter et de caractériser les contraintes à respecter. En particulier, nous avons vu que nous avions surtout à traiter des intersections visuelles et des recouvrements de courbe. Nous devions également être capables de gérer les auto-intersections. Ces intersections proviennent essentiellement :

- de la modélisation (choix de la paramétrisation et association à un vecteur nodal) ;
- du relief (par exemple, si la pente est forte, les isobathes sont nombreuses et très proches) ;
- de l'application des opérateurs de généralisation.

La solution retenue se décompose en deux étapes.

La première étape consiste à partitionner la carte avec un quadtree pour limiter les tests d'intersection. La répartition des segments de courbe se fait à l'intérieur de chaque cellule en recherchant les plus petits intervalles paramétriques coupant les cellules. L'intérêt de la méthode est de nécessiter très peu de calculs puisque la décomposition des courbes se fait en comparant les coordonnées des points de contrôle avec les cellules. La décomposition est donc très rapide. Nous n'insérons pas de nouveaux points sur la courbe et nous ne cherchons pas les intersections entre le bord des cellules et les courbes. Elle est également fiable puisque nous n'avons pas d'erreur d'approximation sur les calculs et que la détection se fait à une tolérance près : l'appartenance à une cellule est définie à ϵ_{vis} près. Les segments de courbe sont toujours plus larges que si nous recherchions les intersections exactes entre les courbes et les cellules. Ceci évite le problème de décision sur la position d'un point par rapport à une cellule.

L'autre intérêt de la méthode se situe au niveau du stockage des segments de courbe. Les segments sont définis uniquement par les nœuds déjà existants. Pour délimiter un segment, il suffit donc de connaître à quelle courbe il se rapporte et les indices du premier nœud et du dernier nœud. A la fin de la décomposition, nous avons un quadtree dont les cellules sont soit vides soit constituées d'une liste d'entiers désignant les courbes traversant la cellule et des couples d'indices identifiant les intervalles paramétriques correspondants. Il y a intersection potentielle dans une cellule s'il y a plusieurs segments à l'intérieur.

La deuxième étape est la détection des intersections et des auto-intersections dans chaque cellule. L'approximation se fait à une précision près ϵ_{num} définie *a priori*. Pour cela, nous

	Schéma non uniforme		Intersections	
ϵ_{num}	Temps (s)	Rapport	détectées	assemblées
10^{-3}	44.44	98%	1967	558
2.10^{-3}	24.89	96%	1873	559
5.10^{-3}	16.54	94%	1808	561
10^{-2}	11.65	92%	1741	571

Tab. *2.9* — Résultats pour la carte 1 en fonction de la précision ϵ_{num} (vecteur de nœuds uniforme, schéma non uniforme).

	Schéma uniforme		Intersections	
ϵ_{num}	Temps (s)	Rapport	détectées	assemblées
10^{-3}	35.39	98%	1806	559
2.10^{-3}	18.83	95%	1773	560
5.10^{-3}	8.9	90%	1760	560
10^{-2}	5.86	84%	1723	572

Tab. *2.10* — Résultats pour la carte 1 en fonction de la précision ϵ_{num} (vecteur de nœuds uniforme, schéma uniforme).

	Schéma non uniforme		Intersections	
ϵ_{num}	Temps (s)	Rapport	détectées	assemblées
10^{-3}	80.71	97%	2947	508
2.10^{-3}	48.93	95%	2334	508
5.10^{-3}	26.31	91%	1858	517
10^{-2}	18.61	87%	1625	537

Tab. *2.11* — Résultats pour la carte 2 en fonction de la précision ϵ_{num} (vecteur de nœuds uniforme, schéma non uniforme).

	Schéma uniforme		Intersections	
ϵ_{num}	Temps (s)	Rapport	détectées	assemblées
10^{-3}	34.39	93%	2230	508
2.10^{-3}	23.79	90%	2003	508
5.10^{-3}	12.14	78%	1785	520
10^{-2}	9.15	77%	1615	539

Tab. *2.12* — Résultats pour la carte 2 en fonction de la précision ϵ_{num} (vecteur de nœuds uniforme, schéma uniforme).

définissons une enveloppe contenant la courbe. Cette enveloppe est affinée à l'aide d'un schéma de subdivision jusqu'à ce que la précision voulue soit atteinte. Pour simplifier les calculs, nous définissons une ligne médiane à l'enveloppe puis détectons les intersections par rapport à cette ligne médiane. Les intersections correspondent aux segments qui sont à une distance inférieure à une valeur donnée par la précision visuelle et la précision numérique. La méthode traite également les auto-intersections en segmentant les arcs de courbe et en appliquant les tests d'intersection. Finalement, les intersections sont regroupées afin de disposer d'une liste de conflits sur l'ensemble de la carte et non plus dans chaque cellule.

Dans une dernière partie, nous avons appliqué notre algorithme de détection sur plusieurs jeux d'isobathes. Nous donnons les résultats en fonction de la profondeur du quadtree et de la précision des approximations pour des splines uniformes et non uniformes. Nous remarquons que la hiérarchisation permet de réduire les calculs. Cependant, lorsque la profondeur de l'arbre est trop importante, les temps sont plus importants car les intersections sont calculées plusieurs fois dans différentes cellules. Le temps de calcul et le nombre de solutions trouvées sont également liés à la précision des approximations. Plus ϵ_{num} est petit, plus le temps de calcul est grand et plus la détection est précise. Si l'approximation est grossière ($\epsilon_{num} \gg \epsilon_{mod}$), nous obtenons beaucoup plus de solutions. En conclusion et compte tenu des tests effectués, nous pouvons recommander d'utiliser des quadtrees avec cinq niveaux de profondeur et une précision numérique cmprise entre 2.10^{-3} et 5.10^{-3}.

La généralisation des isobathes se fait en deux étapes. Nous venons de présenter la première étape permettant de localiser les conflits entre les courbes. La deuxième étape consiste à supprimer ces conflits. Pour cela, plusieurs types de correction peuvent être envisagés. Dans le chapitre suivant, nous nous intéressons aux méthodes de correction fondées sur le déplacement des courbes. Les segments de courbe en conflits vont être déformés en utilisant des modèles énergétiques afin de satisfaire les contraintes cartographiques et maritimes.

CHAPITRE 3

Correction des conflits pour la généralisation de courbes B-splines

3.1 Introduction

Dans ce chapitre, nous nous intéressons à la correction d'un conflit entre deux courbes B-splines par des méthodes de déformation. Il s'agit de corriger les intersections détectées dans le chapitre 2. Les segments des courbes à modifier sont connus et repérés par les indices des points de contrôle. Les corrections doivent se faire en respectant les contraintes de sécurité et de lisibilité. Plus précisément, les corrections se font en déplaçant les courbes dans le sens des profondeurs croissantes de sorte que la distance de lisibilité soit atteinte. Ces contraintes sont des contraintes fortes et doivent absolument être respectées. Elles correspondent aux contraintes applicatives et graphiques définies dans le paragraphe 2.2.2. Une autre contrainte prise en compte est la contrainte structurale géomorphologique : afin de préserver le relief et la cohérence des données, il est important de maintenir la forme des courbes. Cette contrainte de conservation de la forme est une contrainte faible qui nous servira de critère pour évaluer la qualité des résultats.

Les algorithmes de déformation s'appuient sur des approches très variées. Les premières méthodes mises en place furent des méthodes géométriques. Les points sont déplacés en fonction des distances et des angles entre les objets. Des méthodes heuristiques comme le recuit simulé [Ware et Jones, 1998] ou les algorithmes génétiques [Wilson et al., 2003] pour la correction de conflits ainsi que les cartes de Kohonen [Jiang et Nakos, 2004] pour le lissage de lignes ont également été utilisées. Le problème est que ces méthodes sont difficilement contrôlables. Enfin, les conflits peuvent être aussi corrigés en appliquant des modèles énergétiques. Les courbes sont associées à des objets déformables soumis à des forces externes et internes. L'intérêt est que le comportement des objets est lié à des paramètres physiques que l'utilisateur peut facilement appréhender. Un prototype de généralisation utilisant les techniques multi-agents a été développé lors du projet européen AGENT [Duchêne et Regnauld, 2002]. Les agents sont des objets présents dans la base de données ou sont créés par regroupement

ou par division d'objets. Ils disposent de certaines fonctionnalités leur permettant de réaliser certaines actions de manière autonome en fonction des contraintes. Ces actions sont en fait effectuées en appliquant les méthodes citées ci-dessus.

Dans le paragraphe 3.2, nous nous intéressons aux différents opérateurs existants pour la correction d'un conflit puis expliquons comment prendre en compte les contraintes cartographiques. Pour cela, un déplacement minimal garantissant la correction du conflit et des critères pour estimer la justesse de la courbe (*fair curve*) déformée sont définis.

Dans le paragraphe 3.3, nous présentons une méthode de correction géométrique. La déformation de la courbe se fait en appliquant un déplacement à chaque point de contrôle. Dans un premier temps, ce déplacement est calculé par rapport au déplacement minimal. Ensuite, cette solution est améliorée en appliquant un processus itératif. Les points de contrôle sont considérés comme les sommets d'un réseau de barres en équilibre et soumis à l'action de forces internes et externes.

Dans le paragraphe 3.4, nous présentons une deuxième méthode fondée sur les contours actifs. Il s'agit d'un modèle déformable où les contraintes sont exprimées sous forme d'énergies. Nous discutons du choix des paramètres de forme afin d'automatiser le processus de correction pour l'appliquer à tout un ensemble de conflits. Nous présentons et comparons ensuite les résultats obtenus avec les deux méthodes.

Enfin, la méthode des contours actifs est étendue à la correction des auto-intersections visuelles et singulières en calculant un déplacement minimal adapté à ce genre de conflit.

3.2 Prise en compte des contraintes maritimes

3.2.1 Stratégie de généralisation

Lors de la généralisation d'une carte, nous devons tenir compte d'un certain nombre de contraintes que nous avons énoncées au paragraphe 2.2.2. Parmi ces contraintes, un ordre de priorité est défini. Les contraintes de sécurité et de lisibilité sont des contraintes fortes qui doivent absolument être respectées. Ensuite, les contraintes de préservation de la forme doivent conserver la morphologie du fond pour que l'utilisateur en ait une bonne représentation et puisse faire les bons choix de navigation. En effet, si l'on supprime trop d'informations, certaines routes maritimes peuvent ne plus apparaître sur la carte parce que, par exemple, un chenal a été supprimé.

Actuellement, la généralisation des cartes est effectuée manuellement par les cartographes. Même si les contraintes sont clairement définies, la stratégie de généralisation n'est pas clairement formalisée et dépend beaucoup du cartographe et de son expérience. Le choix d'un opérateur de généralisation dépend de l'information à mettre en évidence et des connaissances structurelles et procédurales doivent parfois être intégrées. L'automatisation entière du processus de généralisation n'est donc pas envisageable pour le moment du fait de la com-

plexité de certains conflits et du manque de formalisme dans le choix et l'enchaînement des opérateurs. Une approche plus intéressante est donc l'approche interactive. Cette approche fut suggérée par Weibel dans [Weibel, 1991]. Les tâches de bas niveau y sont traitées par l'ordinateur alors que les tâches telles que le choix des objets à généraliser ou les procédures de traitement sont gérées par l'utilisateur.

Les différents opérateurs de correction applicables sont la suppression, la déformation d'une isobathe et l'agrégation de deux isobathes entre elles. L'agrégation [Saux, 1999] consiste à grouper deux isobathes de même profondeur (figure 3.1). Il faut que l'une au moins soit fermée et que la contrainte de sécurité soit respectée.

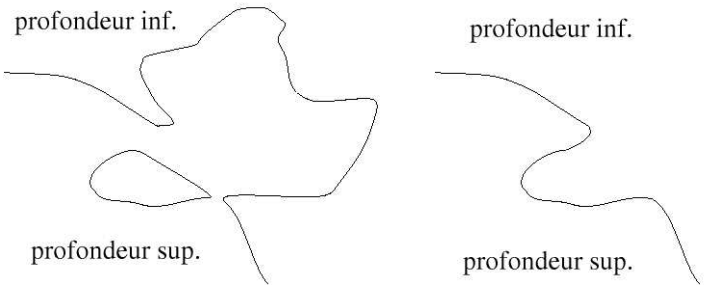

Fig. *3.1* — Agrégation de deux isobathes.

Dans les zones de fortes pentes où les isobathes se recouvrent, il est plus simple de supprimer des isobathes que de les déplacer. Dans ce cas, seules la plus profonde et la moins profonde des isobathes sont conservées (figure 3.2).

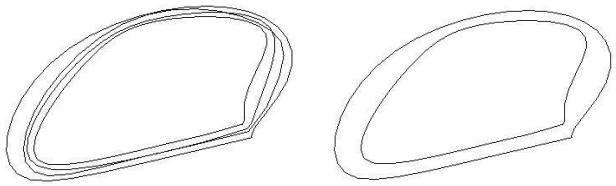

Fig. *3.2* — Suppression d'isobathes : les isobathes les moins profondes et les plus profondes sont conservées.

Enfin, la correction peut se faire par déformation. Lors de la déformation de deux courbes, seule l'isobathe la plus profonde est déplacée, l'autre étant considérée comme fixe.

Dans ce chapitre, nous nous intéressons uniquement à la correction d'un conflit par déformation. Nous ne traitons pas la correction des auto-intersections. Cependant, si l'auto-intersection a lieu entre deux segments d'une courbe définis avec des points de contrôle différents, nous pouvons appliquer les méthodes ci-dessous. La correction est uniquement géométrique et l'information sémantique n'est pas modifiée. Par exemple, la profondeur des isobathes n'est pas modifiée. Les contraintes les plus importantes sont les contraintes de sécurité et de lisibilité. Afin de respecter la contrainte de lisibilité, nous devons nous assurer

que le déplacement effectué est suffisant. Pour cela, nous calculons un déplacement minimal. Cela nous permet de définir une polyligne au delà de laquelle la courbe B-spline représentant l'isobathe doit être déplacée pour satisfaire la contrainte de lisibilité. La forme de la courbe est également considérée pour évaluer la qualité du résultat. Nous cherchons donc à limiter le déplacement et à conserver la forme de la courbe initiale. Pour cela, nous cherchons à avoir une courbe déplacée au plus proche de la courbe de déplacement minimal et dont la déformation est homogène le long de la courbe.

Ce déplacement minimal ne peut pas être considéré comme une solution de notre problème. Les isobathes à corriger sont définies par des arcs de courbes B-splines. Le déplacement minimal étant représenté par une polyligne, il n'est pas compatible avec la structure de données utilisée puisque nous travaillons avec des segments de courbes B-splines délimités par les indices des points de contrôle. De plus, le segment déformé est défini sans modifier le nombre de points de contrôle pour deux raisons. D'abord, cela permet d'intégrer facilement les changements dans la structure hiérarchique définie dans le chapitre 2. Ensuite, si nous insérons de nouveaux points, nous avons plus de libertés pour la déformation mais nous pouvons aussi créer des oscillations, ce qui n'est pas compatible avec la contrainte de forme.

La prise en compte des trois contraintes énoncées ci-dessus est détaillée dans les paragraphes suivants.

3.2.2 La contrainte de sécurité

Lorsque les courbes sont modifiées, les profondeurs indiquées sont différentes des profondeurs réelles. Pour des raisons de sécurité, la profondeur indiquée ne doit jamais être supérieure à la profondeur réelle. Les courbes doivent donc être déplacées dans le sens des profondeurs croissantes. Si nous corrigeons un conflit entre deux isobathes en déplaçant l'isobathe la moins profonde, ce déplacement doit se faire dans le sens des profondeurs décroissantes afin d'éloigner la courbe de l'isobathe la plus profonde. Ce déplacement est contradictoire avec la contrainte de sécurité. La correction d'un conflit par déformation ne peut donc se faire qu'en déplaçant l'isobathe la plus profonde dans le sens des profondeurs croissantes. De ce fait, la correction d'un ensemble de conflits se fait en partant des isobathes les moins profondes vers les plus profondes. Ainsi, lorsque toutes les isobathes à une même profondeur ont été traitées, elles ne peuvent plus être modifiées.

3.2.3 La contrainte de lisibilité

Soient deux courbes f^0 et f^1 avec f^0 en intersection avec f^1 la courbe la plus profonde. Cela signifie qu'il existe deux intervalles paramétriques $[t^0_{i_{min}}, t^0_{i_{max}}]$ et $[t^1_{j_{min}}, t^1_{j_{max}}]$ tels que la distance de lisibilité n'est pas respectée entre les arcs de courbe $f^0([t^0_{i_{min}}, t^0_{i_{max}}])$ et $f^1([t^1_{j_{min}}, t^1_{j_{max}}])$. La correction doit se faire en déformant les courbes sur ces intervalles pour que la distance entre elles soit supérieure à ϵ_{vis}. Du fait de la contrainte de sécurité, la

correction se fait en déplaçant f^1 uniquement.

La contrainte de lisibilité peut s'exprimer géométriquement. Elle correspond à un déplacement minimal devant être appliqué à la courbe pour corriger le conflit. Nous définissons ici une ligne polygonale d_{min} représentant ce déplacement minimal à appliquer aux points de f^1.

Nous avons vu dans le paragraphe 2.3.3 que les distances entre les courbes sont calculées à partir des approximations polygonales s^0 et s^1. Nous notons $P_i^0 = s^0(\zeta_i^0)$ et $P_j^1 = s^1(\zeta_j^1)$ les points de s^0 et s^1 avec ζ_i^0 et ζ_j^1 les paramètres obtenus par subdivision des courbes B-splines. Il y a conflit entre s_0 et s_1 si la plus courte distance $d > 0$ entre deux segments $P_i^0 P_{i+1}^0$ et $P_j^1 P_{j+1}^1$ est inférieure à ϵ_{vis} ou si les segments se coupent. Dans ce cas, la distance d'interpénétration est notée négativement $d < 0$. Pour supprimer le conflit, le segment $P_j^1 P_{j+1}^1$ doit être translaté dans une direction δP_j^1 de longueur $\epsilon_{vis} - d$ (figure 3.3). Cette direction est donnée par le vecteur ayant la plus petite norme et permettant de corriger l'intersection entre les segments. Pour tous les segments de s_1 en conflit, un vecteur de déplacement δP_j^1 peut être défini.

Pour respecter la contrainte de lisibilité, nous définissons un déplacement minimal ΔP_j^1 pour chaque point P_j^1. Ce déplacement est un vecteur calculé à partir des vecteurs δP_{j-1}^1 et δP_j^1. Il correspond au vecteur permettant de corriger les conflits sur les deux segments $P_{j-1}^1 P_j^1$ et $P_j^1 P_{j+1}^1$. L'ensemble des vecteurs ΔP_j^1 définit une suite de vecteurs d_{min}. Nous notons l_{min} la polyligne représentant la position valide la plus proche de s^1. d_{min} et l_{min} peuvent être paramétrées de sorte que $d_{min}(\zeta_j^1) = \Delta P_j^1$ et $l_{min}(\zeta_j^1) = s^1(\zeta_j^1) + d_{min}(\zeta_j^1)$. Le conflit sera corrigé si les segments de s^1 en conflit sont repoussés au-delà de l_{min}. S'il n'y a pas de conflit alors les polylignes s_1 et l_{min} sont égales. l_{min} et d_{min} sont donc définies avec autant de points que s^1.

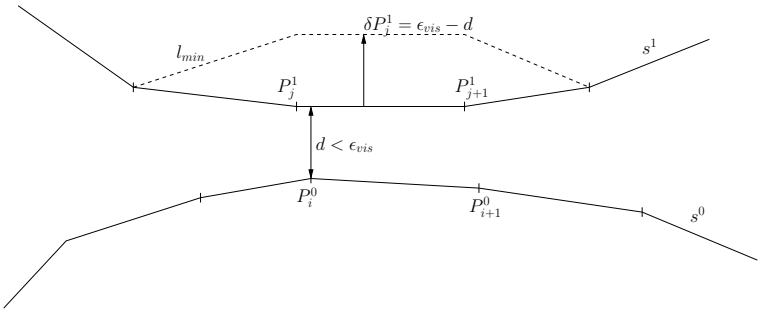

Fig. *3.3* — Calcul du déplacement minimal du segment $P_j^1 P_{j+1}^1$ en fonction de la distance avec le segment $P_i^0 P_{i+1}^0$.

Nous présentons deux exemples de déplacement minimal pour corriger des intersections. Sur les deux figures 3.4 et 3.5, nous avons à gauche les courbes en conflit et à droite la courbe l_{min} calculée à partir du déplacement minimal. Les courbes à déplacer sont en trait plein. Sur la première figure, la correction peut être faite en repoussant la courbe au delà du déplacement minimal. Sur la deuxième figure, le déplacement est impossible car il créerait

une auto-intersection. Pour cet exemple, nous devrons appliquer un opérateur de suppression. La courbe sera soit entièrement supprimée, soit scindée.

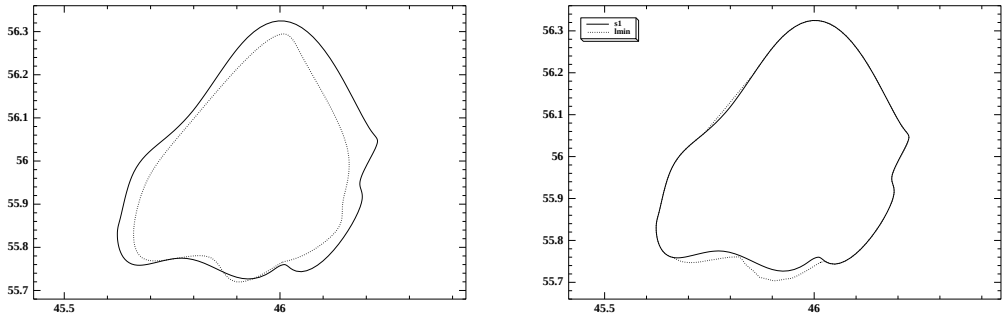

Fig. *3.4* — A gauche, les deux isobathes sont en conflit, la courbe en trait plein doit être déplacée. A droite, la courbe peut être déplacée au-delà de l_{min} sans créer de conflit.

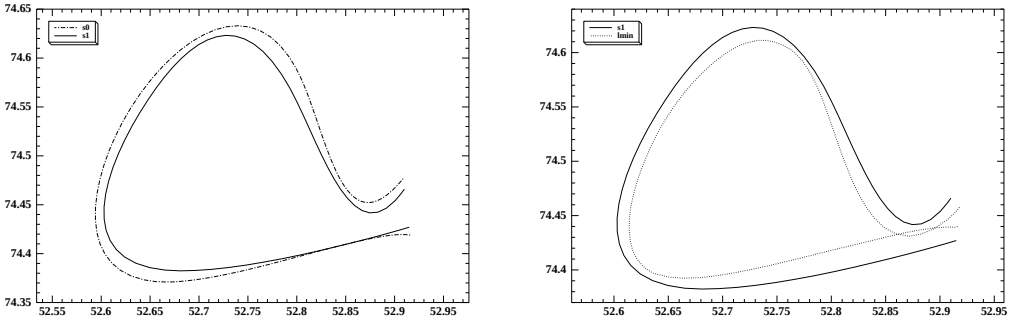

Fig. *3.5* — A gauche, la courbe en trait plein doit être déplacée car elle représente l'isobathe la plus profonde. A droite, la courbe ne peut pas être déplacée car une auto-intersection serait créée.

3.2.4 La contrainte géomorphologique

Pour évaluer la qualité des résultats, nous tenons compte de la contrainte géomorphologique. Son intérêt est de préserver le relief sous-marin et de conserver la morphologie du fond. Pour cela, les déformations doivent être limitées afin de maintenir la forme des courbes. Cette contrainte est une contrainte faible. Contrairement aux deux contraintes précédentes, elle n'est pas utilisée pour accepter ou rejeter une solution mais pour mesurer la déformation.

Nous notons $\bar{f}^1$ la courbe après déplacement, $\bar{s}^1$ son approximation et d le déplacement effectué. d est une B-spline définie par $\bar{f}^1 - f^1$ et approchée par $\tilde{d}(\zeta_j^1) = \bar{s}^1(\zeta_j^1) - s^1(\zeta_j^1)$. Pour que la contrainte géomorphologique soit respectée, il faut que le déplacement d soit petit et qu'il soit lisse pour que la déformation soit homogène et la forme de f^1 conservée.

Le déplacement effectué est évalué de deux façons. Nous mesurons le déplacement total effectué le long de la courbe et le plus grand déplacement effectué. La première mesure est donnée par la norme L_2 de d (équation (3.1)). En pratique, nous calculons une approximation discrète obtenue à partir de $\tilde{d}$ (équation (3.2)). Cela nous permet de connaître le déplacement global effectué pour la correction et indique si la courbe déplacée est proche de la courbe d'origine.

$$\|d\|_2 = \left(\int_a^b \|d(t)\|^2 dt \right)^{\frac{1}{2}} \tag{3.1}$$

$$\|\tilde{d}\|_2 = \left(\sum_{j=0}^{n-1} \|\tilde{d}(\zeta_j)\|^2 (\zeta_{j+1}^1 - \zeta_j^1) \right)^{\frac{1}{2}} \tag{3.2}$$

Un autre critère utilisé est le plus grand déplacement effectué. Il permet de majorer l'écart maximal entre la courbe initiale et la courbe finale. Là aussi, le plus grand déplacement de d est approché par le plus grand déplacement de $\tilde{d}$. Son expression est donnée par (3.3). Lorsqu'un conflit est corrigé, nous avons toujours $\|d\|_2 \geq \|d_{min}\|_2$ et $\|d\|_{max} \geq \|d_{min}\|_{max}$.

$$\|\tilde{d}\|_{\max} = \sup_{j=0,\ldots,n} \|\tilde{d}(\zeta_j^1)\| \tag{3.3}$$

Outre la proximité par rapport à la courbe d'origine, nous sommes également intéressés par la conservation de la forme de la courbe. Ce critère est relativement subjectif puisqu'il se base sur des considérations esthétiques : pour le cartographe, la forme est préservée si globalement, la courbe finale présente les mêmes inflexions et les mêmes changements d'orientation que la courbe d'origine. Notamment, il faut éviter d'introduire de nouvelles variations qui choqueraient immédiatement l'œil et il faut éviter de supprimer les oscillations sur la courbe finale qui supprimeraient certaines formes caractéristiques.

Pour préserver la forme de la courbe, il faut donc que le déplacement d soit homogène afin d'avoir une déformation qui soit régulière le long de la courbe. Le caractère lisse ou "juste" d'une courbe (*fair curve*) est généralement représenté par sa courbure. La courbure en un point étant donnée par l'inverse du rayon du cercle osculateur, cela signifie que lorsque la courbure est petite en un point de d, le déplacement varie peu autour de ce point. Nous prenons donc comme critère de forme la norme de la courbure du déplacement d. Quand cette norme est petite, la forme de la courbe est mieux préservée. Comme pour le déplacement, nous prenons comme critères de justesse la norme L_2 de la courbure et la courbure maximale. La courbure est calculée en prenant la courbure discrète en chaque point de $\tilde{d}$. Le calcul de la courbure discrète est détaillé au paragraphe 3.4.2.2.

$$\begin{array}{l} \|\kappa_d\|_2 \approx \left(\sum_{j=0}^{n-1} |\kappa_{\tilde{d}}(\zeta_j^1)|^2 (\zeta_{j+1}^1 - \zeta_j^1) \right)^{\frac{1}{2}} \\ \|\kappa_d\|_{\max} \approx \sup_{j=0,\ldots,n} |\kappa_{\tilde{d}}(\zeta_j^1)| \end{array} \tag{3.4}$$

3.3 Méthodes géométriques

En généralisation cartographique, les méthodes géométriques sont appliquées à des listes de points représentant des polylignes (routes, rivières) ou des polygones (bâtiments). Le principe de ces méthodes est d'exprimer les contraintes sous forme géométrique et de calculer le déplacement de chaque point directement en fonction de ces contraintes. Nous présentons quelques méthodes de généralisation de lignes polygonales dans le paragraphe suivant. Les courbes B-splines étant définies par un polygone de contrôle et un vecteur de nœuds, les modifications doivent être appliquées aux points de contrôle ou aux nœuds et non pas aux points de la courbe. Ces méthodes ne peuvent pas être directement utilisées pour des courbes B-splines. Par contre, le déplacement d'un point de contrôle peut être défini en fonction des déplacements des points de la courbe sous la forme d'un système d'équations. Nous présentons donc des méthodes de déformation pour les courbes B-splines en fonction de contraintes de position et de forme.

Le problème des méthodes présentées est qu'elles ne permettent pas de garantir un déplacement minimal. Elles permettent d'approcher ou d'interpoler des points données et d'intégrer des contraintes sur les dérivées mais elles ne permettent pas de garantir la position de la courbe par rapport à une autre courbe. C'est-à-dire, nous ne pouvons pas garantir que la courbe $\bar{f}^1$ est au-delà de l_{min} tout en respectant la contrainte de forme. En conséquence, nous proposons dans le paragraphe 3.3.2, une méthode de correction où le conflit est corrigé en déplaçant les points de contrôle suivant un critère géométrique.

Ensuite, nous améliorons ce résultat en considérant les points de contrôle comme les sommets d'un réseau de barres soumis à des forces internes et externes. La déformation est calculée en appliquant des efforts extérieurs tendant à ramener la courbe déplacée vers l_{min}.

3.3.1 Revue des méthodes existantes

3.3.1.1 Généralisation cartographique de polylignes

Les méthodes géométriques sont les premières à avoir été utilisées en généralisation pour la correction de conflits spatiaux entre polylignes. Il existe deux approches possibles : locale et globale. La plus ancienne est l'approche locale.

Approche locale Dans [Nickerson, 1988], l'auteur présente un processus de généralisation de lignes permettant l'élimination et la simplification de lignes ainsi que la détection et la correction des intersections. La détection des intersections entre les polylignes se fait en tenant compte de leur épaisseur sur la carte finale. Par exemple, sur une carte routière, les routes principales sont représentées avec des traits plus épais que les routes secondaires. Pour cela, chaque polyligne est remplacée par un polygone de demi-largeur d, centré sur la polyligne avec d la demi-épaisseur de l'objet sur la carte finale. Le polygone est défini en calculant deux courbes parallèles par déplacement à gauche et à droite d'une distance d (figure 3.6). Afin

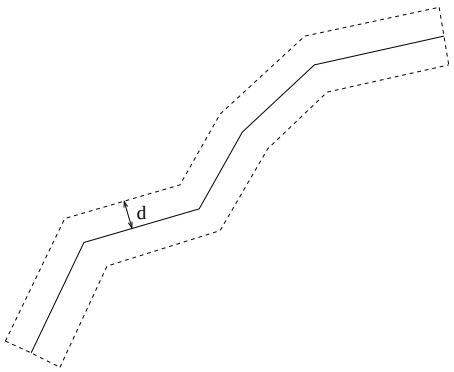

Fig. *3.6* — Polygone encadrant une polyligne de demi-largeur d.

d'éviter l'apparition de boucles sur les courbes parallèles, l'auteur donne un critère d'auto-intersection [Nickerson, 1988] :

Critère 3 *En prenant les notations de la figure 3.7, si*

$$d > \frac{\|P_{i-1}P_i\| \cos\frac{\alpha}{2} \cos\frac{\beta}{2}}{\sin(\frac{\alpha+\beta}{2})} \tag{3.5}$$

alors la courbe admet une auto-intersection.

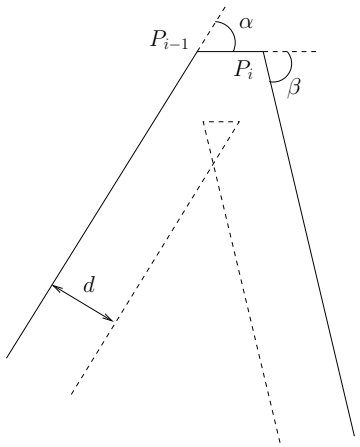

Fig. *3.7* — Auto-intersection obtenue par déplacement de la courbe.

Ensuite, la détection des conflits se fait en calculant les intersections entre les polygones. Cela fournit une liste de segments en conflit. Les intersections sont caractérisées afin de définir l'ordre dans lequel les corrections sont effectuées et quels sont les segments à déplacer. Pour chaque point à déplacer, un vecteur de déplacement corrigeant le conflit est alors calculé. Pour que le déplacement soit homogène, l'auteur choisit parmi ces vecteurs le vecteur de déplacement le plus grand et applique la même direction de déplacement à tous les points. Un filtre est également appliqué pour lisser le déplacement le long de la courbe. Le filtre utilisé par l'auteur est une fonction triangle.

Cette méthode est intéressante pour corriger des intersections simples avec de petits déplacements. Dans d'autres cas, elle semble limitée car elle ne permet pas de prendre en compte des formes complexes. De plus, les conflits sont traités et corrigés indépendamment. La qualité du résultat dépend de l'ordre dans lequel sont effectuées les corrections. Cette approche ne permet pas la prise en compte de situations complexes où une déformation doit être propagée à d'autres objets [Barrault et Bader, 2002].

Une autre méthode locale est présentée dans [Lonergan et Jones, 2001]. Il s'agit d'une méthode itérative pour la suppression des conflits spatiaux entre des bâtiments en généralisation des cartes urbaines mais cette méthode peut être étendue aux déplacements de lignes. Les conflits sont détectés en calculant les distances entre les arêtes d'un bâtiment avec les arêtes des bâtiments voisins. Si une distance est inférieure à la distance de lisibilité, il y a conflit. Dans ce cas, le bâtiment est déplacé en tenant compte du voisinage.

Pour chaque arête, un champ d'influence est défini correspondant à une zone située autour de l'arête. Ce champ d'influence s'étend à une distance $2\epsilon_{vis}$ de l'arête. Si une arête e_k est située dans le champ d'influence d'une autre arête e_j, une force orientée suivant le plus petit vecteur reliant e_j et e_k est définie. L'intensité de la force est égale à $2\epsilon_{vis} - d_{j,k}$ où $d_{j,k}$ est la distance entre les deux arêtes. Toutes les forces appliquées à un objet sont alors combinées pour calculer le déplacement de l'objet. Les auteurs ne font pas la moyenne ou la somme des forces mais calculent un déplacement suivant chaque composante x et y en additionnant pour chaque composante la plus grande valeur positive avec la plus grande valeur négative. Si le déplacement est important, il est réduit afin d'éviter les trop grandes modifications.

Chaque conflit est résolu séparément. En déplaçant un objet, d'autres conflits peuvent être créés dans le voisinage. La méthode est donc appliquée pour les objets voisins. Le processus de généralisation est itératif. La méthode est appliquée jusqu'à ce que la distance entre les bâtiments soit maximale et qu'il n'y ait plus de déplacement.

Approche globale L'approche globale consiste à considérer l'ensemble des objets de la carte et à résoudre les conflits simultanément. La résolution se fait alors à l'aide de techniques d'optimisation [Harrie, 1999, Sester, 2000]. Il ne s'agit plus de corriger tous les conflits mais de trouver un compromis entre différentes solutions. Le principe de la méthode est de définir pour chaque objet l'ensemble des contraintes géométriques auxquelles il est soumis. Les contraintes peuvent être de plusieurs sortes :

- contraintes de connexion servant à maintenir les liens entre les points d'un même objet en limitant les déformations ou en imposant un même type de déplacement aux points ;
- contraintes spatiales exprimant les intersections ou les problèmes de proximité ;
- contraintes de forme afin de lisser, simplifier ou exagérer des formes [Harrie et Sarjakoski, 2002].

Ces contraintes sont traduites sous forme d'équations où les inconnues sont les points ou les déplacements des points de chaque objet. En général, nous avons beaucoup plus de contraintes que de points. Les équations forment donc un système sur-déterminé et une meilleure solution

est recherchée à l'aide de méthodes d'optimisation.

Dans [Harrie, 1999], les contraintes sont exprimées sous la forme d'équations linéaires du type

$$\sum_{i=0}^{n} c_i \Delta P_i = \text{cste} \tag{3.6}$$

avec $n+1$ le nombre de points, P_i^0 le point avant déplacement, P_i le point après déplacement et $\Delta P_i = P_i - P_i^0$. Dans le cas où une contrainte ne s'exprime pas sous forme linéaire, comme par exemple une contrainte de courbure ou de distance, elle sera exprimée à l'aide d'un développement de Taylor. Les poids c_i sont ajoutés afin de donner plus d'importance à certaines contraintes par rapport à d'autres. Une fois que toutes les contraintes sont établies, elles sont assemblées en un seul système matriciel. Ce système étant sur-déterminé, nous nous ramenons à un problème de moindres carrés qui peut être résolu avec des méthodes de type QR.

Il est également possible de ne pas se restreindre à des contraintes linéaires. Dans ce cas, des méthodes de descente sont utilisées comme par exemple le gradient conjugué [Harrie et Sarjakoski, 2002].

Ces méthodes présentent deux inconvénients. Le premier est le réglage des coefficients c_i. Pour chaque équation, les valeurs n'ont pas le même ordre de grandeur puisque les contraintes ne sont pas toutes du même ordre. Ensuite, les coefficients dépendent de l'importance donnée aux contraintes, par exemple, si l'on veut préserver la forme des courbes ou limiter les déplacements. Le choix des coefficients nécessite donc une certaine expérience des problèmes cartographiques et est relativement empirique. Ensuite, les systèmes à résoudre peuvent être de grande taille avec un nombre d'équations beaucoup plus grand que le nombre d'inconnues. Ces méthodes recherchent une meilleure solution et ne garantissent pas que tous les conflits soient corrigés. Ceci est problématique dans le cadre de la généralisation des cartes marines où nous avons des contraintes fortes à respecter obligatoirement et où tous les conflits ne peuvent pas être corrigés par déplacement.

Comme les isobathes doivent être traitées dans un ordre donné, les méthodes de déformation locale sont plus intéressantes pour notre problème. De plus, cette approche est compatible avec une généralisation interactive où les conflits sont traités séquentiellement. Cela permet également de choisir le type d'opérateur à appliquer contrairement aux méthodes globales ou à la méthode de Lonergan où un seul opérateur de déplacement est utilisé.

3.3.1.2 Déformation de courbes B-splines

Le déplacement ou la déformation d'une courbe B-spline se fait en modifiant les points de contrôle ou les nœuds. En général, le problème est résolu en exprimant dans un système d'équations le déplacement des points de contrôle ou des nœuds en fonction des déformations à effectuer.

Récemment, des méthodes de déformation fondées sur la modification du vecteur de nœuds

ont été présentées [Hoffmann et Juhász, 2001, Goldenthal et Bercovier, 2003]. La première méthode permet de déplacer un nœud pour interpoler un point donné. Le polygone de contrôle n'étant pas modifié, le point doit se trouver dans l'enveloppe convexe. Les solutions offertes sont donc limitées car la courbe est déplacée à l'intérieur de l'enveloppe convexe du polygone de contrôle. Par cette restriction, nous ne pouvons pas garantir que les contraintes soient toujours respectées.

Dans la deuxième méthode, les auteurs considèrent le problème d'approximation ou d'interpolation de points par une courbe B-spline comme un problème d'optimisation où la fonction de coût est, suivant l'objectif à atteindre, l'erreur d'approximation ou un critère sur la forme de la courbe. La fonction de coût dépend des différentes variables définissant la courbe B-splines, c'est-à-dire, les coordonnées des points de contrôle, la paramétrisation et le vecteur de nœuds. La solution optimale est calculée itérativement. A chaque étape, la fonction coût est optimisée en fonction du vecteur de nœuds puis en fonction de la paramétrisation. L'algorithme est arrêté lorsqu'une solution stable est atteinte. Cette méthode permet de prendre en compte des critères de formes pour l'approximation de données. Le problème est que la méthode est relativement lente puisque, d'après ses auteurs, la résolution du problème peut prendre plusieurs minutes pour une courbe avec une dizaine de points de contrôle.

Plus couramment, les déformations se font en déplaçant les points de contrôle. Une méthode élémentaire serait de construire une courbe B-spline approchant l_{min}. Le problème est de s'assurer que la courbe est toujours du bon côté de l_{min}. Des conditions sur les dérivées peuvent être ajoutées pour contraindre la forme de la courbe [Dietz, 1996, Berglund et al., 2001]. Des pondérations peuvent également être introduites pour que la courbe soit plus proche de certains points que d'autres mais nous ne pouvons jamais garantir que la contrainte de lisibilité est respectée. Il existe également des méthodes locales d'approximation visant à corriger certains artefacts apparaissant sur une ligne. Dans [Farin, 1992], une méthode est présentée où l'on applique un schéma de subdivision dans le sens inverse pour réduire le nombre de points de contrôle. Le problème est que la solution n'est pas toujours valide et nous ne contrôlons pas le déplacement final. Enfin, la courbe peut également être lissée localement en corrigeant les points donnant une mauvaise approximation [Zhang et al., 2001] : lorsqu'un point n'est pas bon, l'énergie de déformation est minimisée localement et une nouvelle position est calculée.

Les méthodes présentées ont toutes la même limite : dans le cadre de la généralisation marine, elles ne permettent pas de garantir que le conflit soit corrigé et que la contrainte de lisibilité soit respectée. En effet, le problème n'est pas d'approcher au mieux l_{min} mais de définir une courbe située entièrement du même côté de l_{min}. Nous n'avons pas trouvé dans la littérature de méthodes répondant à ce problème. Une méthode d'interpolation avec des contraintes de positivité est présentée dans [Meek et al., 2003] mais cette méthode est appliquée aux courbes rationnelles non polynomiales. La positivité est établie en modifiant les poids des fonctions et non les nœuds ou les points de contrôle.

3.3.1.3 Déformation de courbes par des approches mécaniques

Les approches mécaniques consistent à appliquer des forces aux points à déplacer, ces forces étant définies en fonction des contraintes à respecter. Des forces internes reliant les points entre eux sont également définies. De ce fait, lorsqu'un point est déplacé par l'action d'une force, la déformation est propagée aux autres points. Nous présentons deux modèles mécaniques utilisés en généralisation cartographique.

Le modèle des ressorts Un modèle utilisant des ressorts est présenté dans [Bobrich, 2001] pour la déformation de polylignes. Les segments reliant les points sont assimilés à des ressorts sous tension. En fonction des conflits à corriger, les points peuvent être fixes ou libres. Des ressorts de torsion sont également placés autour des points pouvant être déplacés (figure 3.8). Lorsque des points sont modifiés, les points situés dans le voisinage sont également déplacés. Les nouvelles positions sont calculées en minimisant la somme des potentiels des ressorts de tension et de torsion. La minimisation se fait par une méthode de descente.

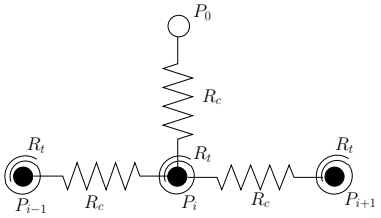

Fig. *3.8* — Modèle des ressorts : P_0 est un point fixe, R_t sont les ressorts de torsion (d'après [Bobrich, 1996]).

Les réseaux de barres Les réseaux de barres ont été introduits pour la modélisation de courbes et de surfaces par Léon dans [Léon et Trompette, 1995]. Les auteurs présentent une méthode de déformation de courbes où le polygone de contrôle est assimilé à un réseau de barres. Les déformations sont effectuées en modifiant les forces internes donnant la tension dans les barres ou les forces externes appliquées aux points de contrôle. L'intérêt de la méthode est que la condition d'équilibre s'exprime sous la forme d'un système d'équations linéaires. La déformation peut être contrôlée en fixant la position de certains points de contrôle et en laissant les autres points libres. La méthode de Léon est présentée en annexe A.

Les réseaux de barres ont été utilisés pour le lissage et le déplacement d'isobathes représentées par des courbes B-splines dans [Saux, 1999]. L'objectif est de lisser une courbe en déplaçant ses points de contrôle tout en tenant compte de la contrainte de sécurité, c'est-à-dire, en la déformant dans le sens des profondeurs décroissantes. L'opération se fait en deux étapes. D'abord, il faut choisir les points de contrôle fixes et libres. Pour cela, nous fixons les points de contrôle situés du côté de la plus grande profondeur par rapport à la courbe. Les autres points de contrôle sont laissés libres. Ensuite, les points de contrôle libres sont

déplacés dans le sens des profondeurs croissantes. De ce fait, la sécurité est préservée et la courbe est lissée puisque les points de contrôle libres sont plus proches des points fixes.

Le déplacement de la courbe se fait en appliquant des forces aux points de contrôle libres. L'orientation est donnée par la contrainte de sécurité. La force en un point de contrôle Q_i est dirigée suivant la normale intérieure N_i de l'angle (Q_{i-1}, Q_i, Q_{i+1}) (figure 3.9 à gauche) ou suivant les branches du polygone de contrôle (figure 3.9 à droite) en fonction de la géométrie de la courbe et de la distribution des points fixes et libres.

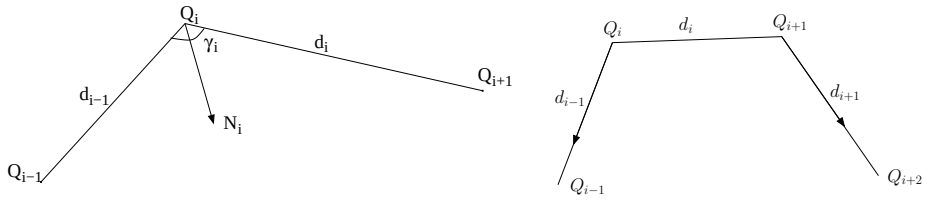

Fig. *3.9* — Direction et sens des efforts extérieurs : suivant la normale à gauche ou suivant les branches du polygone de contrôle à droite.

L'intensité des forces est souvent empirique et dépend du résultat voulu. Notamment, lorsque la forme de la courbe est complexe (avec des étranglements et des pics), il est difficile de choisir de bons paramètres permettant de respecter l'ensemble des contraintes et de préserver la géométrie.

3.3.2 Calcul d'une solution géométrique

Nous voyons que l'ensemble des méthodes présentées ci-dessus ne permettent pas de garantir des contraintes aussi fortes que les contraintes de lisibilité et de sécurité. Nous proposons dans ce paragraphe une méthode de correction tenant compte de ces contraintes spécifiques à la cartographie marine.

Nous avons caractérisé les contraintes de lisibilité et de sécurité grâce à la ligne de déplacement l_{min}. Nous devons maintenant faire passer f^1 du bon côté de l_{min} en déplaçant les points de contrôle de $f^1([t^1_{j_{min}}, t^1_{j_{max}}])$ définissant le segment en conflit. Supposons que le segment $P^1_j P^1_{j+1}$ avec $t^1_i \leq \zeta^1_j < \zeta^1_{j+1} \leq t^1_{i+1}$ doive être déplacé (dans le cas où nous avons $t_{i-1} \leq \zeta^1_j < t_i < \zeta^1_{j+1} \leq t_{i+1}$, nous nous ramènerons au cas général présenté ci-après). L'intervalle $[t^1_i, t^1_{i+1}]$ est lié aux points de contrôle $Q^1_{i-k+1}, \ldots, Q^1_i$. L'équation de la courbe sur cet intervalle est :

$$f(t) = \sum_{l=i-k+1}^{i} Q^1_l N^k_l(t), \quad t \in [t^1_i, t^1_{i+1}] \tag{3.7}$$

Si les points de contrôle sont déplacés de ΔP_j^1, nous avons

$$\sum_{l=i-k+1}^{i} (Q_l^1 + \Delta P_j^1) N_l^k(t) = \sum_{l=i-k+1}^{i} Q_l^1 N_l^k(t) + \Delta P_j^1 \sum_{l=i-k+1}^{i} N_l^k(t) \quad (3.8)$$

$$= f(t) + \Delta P_j^1 \quad (3.9)$$

Par conséquent, en déplaçant les points de contrôle de ΔP_j^1, la courbe est déplacée sur l'intervalle correspondant et le conflit est corrigé.

Pour corriger l'intersection de ΔP_j^1, il est possible de prendre une autre direction v. Il faut que l'orientation du déplacement soit la même, c'est-à-dire $\Delta P_j^1.v > 0$. Ensuite, il faut que l'intensité du déplacement soit suffisante pour que la distance entre le segment avant et après déplacement soit au moins égale à $\|\Delta P_j^1\|$ (figure 3.10). Cela signifie que la norme du déplacement suivant v doit être au moins égale à $\frac{\|\Delta P_j^1\|}{\cos(\Delta P_j^1, v)}$.

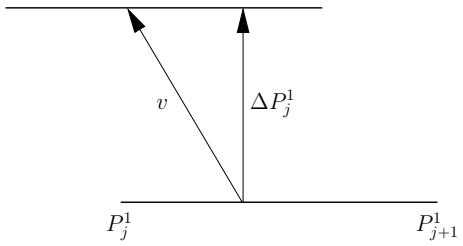

Fig. *3.10* — Déplacement minimum suivant une direction quelconque v conservant la distance entre les deux segments.

Supposons maintenant que nous ayons plusieurs segments $[P_j^1, P_{j+1}^1]$ en conflit pour un même intervalle $[t_i^1, t_{i+1}^1]$. Dans ce cas, il faut choisir une direction ΔQ_i^1 corrigeant tous les conflits et donnant le plus petit déplacement. Dans certains cas, le plus grand déplacement ΔP_j^1 peut corriger tous les conflits. Nous avons alors $\Delta Q_i^1 = \Delta P_j^1$ (figure 3.11). En général, ceci n'est pas possible. Sur l'exemple de la figure 3.12, nous donnons l'ensemble des solutions possibles de ΔQ_i^1 pour deux directions ΔP_1^1 et ΔP_2^1. La meilleure solution est donnée par l'intersection des droites orthogonales à ΔP_1^1 et ΔP_2^1. Cette méthode peut être étendue à plusieurs vecteurs en calculant les intersections deux à deux puis en prenant la solution corrigeant tous les conflits.

Le calcul des intersections entre les droites orthogonales est d'autant plus coûteux que les ΔP_j^1 sont nombreux. Afin de réduire les calculs, nous restreignons l'ensemble des directions possibles pour ΔQ_i^1 aux directions données par les ΔP_j^1. Cela permet de limiter les calculs et de ne pas traiter le cas où un ΔP_j^1 est solution comme un cas particulier. Pour chaque direction ΔP_j^1, nous calculons la norme du déplacement corrigeant tous les conflits puis nous choisissons la direction donnant le plus petit déplacement.

$$\|\Delta Q_i^1\| = \min_p \max_j \frac{\|\Delta P_j^1\|}{\cos(\Delta P_p^1, \Delta P_j^1)}, \forall p, j \quad \zeta_p^1, \zeta_j^1 \in [t_i, t_{i+1}] \quad (3.10)$$

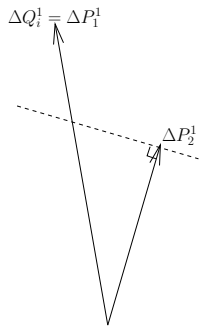

Fig. *3.11* — Le déplacement ΔP_1^1 corrige le conflit.

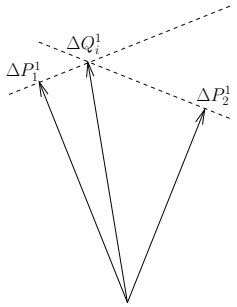

Fig. *3.12* — Calcul du plus petit déplacement corrigeant le conflit.

Le vecteur ΔQ_i^1 solution de (3.10) est dirigé suivant le ΔP_p^1 minimisant (3.10) (figure 3.13). Ce déplacement est appliqué aux points de contrôle $Q_{i-k+1}^1, \ldots, Q_i^1$.

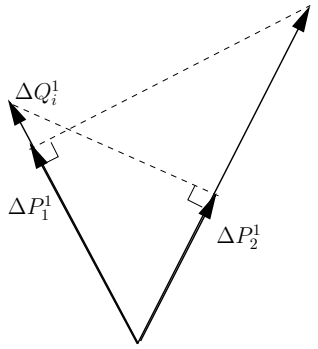

Fig. *3.13* — Déplacement ΔQ_i^1 d'un point de contrôle Q_i^1 en fonction des déplacements ΔP_j^1.

Dans le cas général, nous avons plusieurs segments $P_{j_{\min}}^1 \ldots P_{j_{\max}}^1$ en conflit avec $t_{i_{\min}} \leq \zeta_{j_{\min}} < \zeta_{j_{max}} < t_{i_{\max}}$. Les points de contrôle à modifier sont les points $Q_{i_{\min}-k+1}, \ldots, Q_{i_{\max}-1}$. Un point Q_i est lié aux k intervalles $[t_i, t_{i+1}], \ldots, [t_{i+k-1}, t_{i+k}]$. D'après (3.10), il y a donc k valeurs possibles de déplacement pour chaque Q_i. Parmi ces k valeurs, nous prenons à nouveau celle permettant le plus petit déplacement. Cela revient, pour chaque Q_i, à calculer un déplacement ΔQ_i correspondant au plus petit déplacement corrigeant tous les conflits sur

l'intervalle $[t_i, t_{i+k}]$. La longueur du déplacement est donnée par (3.11).

$$\|\Delta Q_i\| = \min_p \max_j \frac{\|\Delta P_j^1\|}{\cos(\Delta P_p^1, \Delta P_j^1)}, \forall p, j, \quad \zeta_p, \zeta_j \in [t_i, t_{i+k}] \tag{3.11}$$

où ΔQ_i est orienté dans le sens du ΔP_p^1 minimisant (3.11).

Enfin, une dernière chose à vérifier est que ΔQ_i est bien orienté dans le sens de la sécurité. Pour cela, il faut comparer le vecteur de déplacement avec les branches Q_iQ_{i-1} et Q_iQ_{i+1} du polygone de contrôle : si Q_i est déplacé du côté plus profond du polygone de contrôle, la courbe est déformée dans le sens des profondeurs croissantes. Sinon, Q_i est déplacé du côté le moins profond et la courbe est déformée dans le sens contraire de la sécurité. Dans ce cas, nous recalculons un ΔQ_i corrigeant le conflit mais orienté dans la direction de la branche la plus proche (figure 3.14) suivant la même formule que précédemment.

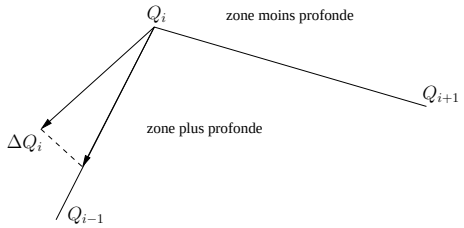

Fig. *3.14* — Déplacement en fonction de la sécurité : ΔQ_i doit être projeté sur la branche Q_iQ_{i-1}.

Cette méthode donne de bons résultats lorsque la géométrie des courbes est simple. En effet, nous supposons dans la formule (3.11) que le cosinus est positif. Si la courbe a une forme complexe, par exemple, s'il y a des étranglements, le cosinus peut changer de signe et (3.11) n'est plus valide. Nous choisissons alors ΔQ_i dirigé suivant la bissectrice de l'angle formé par Q_{i-1}, Q_i, Q_{i+1} et orienté dans le sens des profondeurs croissantes. L'intensité du déplacement est alors fixée par :

$$\|\Delta Q_i\| = \max_j \frac{\|\Delta P_j^1\|}{|\cos(\Delta Q_i, \Delta P_j^1)|}, \forall j, \quad \zeta_j \in [t_i, t_{i+k}] \tag{3.12}$$

Dans ce cas, le choix de la direction étant arbitraire, nous devons alors contrôler si la solution obtenue est valide. Si la courbe n'est pas entièrement du bon côté de la sécurité, nous recalculons à partir de cette courbe une nouvelle solution avec la même méthode.

Dans la figure 3.15, nous donnons la correction obtenue par cette méthode pour supprimer le conflit de la figure 3.4.

La solution donnée par l'équation (3.11) nous fournit un déplacement suffisant pour corriger l'intersection mais qui peut être parfois très grand. Par exemple, sur la figure 3.16, le conflit est corrigé mais le déplacement est beaucoup plus grand que nécessaire : en corrigeant l'intersection, une nouvelle auto-intersection a été créée. Il est donc intéressant de modifier le résultat obtenu avec cette méthode afin d'effectuer un plus petit déplacement. Pour cela, nous utilisons l'approche des réseaux de barres citée au paragraphe 3.3.1.3.

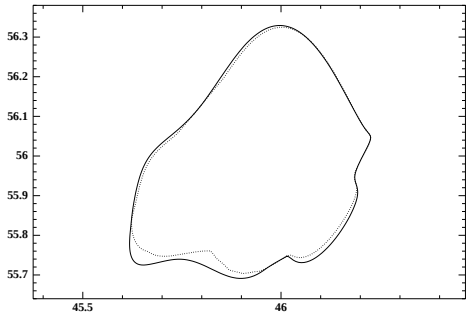

Fig. *3.15* — Méthode géométrique : le conflit est corrigé car la courbe obtenue après déformation (en trait plein) est passée au-delà de l_{min} définie à partir du déplacement minimal (en pointillés).

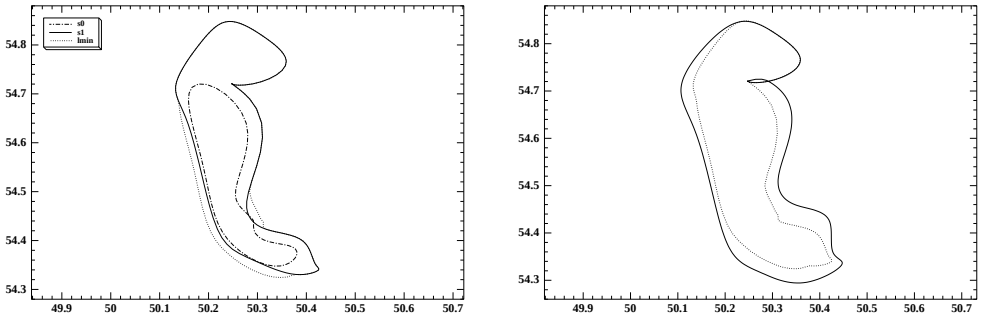

Fig. *3.16* — A gauche : l'isobathe à l'extérieur doit être déplacée. A droite : solution géométrique (trait plein) et déplacement minimal (pointillés).

3.3.3 Amélioration de la solution par déformation mécanique

Nous proposons ici une méthode pour améliorer la solution géométrique [Guilbert et al., 2004]. Elle consiste à appliquer des déplacements progressifs aux points de contrôle pour attirer la courbe vers l_{min}. Pour cela, nous considérons le polygone de contrôle comme un réseau de barres soumis à des forces.

La difficulté de l'approche mécanique est de déterminer l'intensité des forces donnant le déplacement voulu. Comme Léon, nous considérons pour notre problème que les barres sont homogènes et que la densité de force est constante dans tout le réseau. Seules les forces externes sont inconnues et ce sont leurs variations qui impliquent les déformations du polygone.

Nous notons f^1 la courbe B-spline avant déformation et $\tilde{f}^1$ la courbe B-spline obtenue par la méthode géométrique. Nous considérons les polygones de contrôle de chaque courbe comme des réseaux de barres en équilibre. Pour chacun de ces réseaux, les coordonnées des points sont connues et les forces externes sont inconnues (nous supposons que la densité de force interne est constante). A partir du système d'équations (A.6), nous calculons les forces externes pour chaque réseau.

Nous notons F_i^{ext} les forces externes appliquées aux points de contrôle Q_i de f^1 et $\tilde{F}_i^{ext}$ les forces externes appliquées aux points de contrôle $\tilde{Q}_i$ de $\tilde{f}^1$. Nous définissons $\Delta F_i^{ext} = F_i^{ext} - \tilde{F}_i^{ext}$ les variations de force entre les points de contrôle de $\tilde{f}^1$ et de f^1. Les ΔF_i^{ext} représentent les forces externes qu'il faut appliquer à $\tilde{f}^1$ pour passer de $\tilde{f}^1$ à f^1 par déformation mécanique.

Si des forces $c\Delta F_i^{ext}$ sont appliquées aux points de $\tilde{f}^1$ avec $0 < c < 1$, nous obtenons une courbe située entre f^1 et $\tilde{f}^1$. Le résultat obtenu par la méthode géométrique peut donc être amélioré en choisissant un coefficient $c < 1$ nous donnant une solution admissible (c'est-à-dire située du bon côté de $l_{\min}$), l'intérêt étant de prendre le coefficient le plus grand possible pour être au plus près de f^1.

Pour cela, nous procédons par dichotomie sur le paramètre c. L'intervalle de départ est $[0, 1]$. Nous cherchons un intervalle $[c_{min}, c_{max}]$ tel que c_{min} donne une solution admissible et c_{max} donne une solution non admissible. A chaque fois, le calcul d'une nouvelle solution se fait en résolvant l'équation (A.6) avec les coordonnées des points libres comme inconnues. Pour savoir si une solution est admissible, nous calculons une approximation de la courbe avec la méthode de subdivision présentée au paragraphe 2.3.3 et regardons si l'approximation est du bon côté par rapport au déplacement minimal l_{min}. Nous arrêtons lorsqu'une largeur donnée pour l'intervalle ou lorsqu'un nombre d'itérations est atteint. La nouvelle courbe est obtenue en appliquant les forces $c_{min}\Delta F_i^{ext}$ sur les points de contrôle de $\tilde{f}^1$.

La solution calculée à partir de c_{max} n'étant pas admissible, cela signifie que certains segments de la courbe sont du mauvais côté par rapport à l_{min}. Les points de contrôle définissant ces segments sont donc mal placés. Comme la solution calculée à partir de c_{min} est admissible, cela signifie que ces points de contrôle ne peuvent plus être déplacés. Par contre, les autres points peuvent encore être déplacés afin d'améliorer le résultat. Les points ne pouvant plus être déplacés sont donc fixés et de nouvelles forces sont appliquées aux points libres.

Comme la structure du réseau a été modifiée (les points de contrôle ont été déplacés et certains points sont fixes), les forces externes ne sont plus les mêmes. Nous calculons donc de nouvelles forces $\tilde{F}_i^{ext}$ et de nouvelles variations de forces ΔF_i^{ext}. Le processus de dichotomie ci-dessus est à nouveau appliqué pour déformer la courbe et aboutir à une nouvelle solution plus proche de la courbe d'origine. Là encore, certains points de contrôle peuvent être fixés pour relancer la méthode. Nous relançons alors l'algorithme jusqu'à ce que tous les points de contrôle soient fixés, ce qui signifie que la courbe ne peut plus être déformée.

La méthode est illustrée sur les figures 3.17 et 3.18. Sur la figure 3.17 à gauche, nous avons le polygone de contrôle de la solution géométrique. Les points de contrôle représentés par des croix sont fixes. Les autres points de contrôle sont modifiés en leur appliquant des forces externes. L'intensité des forces est calculée par dichotomie pour avoir le plus grand déplacement possible. Un déplacement plus important créerait un conflit puisque la courbe couperait l_{min} (figure 3.17 à droite. Nous obtenons alors un nouveau polygone de contrôle représenté par des tirets. Les points de contrôle définissant le segment où le déplacement est

maximal sont fixés et de nouvelles forces sont calculées pour déplacer les points de contrôle libres restants. Les étapes suivantes sont représentées sur la figure 3.18. A chaque étape, de nouveaux points de contrôle sont fixés. L'algorithme de la méthode est résumé par l'algorithme 3.

Algorithme 3 *Amélioration de la solution géométrique à l'aide d'un réseau de barres.*

```
Début
Données en entrée : courbe initiale f^1, solution géométrique f̃^1
     Choix des points libres
     Calcul des forces externes de f^1
     Tant qu'il reste des points de contrôle libres
          Calcul des forces externes de f̃^1
          Calcul des variations de force ΔF
          c_min = 0,  c_max = 1
          Tant que c_max − c_min >seuil
               c = (c_min + c_max)/2
               Application des forces −cΔF
               Si la solution est valide
                    c_min = c
                    Obtention d'une nouvelle courbe f̂^1
               sinon
                    c_max = c
               Fin si
          Fin tant que
          Fixer les points de contrôle non admissibles
          f̃^1 = f̂^1
     Fin tant que
Données en sortie : f̃^1 solution mécanique
Fin
```

Nous présentons sur la figure 3.19 à gauche la solution obtenue à partir de la solution géométrique de la figure 3.15 et à droite les isobathes de la figure 3.4 après correction. Dans le tableau 3.1, nous donnons les normes des déplacements pour les deux solutions. Nous voyons que la solution mécanique est meilleure que la solution géométrique en terme de déplacement.

Sur la figure 3.20, nous donnons la correction obtenue pour le conflit de la figure 3.16. Les différentes mesures sont données dans le tableau 3.2. La solution mécanique donne un résultat nettement meilleur pour la distance de déplacement. Dans notre méthode de correction, nous ne tenons compte que de la distance. La contrainte de forme n'est pas prise en compte pour calculer le déplacement des points. De plus, le sens de déplacement est imposé par l'orientation des forces. Enfin, pour le deuxième conflit, même si la déformation est moins importante

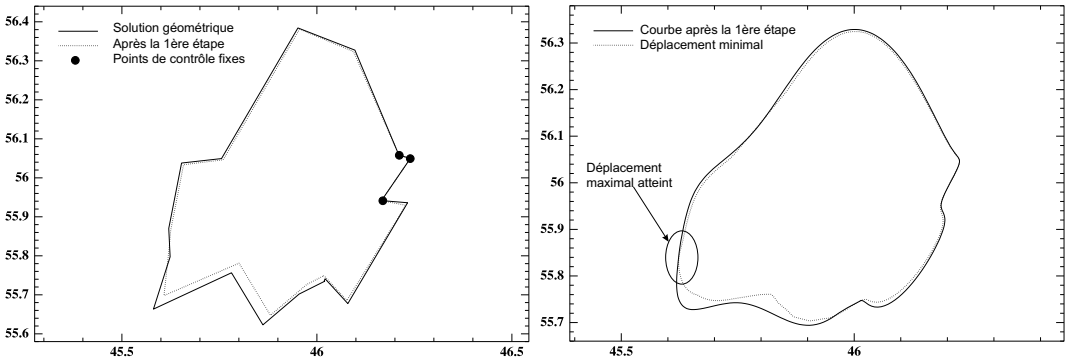

Fig. ***3.17*** — Polygone de contrôle et courbe obtenus après la première étape de déformation mécanique.

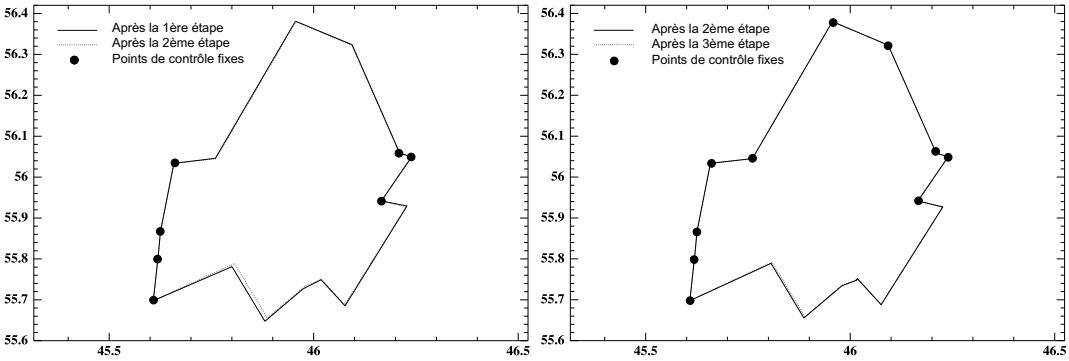

Fig. ***3.18*** — Polygones de contrôle lors des deuxième et troisième étapes de déformation.

qu'avec la méthode géométrique, nous avons toujours une auto-intersection. Pour éviter ce problème, il faudrait que la déformation se fasse plus localement pour que la courbe soit modifiée uniquement dans une petite zone autour du conflit. Nous présentons dans la partie suivante une méthode énergétique dans laquelle la forme des courbes est prise en compte et permettant des déformations plus locales.

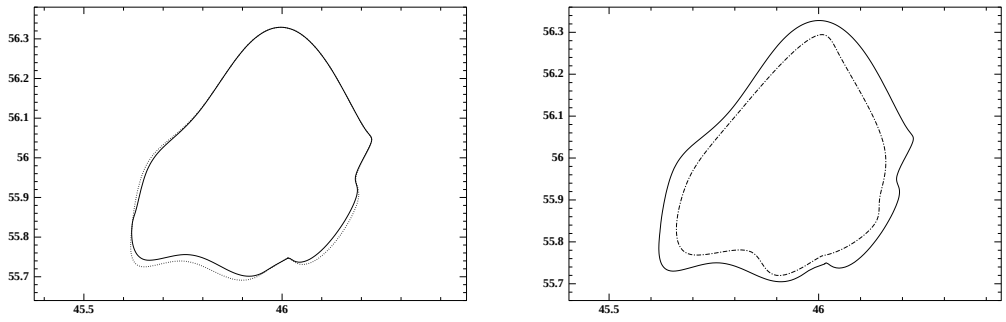

Fig. ***3.19*** — A gauche : solutions mécanique (trait plein) et géométrique (pointillés). A droite : isobathes après correction.

	$\|d\|_2$	$\|d\|_{\max}$
Déplacement minimal	0,027	0,029
Solution géométrique	0,129	0,078
Solution mécanique	0,054	0,035

Tab. ***3.1*** — Norme des déplacements.

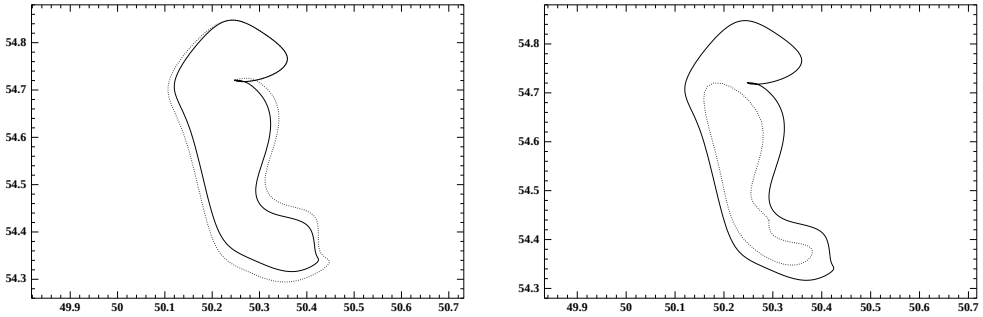

Fig. ***3.20*** — A gauche : solutions géométrique et mécanique. A droite : isobathes après déformation mécanique.

3.4 Méthodes énergétiques

Les méthodes énergétiques consistent à appliquer un modèle physique à un objet. L'objet dispose alors d'une énergie interne engendrée par les forces internes liées à sa structure et à sa forme ainsi que d'une énergie externe liée à l'environnement. Un système physique est en équilibre lorsque son énergie est minimale. L'objet considéré va donc se déformer afin de minimiser son énergie totale et atteindre cette position d'équilibre.

Par rapport aux méthodes présentées dans le paragraphe précédent, ces méthodes ont plusieurs intérêts. La déformation est calculée globalement sur toute la courbe alors que pour la méthode géométrique, les déplacements des points de contrôle sont calculés indépendamment des déplacements voisins. Le modèle physique est un modèle continu. Son énergie est définie en tout point de la courbe alors que pour les réseaux de barres, les forces ne sont définies qu'aux points de contrôle.

Deux modèles déformables sont présentés : les poutres élastiques (*elastic beam*) et les

	$\|d\|_2$	$\|d\|_{\max}$
Déplacement minimal	0,032	0,021
Solution géométrique	0,132	0,049
Solution mécanique	0,055	0,021

Tab. ***3.2*** — Normes des déplacements et des courbures.

contours actifs. Le premier modèle traite la courbe comme une poutre flexible et fait appel aux propriétés de la mécanique des structures. Le deuxième modèle a été utilisé à l'origine en traitement d'images.

3.4.1 Les modèles déformables

3.4.1.1 Les poutres élastiques

La méthode présentée est extraite de [Bader, 2001]. Une ligne polygonale est assimilée à une poutre élastique soumise à des tractions et des flexions. Seules les tractions appliquées suivant l'axe de la barre et les flexions normales à la barre sont prises en compte. Les notions de base de la mécanique des structures sont résumées en annexe B. Dans notre cas, la loi de Hooke se résume à :

$$\sigma_x = E\epsilon \tag{3.13}$$

avec σ_x la contrainte interne dans la direction de l'axe de la barre, ϵ la contrainte externe et E le module de Young.

L'auteur utilise une méthode par éléments finis inspirée de [Højholt, 2000]. Les éléments sont définis comme les segments des lignes polygonales. L'équilibre du réseau est obtenu en minimisant l'énergie totale. Cette énergie est composée d'une énergie interne due aux contraintes de tension et de flexion (figure 3.21) et d'une énergie externe liée aux forces extérieures.

Fig. *3.21* — Déformation d'une barre : traction (à gauche) et flexion (à droite).

L'énergie de tension est calculée dans la direction de l'axe de la barre en fonction de la déformation u de la barre et dépend de l'élasticité E, de la longueur L et d'une constante A représentant l'aire de la section de la poutre :

$$U = \int_L \frac{AE}{2} \left(\frac{du}{ds}\right)^2 ds \tag{3.14}$$

avec s l'abscisse curviligne.

L'énergie de flexion dépend du moment d'inertie I de la barre et est donnée par :

$$U = \int_0^L \frac{EI}{2} \left(\frac{d^2u}{ds^2}\right)^2 ds \tag{3.15}$$

Les déformations sont provoquées par l'application de forces externes aux points à déplacer. Ces forces sont définies en fonction de la proximité avec les courbes voisines. Pour

un point P d'une courbe s^j et en notant v_i le vecteur reliant P au point le plus proche de s^i, la force au point P est choisie comme étant :

$$F_P = \sum_{i \neq j} \frac{v_i}{|v_i|}(\epsilon_{vis} - \min(|v_i|, \epsilon_{vis})) \tag{3.16}$$

Cette formule permet de déduire l'énergie externe en un point sachant que la force en un point est égale au gradient d'énergie.

Les équations sont exprimées dans des repères locaux pour chaque élément. L'intérêt est que les forces ne sont pas exprimées dans les directions x et y mais suivant l'axe de la barre et suivant la normale. Ensuite, toutes les équations sont assemblées dans une matrice exprimée dans un repère global. Le problème s'écrit donc sous la forme d'un système $AX = F$ où A est la matrice des coefficients, X les coordonnées des points et F les composantes des forces en chaque point. Le problème à résoudre n'étant pas un problème linéaire (les énergies sont fonction des dérivées premières et secondes), le système doit être résolu itérativement par petits déplacements successifs pour éviter les grandes variations de X. C'est-à-dire, au lieu de résoudre directement $AX = F$, un coefficient de relaxation γ est introduit et, à chaque étape n, le système $(1 + \gamma A)X^{(n)} = X^{(n-1)} + \gamma F^{(n-1)}$ est résolu jusqu'à ce qu'une solution stable soit atteinte.

3.4.1.2 Les contours actifs

Définition Les contours actifs ou snakes ont été utilisés à l'origine en traitement d'images pour détecter des contours d'objets [Kass et al., 1987]. Le principe est de placer une courbe sur l'image et de la déformer pour adhérer au contour recherché. La solution est obtenue lorsqu'une position stable est atteinte, c'est-à-dire lorsque l'énergie du snake est minimale.

Si nous notons $u(s)$ le snake de longueur l avec s l'abscisse curviligne, son énergie se décompose sous la forme

$$E_{snake} = \int_0^l E_{int}(u(s)) + E_{ext}(u(s))ds \tag{3.17}$$

où E_{int} représente l'énergie interne à la courbe permettant de contrôler la rigidité et la flexion de la courbe et E_{ext} représente les contraintes engendrées par l'image. Par exemple, dans le cas de la détection de contours, un contour est caractérisé par un changement de couleur. L'énergie externe est donc liée au gradient de couleur et est donnée par $E_{ext}(x, y) = -|\nabla I(x, y)|^2$ où I est la couleur en un point (x, y) de l'image.

En minimisant l'énergie du système (3.17), le contour se déplace vers les zones où l'énergie externe est la plus faible tandis que le premier terme E_{int}, défini en fonction des dérivées de la courbe et donné par l'équation (3.18), sert à contraindre l'allure de la courbe. Les variables α et β servent à contrôler la tension donnée par la dérivée première et la flexion donnée par la dérivée seconde. En général, les paramètres α et β sont fixés manuellement et sont constants

par rapport à s.

$$E_{int} = \frac{1}{2}\left(\alpha(s)\left|\frac{du}{ds}(s)\right|^2 + \beta(s)\left|\frac{d^2u}{ds^2}(s)\right|^2\right) \tag{3.18}$$

Sur une image, il y a généralement plusieurs contours qui peuvent être détectés. Chaque contour se caractérise comme étant un minimum local de la fonction d'énergie (3.17). En pratique, l'utilisateur pose un snake dans le voisinage d'un contour et recherche le minimum local de (3.17) dans ce voisinage. Pour cela, la résolution se fait itérativement par déplacement progressif du snake. En effet, la recherche d'une solution directe pourrait nous donner une solution non acceptable comme un snake adhérant à deux contours différents (figure 3.22). Pour limiter ce problème, la fonction d'énergie externe peut être lissée par un filtre gaussien. Lorsque le snake se rapproche d'un contour, le lissage est progressivement supprimé afin d'améliorer la précision. Dans [Radeva et al., 1995], les auteurs proposent, pour la détection de contours, d'ajouter une contrainte d'orthogonalité pour que le snake soit normal au gradient de couleur. Enfin, dans [Cohen et Cohen, 1993], un terme de pression est ajouté pour que le snake se dilate jusqu'à adhérer à un contour.

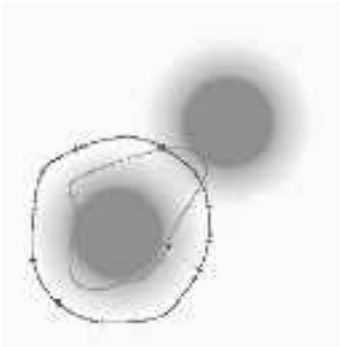

Fig. *3.22* — Snake attiré par deux contours différents : contour initial en noir, contour final en gris (extrait de [Jacob et al., 2001]).

Lorsque le snake est représenté sous la forme d'une ligne polygonale, les dérivées et la fonction d'énergie externe sont calculées sous forme discrète en chaque point de la ligne. Les représentations B-splines ont également été utilisées dans [Brigger et al., 2000] pour la détection de contours et dans [Pottmann et al., 2002] pour l'approximation de courbes et surfaces. L'utilisation de B-splines présente certains avantages [Brigger et al., 2000] :

- la représentation de la courbe nécessite moins de points qu'un snake polygonal ;
- le caractère lisse est intrinsèque à la courbe ;
- un meilleur contrôle de la courbe est obtenu grâce aux points de contrôle.

Les contours actifs peuvent aussi être représentés par des courbes NURBS [Meegama et Rajapakse, 2003]. Les poids permettent d'avoir plus de flexibilité et sont réglés en fonction des critères de forme : les auteurs définissent des poids plus importants aux endroits où la courbure est forte afin de mieux approcher les contours des objets.

Dans le cadre de l'approximation, si l'erreur entre le snake et les données est importante,

de nouveaux points de contrôle peuvent être insérés [Yang et al., 2004]. Cette technique n'est pas utilisée pour notre problème car elle n'est pas compatible avec la préservation de la forme de la courbe. En effet, si de nouveaux points de contrôle sont insérés, il peut apparaître des oscillations qui n'existaient pas sur la courbe au départ. La forme de la courbe est alors plus complexe. De plus, l'insertion de points n'est pas compatible avec la structure de stockage puisque les segments en conflit sont repérés par les indices des points de contrôle. Cela demanderait à parcourir la structure pour remettre les indices à jour.

Expression de l'énergie interne d'un snake B-spline Dans (3.18), l'énergie interne est fonction de la dérivée première et de la dérivée seconde données en fonction de l'abscisse curviligne. Ces valeurs ont un sens physique. Si l'on considère la courbe comme la trajectoire d'un objet, la dérivée première donne la vitesse de l'objet à l'abscisse s et son énergie correspond à la longueur de la courbe. Minimiser l'énergie suivant la dérivée première revient donc à tendre la courbe pour en réduire la longueur. Nous disons donc que le premier terme représente la tension de la courbe.

La dérivée seconde peut être interprétée de deux façons. Dans le cas du déplacement d'un objet le long de la courbe, elle représente l'accélération. Elle définit également la courbure en l'abscisse s. Cela signifie que minimiser l'énergie de la dérivée seconde revient à lisser la courbe en atténuant les oscillations.

Lorsque le snake est représenté par une courbe B-spline, il est exprimé sous la forme d'une courbe paramétrique. Les énergies doivent donc être exprimées suivant le paramètre t à la place de l'abscisse curviligne s. La dérivée première suivant t a la même signification que suivant l'abscisse curviligne s. Par contre, la dérivée seconde suivant t ne correspond plus à la courbure. Pour pallier ce problème, les auteurs de [Jacob et al., 2001] définissent une paramétrisation proche de l'abscisse curviligne et considèrent que la dérivée seconde constitue une bonne approximation de la courbure. Ceci n'est pas toujours possible.

L'abscisse curviligne s est donnée par $s(t) = \int_a^t \|u'(t)\|dt$ où u est définie sur l'intervalle $[a, b]$. Nous avons donc

$$s'(t) = \|u'(t)\| \tag{3.19}$$

Le repère de Frenet en un point $u(t)$ d'une courbe est donné par les vecteurs normés T tangent à u en $u(t)$ et N directement orthogonal à T. Nous avons alors

$$T = \frac{u'(t)}{\|u'(t)\|} \tag{3.20}$$

N est également défini par

$$N = \frac{dT}{d\varphi} \tag{3.21}$$

où $d\varphi$ est la variation d'angle apportée à T. D'après (3.19) et (3.20), nous avons $u'(t) = s'(t)T$. La courbure correspondant à la dérivée seconde suivant l'abscisse curviligne, nous avons

$\frac{dT}{ds} = \frac{1}{R}N$ avec R le rayon de courbure (figure 3.23). Par conséquent, nous avons

$$u''(t) = \frac{d^2u}{dt^2}(t) = \frac{d}{dt}(\frac{ds}{dt}T) \tag{3.22}$$

$$= \frac{d^2s}{dt^2}T + (\frac{ds}{dt})^2\frac{1}{R}N \tag{3.23}$$

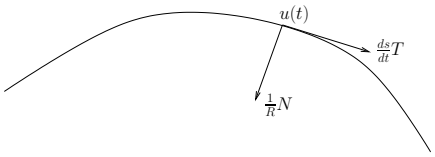

Fig. *3.23* — Dérivées première et seconde en $u(t)$.

Comme T et N sont orthogonaux et normés, nous avons $\det(T, N) = 1$ et $\det(T, T) = 0$. Nous obtenons donc

$$\det(u'(t), u''(t)) = \det(\frac{ds}{dt}T, \frac{d^2s}{dt^2}T + (\frac{ds}{dt})^2\frac{1}{R}N) \tag{3.24}$$

$$= \det(\frac{ds}{dt}T, (\frac{ds}{dt})^2\frac{1}{R}N) \tag{3.25}$$

$$= (\frac{ds}{dt})^3\frac{1}{R} = \|u'(t)\|^3\frac{1}{R} \tag{3.26}$$

R étant le rayon de courbure, nous pouvons en déduire la courbure $\kappa(t)$ d'une courbe paramétrique dans le plan :

$$\kappa(t) = \frac{\det(u'(t), u''(t))}{\|u'(t)\|^3} \tag{3.27}$$

et l'énergie interne devient :

$$E_{int} = \frac{1}{2}\left(\alpha(t)\left|\frac{du}{dt}(t)\right|^2 + \beta(t)\left|\kappa(t)\right|^2\right) \tag{3.28}$$

Méthodes de résolution Comme dit précédemment, les snakes sont utilisés pour rechercher des minima locaux. Afin d'éviter les variations d'énergie trop importantes, la résolution de (3.17) se fait par petits déplacements à l'aide de méthodes itératives. Dans le cadre de la détection de contours sur une image, un algorithme de minimisation de l'équation (3.17) est le "*greedy algorithm*" présenté dans [Williams et Shah, 1992]. Il s'agit, pour chaque point du contour de rechercher le pixel voisin pour lequel l'énergie est minimale et de déplacer le point vers ce nouveau pixel. L'algorithme est répété jusqu'à ce qu'une solution stable soit obtenue.

La solution de (3.17) peut également être obtenue par des méthodes d'optimisation classique. Dans [Pottmann et al., 2002], le snake est utilisé pour approcher des points donnés. L'énergie externe est donc le carré de la distance entre les points. Les termes étant tous quadratiques, le problème se ramène à un problème de moindres carrés.

Des méthodes variationnelles de type différences finies [Kass et al., 1987] ou éléments finis [Cohen et Cohen, 1993] sont aussi utilisées. Comme nous cherchons une solution locale, il faut éviter que le snake ne subisse de trop grandes variations. Dans le cas où la résolution se ramène à l'inversion d'un système linéaire, une composante temporelle τ est ajoutée et une énergie de dissipation est définie :

$$D(u) = \int_0^1 \frac{1}{2} |\frac{\partial u}{\partial \tau}(s)|^2 ds \tag{3.29}$$

La résolution est donc itérative et fonctionne sur le même principe que pour les méthodes mécaniques où le système est sous-relaxé : le snake est déformé jusqu'à ce que le terme dissipatif soit négligeable et qu'une position stable soit atteinte.

Les contours actifs en généralisation cartographique Les contours actifs ont été appliqués en généralisation cartographique par [Burghardt et Meier, 1997] et [Bader, 2001]. Ils ne sont pas utilisés pour modéliser les courbes mais leurs déplacements. Soient $f(t)$ la courbe initiale et $\bar{f}(t)$ la courbe après déplacement, ce déplacement est un contour actif donné par (3.30).

$$d(t) = \bar{f}(t) - f(t) \tag{3.30}$$

Dans le cas où aucune énergie extérieure n'est appliquée, l'énergie est minimale quand le déplacement est nul. L'énergie de déplacement est donc composée d'une énergie externe qui tend à déformer la courbe pour respecter les contraintes extérieures (sécurité, lisibilité) et d'une énergie interne qui tend à limiter ces déformations et à conserver l'allure initiale de la courbe (contrainte géomorphologique).

Si une courbe est à une distance suffisante de toutes ses voisines, il n'y a pas de contrainte externe et l'énergie extérieure est nulle. Sinon, une énergie externe fonction de la distance est appliquée. Dans [Burghardt et Meier, 1997], cette énergie est simplement définie en chaque point (x_i, y_i) d'une ligne par

$$E_{ext} = \begin{cases} (1 - \frac{v}{\epsilon_{vis}}) & \text{si } v < \epsilon_{vis} \\ 0 & \text{si } v \geq \epsilon_{vis} \end{cases} \tag{3.31}$$

où v est la distance entre un objet du plan et le point (x_i, y_i) .

Dans [Bader, 2001], une force agissant entre les lignes est définie. La force appliquée en un point est dirigée par le vecteur reliant ce point avec le point le plus proche de la ligne en conflit. Cette force est la même que celle définie en (3.16) pour les réseaux de barres.

Les autres contraintes imposées sont les conditions aux limites de la courbe définies en fonction des points extrêmes. En général, le déplacement et la dérivée aux extrémités du segment à déformer sont nuls pour que le raccordement soit lisse. Si l'extrémité du segment correspond à l'extrémité de la courbe, le déplacement est permis mais la dérivée est fixée pour qu'il n'y ait pas de rotation et pour conserver l'orientation de la courbe.

3.4.2 Calcul d'une solution à partir d'un contour actif

Dans sa thèse, Bader a utilisé les poutres élastiques et les contours actifs pour la généralisation de routes et de bâtiments. Les poutres représentent les routes alors que les snakes représentent leur déplacement. Dans les deux cas, la résolution se fait avec des éléments finis. Pour Bader, les poutres donnent de meilleurs résultats parce qu'il résout un seul système où sont assemblées les coordonnées de tous les points alors qu'avec les contours actifs, les déplacements sont calculés séparément pour chaque coordonnée x et y.

Dans ce paragraphe, nous présentons une méthode de correction des conflits pour la généralisation des cartes marines fondée sur les contours actifs. L'intérêt est que les paramètres de forme α et β du snake peuvent être variables alors que les paramètres des poutres (l'élasticité et l'aire de la section) sont fixes. De plus, dans la méthode de résolution que nous avons mise en place, nous ne séparons pas les coordonnées pour que le résultat soit indépendant du repère choisi.

3.4.2.1 Définition des énergies

Les énergies sont définies afin de garantir les contraintes fortes (sécurité, lisibilité) lorsque le minimum est atteint. Une solution est atteinte lorsque le déplacement d défini par (3.30) est plus grand que le déplacement minimal $d_{\min}$ donné au paragraphe 3.2.3. d est un snake défini par une courbe B-spline. $d_{\min}$ étant une polyligne, la comparaison entre les deux courbes se fait en approchant d par une polyligne toujours avec la méthode du paragraphe 2.3.3.

Les courbes représentant les isobathes ont été construites suivant des critères de compression. Les isobathes nous ont été fournies sous la forme de listes de points. Les courbes B-splines ont été construites pour approcher ces points à $\epsilon_v is$ près. La paramétrisation est différente de l'abscisse curviligne. La dérivée seconde et la courbure ne sont donc pas identiques. La méthode devant être appliquée à la correction de conflits variés, pour contrôler la forme de la courbe, il est préférable de travailler avec la courbure plutôt que la dérivée seconde. L'énergie interne est donc définie par (3.32). Au départ, tous les points de contrôle de d sont nuls. L'énergie interne initiale du snake est donc également nulle.

$$E_{int} = \frac{1}{2}\left(\alpha(t)\left|\frac{dd}{dt}(t)\right|^2 + \beta(t)\left|\kappa_d(t)\right|^2\right) \tag{3.32}$$

L'énergie externe est définie en fonction du déplacement $d_{\min}$ entre les courbes. S'il n'y a pas de conflit, les coordonnées de $d_{\min}$ sont toutes nulles. L'énergie externe et l'énergie interne sont nulles et la courbe n'est pas modifiée. Sinon, le snake est déformé pour réduire cette énergie. Si un vecteur $d_{\min}(\zeta_j)$ est non nul, l'énergie externe de d est donnée par

$$E_{ext}(d(\zeta_j)) = \begin{cases} \frac{\|d_{\min}(\zeta_j) - d(\zeta_j)\|^2}{\epsilon_{vis}^2} & \text{si } d(\zeta_j) < d_{\min}(\zeta_j) \\ 0 & \text{sinon} \end{cases} \tag{3.33}$$

De ce fait, dès que le déplacement est supérieur à $d_{\min}$, l'énergie externe du snake est nulle. Nous rappelons que d est signé afin de tenir compte des inter-pénétrations.

3.4.2.2 Résolution du système énergétique

Comme nous avons vu, nous ne calculons pas directement la position finale du snake. La solution est calculée par déplacements successifs jusqu'à atteindre une position stable. Dans notre problème, la courbure est exprimée sous forme rationnelle. Nous ne pouvons donc pas nous ramener à la minimisation d'une fonction quadratique par des moindres carrés. Nous utilisons la méthode du gradient où les inconnues sont les coordonnées des points de contrôle. Le principe consiste à calculer à chaque étape une direction de descente nous donnant une énergie plus petite. Le choix de cette méthode est justifiée dans le paragraphe 3.4.2.3. La direction est donnée par l'opposé du gradient. Le pas de descente est calculé par la méthode de la section dorée.

Nous notons $d^{(n)}$ le snake calculé à l'itération n. Pour calculer la solution à l'itération suivante, nous avons besoin de calculer son énergie. Pour cela, $d^{(n)}$ est approché par une ligne polygonale. L'énergie externe est calculée à partir de (3.33). La dérivée première est approchée par des différences finies. La courbure est obtenue soit par différences finies en approchant les dérivées première et seconde, soit en calculant le rayon du cercle osculateur en chaque ζ_j. Dans [Delingette, 1994], la courbure en un point $d^{(n)}(\zeta_i)$ est estimée par l'inverse du rayon du cercle passant par ce point et ses deux voisins. Elle est donnée par la formule.

$$\kappa_d(\zeta_i) = \frac{\sin(\varphi_i)}{\frac{1}{2}\|d^{(n)}(\zeta_{i+1}) - d^{(n)}(\zeta_{i-1})\|} \tag{3.34}$$

où φ_i est l'angle formé par les points $P_{i-1} = d^{(n)}(\zeta_{i-1})$, $P_i = d^{(n)}(\zeta_i)$ et $P_{i+1} = d^{(n)}(\zeta_{i+1})$ (figure 3.24).

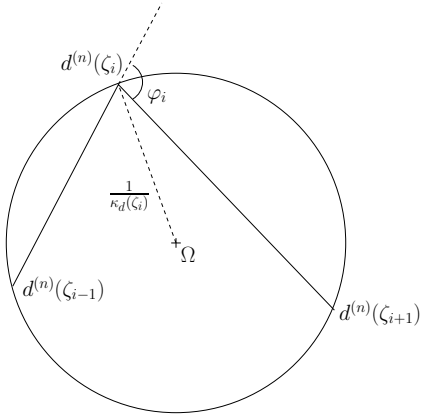

Fig. *3.24* — Définition de la courbure d'après [Delingette, 1994].

Ω est le centre du cercle circonscrit aux trois points. Si ce cercle est une approximation du cercle osculateur, alors la normale N en P_i est donnée par le rayon joignant P_i à Ω. Dans certains cas, notamment si l'angle φ est grand, Ω est situé à l'extérieur du secteur formé par P_{i-1}, P_i et P_{i+1}. La normale en P_i est donc mal approchée et la courbure calculée est loin de la courbure réelle. L'erreur vient du fait que la courbure calculée en (3.34) dépend

de la longueur des segments. En effet, d'après l'équation (3.21), N est indépendante de la longueur des segments puisque T est un vecteur unitaire. Une solution est donc de prendre deux segments de même longueur l constante. Pour cela, nous construisons deux points $\hat{P}_{i-1}$ et $\hat{P}_{i+1}$ tels que $\hat{P}_{i-1}$ appartienne à la droite $P_{i-1}P_i$ et $\hat{P}_{i+1}$ appartienne à la droite P_iP_{i+1} avec $\|\hat{P}_{i-1}P_i\| = \|P_i\hat{P}_{i+1}\| = l$. N sera alors toujours dirigée suivant la bissectrice de l'angle $\hat{P}_{i-1}P_i\hat{P}_{i+1}$. Le rayon de courbure est alors donné par le rayon du cercle passant par les points $\hat{P}_{i-1}$, P_i et $\hat{P}_{i+1}$ (figure 3.25). La courbure $\kappa_d(\zeta_i)$ vaut alors :

$$\kappa_d(\zeta_i) = \frac{\sin(\varphi_i)}{\frac{1}{2}\|\hat{P}_{i+1} - \hat{P}_{i-1}\|} \tag{3.35}$$

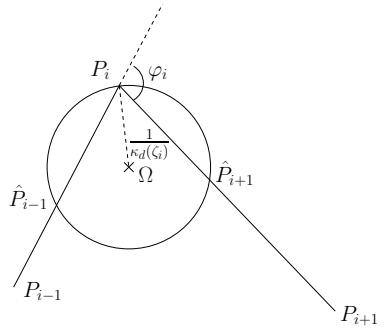

Fig. *3.25* — Définition de la courbure indépendamment de la longueur des segments.

Ces énergies sont assemblées dans un système d'équations correspondant à (3.17). Les inconnues du système sont les points de contrôle de $d^{(n)}$. La méthode est itérée jusqu'à ce qu'une solution stable soit atteinte, c'est-à-dire, lorsque l'énergie du système est minimale et que la position de la courbe est stable.

Le dernier point à prendre en compte est le réglage des paramètres. En effet, il faut s'assurer que le snake converge vers une solution valide en réglant correctement les paramètres de forme.

3.4.2.3 Réglage automatique des paramètres

Lorsque la courbe est déformée, l'énergie externe diminue et l'énergie interne augmente. Si nous ne tenons pas compte de l'énergie interne, nous savons que le conflit est corrigé dès que l'énergie externe est nulle. Sinon, il faut s'assurer que lorsqu'une position stable est atteinte, les contraintes de sécurité et de lisibilité sont vérifiées tout le long de la courbe. En effet, si l'énergie interne est plus importante que l'énergie externe, le snake sera peu déformé afin de préserver la forme initiale et la courbe restera proche de la courbe de départ. Le déplacement ne sera pas suffisant pour corriger le conflit. Pour cela, il faut que le terme d'énergie externe (3.33) soit prépondérant devant le terme d'énergie interne (3.28).

Nous disposons de deux paramètres α et β réglant le rapport entre la tension et la courbure

du déplacement. Le réglage de ces paramètres permet de choisir l'importance donnée aux critères de forme. Si nous posons $\beta = 0$, nous ne tenons pas compte de la courbure. La solution obtenue sera proche de l_{min}, par contre, les normes $\|\kappa_d\|_2$ et $\|\kappa_d\|_{max}$ sont plus grandes. Cette remarque est illustrée par l'exemple de la figure 3.26. A gauche, nous avons choisi $\alpha = 1$ et $\beta = 0$. A droite, les paramètres sont $\alpha = 0$, $\beta = 1$. Sur la deuxième figure, la forme de la courbe est mieux préservée mais le déplacement est plus important. Les résultats sont donnés dans le tableau 3.3.

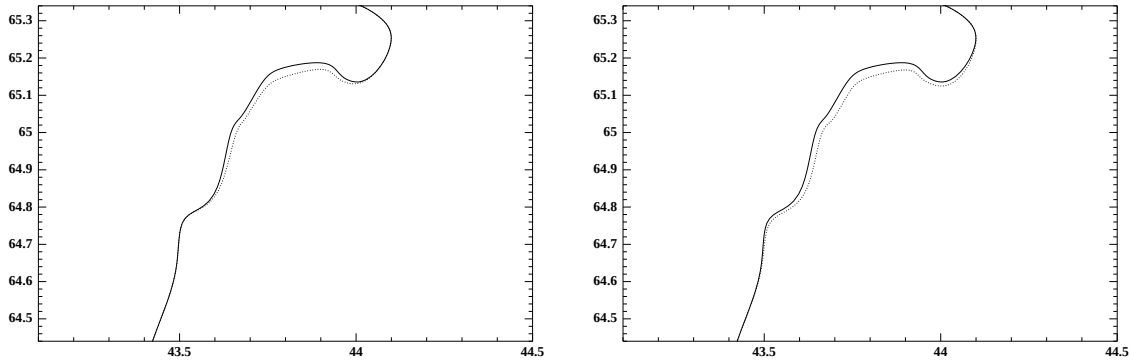

Fig. ***3.26*** — Corrections obtenues pour différentes valeurs de paramètres : courbe originale (trait plein) et courbe déformée (pointillés). A droite, la forme est mieux préservée.

	$\|d\|_2$	$\|d\|_{\max}$	$\|\kappa_d\|_2$	$\|\kappa_d\|_{\max}$
$\alpha = 1$, $\beta = 0$	0,037	0,024	0,779	1,661
$\alpha = 0$, $\beta = 1$	0,048	0,025	0,650	0,652

Tab. ***3.3*** — Normes des déplacements et des courbures pour chaque correction.

De plus, différentes valeurs de paramètres ont été testés sur d'autres exemples et nous remarquons que la valeur de α a peu d'influence sur le résultat final. La prise en compte de la contrainte géomorphologique dépend surtout de β. Par conséquent et afin de simplifier le problème, nous pouvons fixer $\alpha = 0$.

La préservation de la forme dépend uniquement de β. Si β est petit, le conflit est corrigé sans tenir compte de la forme, sinon, le snake est lissé. Le déplacement est alors plus régulier pour éviter les variations. Le problème est que nous ne pouvons pas prendre β aussi grand que l'on veut. Si nous donnons plus d'importance à l'énergie interne qu'à l'énergie externe, la courbe n'est pas déplacée puisque son énergie minimale correspond à sa position de départ où l'énergie interne est nulle. β doit donc être suffisamment petit pour que E_{ext} soit grand devant E_{int}. De ce fait, la solution d'équilibre est atteinte lorsque l'énergie externe est nulle, c'est-à-dire lorsque le conflit est corrigé.

Dans la littérature, les paramètres de forme sont généralement constants et fixés par l'utilisateur. Le problème est que α et β sont propres à chaque courbe. Dans le cadre de

la généralisation d'une carte, le cartographe doit intervenir uniquement pour des actions de haut niveau. Il peut choisir l'opérateur de correction mais l'application de cet opérateur doit être entièrement automatique. Comme β ne peut pas être le même pour toutes les courbes, il importe de le définir automatiquement.

Le réglage automatique des coefficients a été traité dans [Larsen et al., 1995]. Les auteurs définissent α et β de sorte que le snake reste toujours dans le voisinage d'un contour donné. Soient deux snakes u_1 et u_0 avec u_1 à une distance ϵ de u_0. Les valeurs maximales pour lesquelles un snake reste à une distance inférieure à ϵ sont données en fonction de l'abscisse curviligne s par :

$$\alpha(s) = \frac{E^1_{ext}(s) - E^0_{ext}(s)}{2(\|u'_0(s)\|^2 - \|u'_1(s)\|^2)} \quad , \qquad \beta(s) = \frac{E^1_{ext}(s) - E^0_{ext}(s)}{2(\|u''_0(s)\|^2 - \|u''_1(s)\|^2)}. \tag{3.36}$$

Dans notre cas, nous ne cherchons pas à limiter le déplacement mais à choisir une direction de déplacement corrigeant le conflit. A chaque étape, la direction de déplacement est donnée par le gradient d'énergie. Le problème revient donc à prendre β tel que le gradient soit bien orienté. L'énergie du snake est donnée par :

$$E_{snake} = \int_a^b \frac{1}{2}\beta(t)|\kappa(t)|^2 + E_{ext}(t)dt \tag{3.37}$$

Si

$$\delta(t) = \begin{cases} \frac{d(t) - d_{\min}(t)}{\epsilon_{vis}} & \text{si } d(t) < d_{\min}(t) \\ 0 & \text{sinon} \end{cases} \tag{3.38}$$

le gradient est donné par

$$\nabla(\frac{1}{2}\beta(t)|\kappa(t)|^2 + E_{ext}(t)) = \beta(t)|\kappa(t)|\nabla\kappa(t) + \delta(t)\nabla\delta(t) \tag{3.39}$$

Le snake est déformé dans la direction opposée du gradient (3.39). Pour que le snake soit attiré vers une solution admissible, il faut que le gradient d'énergie externe soit plus important que le gradient d'énergie interne. Pour cela, il faut que $\beta(t)$ soit plus petit que

$$\beta_{max}(t) = \frac{\|\delta(t)\|}{|\kappa(t)|}. \tag{3.40}$$

De ce fait, dès que le conflit est corrigé, l'énergie externe est nulle ainsi que β. Les deux énergies sont donc nulles et une solution stable est atteinte.

Dans le paragraphe suivant, nous donnons les résultats obtenus pour cette méthode et les comparons avec la méthode précédente.

3.4.3 Comparaison des méthodes

Les deux méthodes ont été appliquées sur un grand nombre de conflits détectés à partir des données utilisées dans le chapitre 2. Les exemples présentés dans ce chapitre sont des exemples caractéristiques mettant en évidence les avantages et les inconvénients de chaque

méthode. Il s'agit en fait de conflits complexes où les déplacements à effectuer sont relativement importants puisque nous avons des recouvrements ou des intersections franches. Dans la plupart des cas que nous avons traités, il s'agit d'intersections visuelles et les différences entre les deux méthodes sont beaucoup moins marquées aussi bien en temps qu'en qualité du résultat obtenu.

Nous présentons d'abord les résultats obtenus pour les deux conflits corrigés avec le réseau de barres dans le tableau 3.4. Sur la figure 3.27, nous donnons les courbes obtenues par les deux méthodes.

	$\|d\|_2$	$\|d\|_{\max}$	$\|\kappa_d\|_2$	$\|\kappa_d\|_{\max}$
Conflit 1 (barres)	0,054	0,035	1,008	1,927
Conflit 1 (snake)	0,042	0,030	0,701	1,643
Conflit 2 (barres)	0,055	0,021	1,103	1,932
Conflit 2 (snake)	0,048	0,021	0,733	1,722

Tab. *3.4* — Normes des déplacements et des courbures pour chaque conflit.

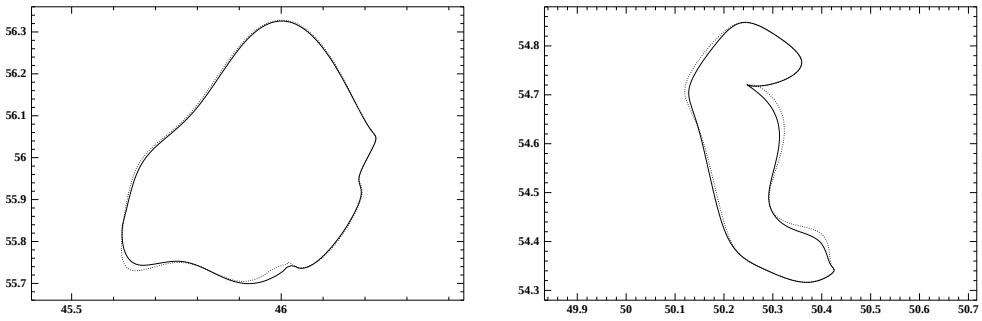

Fig. *3.27* — Solutions obtenues par contours actifs (traits pleins) et déformation mécanique (tirets).

Nous voyons que les contours actifs nous donnent de meilleurs résultats dans tous les cas. Avec les réseaux de barres, la déformation se fait en imposant les déplacements ou les forces aux points de contrôle. En particulier, les directions sont fixées puis les intensités sont calculées afin de donner une solution admissible. Les déformations sont relativement rigides puisque lorsqu'un point de la courbe ne peut plus être déplacé, les points de contrôle liés à ce point sont tous fixés. Les déformations sont globales.

La deuxième méthode, fondée sur les modèles déformables, consiste à réduire l'énergie le long de la courbe et nous ramène à un problème de minimisation où les inconnues sont les points de contrôle. Cette méthode est beaucoup plus souple puisque aucune direction n'est imposée pour les déplacements. Cela permet de faire de plus petits déplacements et de corriger les conflits localement. Par exemple, pour le deuxième conflit, l'auto-intersection qui était créée avec la méthode mécanique n'apparaît plus (figure 3.28).

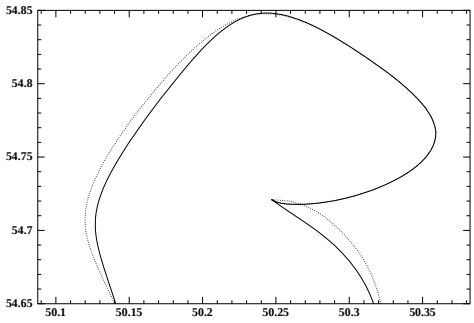

Fig. ***3.28*** — Solution mécanique : la déformation globale entraine une auto-intersection. Contour actif : la déformation étant locale, la courbe n'est pas modifiée.

Dans la méthode de contours actifs, nous avons également introduit une contrainte de forme. Nous voyons en effet dans le tableau 3.4 que la courbure du déplacement est plus petite avec les contours actifs. La forme de la courbe initiale est donc mieux préservée.

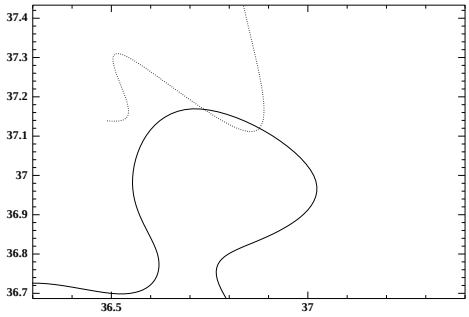

Fig. ***3.29*** — Intersection entre deux isobathes.

Cependant, ceci n'est pas toujours vrai. Pour l'exemple de conflit de la figure 3.29, nous donnons les résultats obtenus dans le tableau 3.5 et sur la figure 3.30. Il apparaît clairement que le déplacement est plus important avec un réseau de barres mais que la forme est mieux préservée. Sur cet exemple, le contour actif ne préserve pas la forme parce que la déformation est trop locale. Le déplacement est beaucoup moins important mais la déformation ne se fait qu'autour de la zone de conflit alors que, avec le réseau de barres, le déplacement total est plus important mais la déformation est plus régulière. La norme du déplacement effectué est donnée sur la figure 3.31. Nous voyons que pour le snake, le déplacement varie beaucoup plus rapidement que pour le réseau de barres.

Dans la plupart des cas traités avec les deux méthodes, les contours actifs nous ont donné de meilleurs résultats que les réseaux de barres. La distance moyenne est toujours plus petite. Par contre, pour certains conflits comme celui de la figure 3.29, les réseaux de barres préservent mieux la forme. Ces cas restent marginaux. Néanmoins, ils mettent en évidence le caractère local des contours actifs par rapport aux réseaux de barres qui proposent des

	$\|d\|_2$	$\|d\|_{\max}$	$\|\kappa_d\|_2$	$\|\kappa_d\|_{\max}$
Déplacement minimal	0,022	0,044	-	-
Réseau de barres	0,070	0,044	0,296	0,360
Contours actifs	0,050	0,047	0,552	1,051

Tab. *3.5* — Norme des déplacements et des courbures.

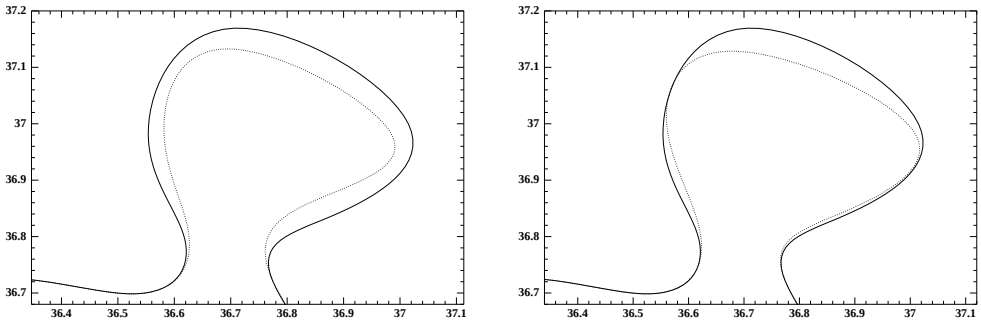

Fig. *3.30* — Courbe initiale (trait plein) et corrections obtenues avec un réseau de barres (à gauche) et un contour actif (à droite).

déformations plus globales.

Enfin, le principal intérêt des réseaux de barres est leur rapidité. Le calcul de la solution mécanique se fait par résolution d'un système linéaire à $2(nl + 1)$ inconnues à chaque itération, nl étant le nombre de points de contrôle libres. La déformation d'un snake demande généralement moins d'itérations mais est beaucoup plus coûteuse puisqu'il faut calculer le gradient d'énergie à chaque itération. Les rapports de temps et le nombre d'itérations sont donnés dans le tableau 3.6. Comme nous avons dit, les méthodes ont été testées sur d'autres conflits. Les exemples donnés correspondent à des cas extrêmes. Dans l'ensemble, le calcul d'une solution avec les réseaux de barres est trois fois plus rapide qu'avec les contours actifs.

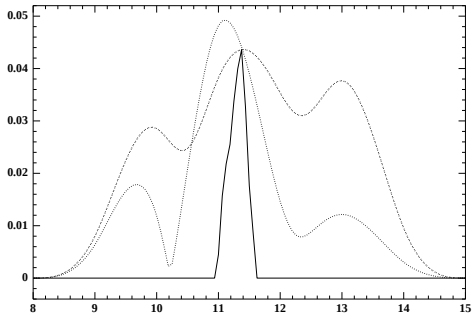

Fig. *3.31* — Norme des déplacements effectués : déplacement minimal (trait plein), réseau de barres (tirets), contour actif (pointillés).

	Itérations		Ratio
	snake	barres	snake/réseau
Conflit 1	18	33	9,61
Conflit 2	3	73	1,05
Conflit 3	15	12	8,93

Tab. *3.6* — Nombre d'itérations et ratio de temps.

Malgré cela, les contours actifs semblent plus intéressants pour la généralisation de courbes étant donné qu'ils fournissent généralement des résultats meilleurs aussi bien pour la distance que pour la courbure. L'approche par modèles déformables est également plus simple à utiliser puisque les contraintes sont exprimées sous forme d'énergies. L'utilisateur peut facilement modifier ou ajouter des contraintes. Les réseaux de barres offrent moins de possibilités puisque les directions des forces sont imposées. Cependant, il est aussi possible d'introduire des contraintes de forme ou de position dans un réseau de barres [Pernot et al., 2003]. Les auteurs se ramènent alors à un problème de minimisation similaire à la méthode de Harrie présentée au paragraphe 3.3.1.1. Les contraintes sont exprimées sous forme d'équations linéaires où les inconnues sont les forces ou les coordonnées des points de contrôle.

Enfin, dans notre problème, nous n'avons pas tenu compte des autres courbes dans le voisinage. Le déplacement d'une courbe peut être limité afin de ne pas créer un nouveau conflit avec une courbe voisine. A partir de la structure hiérarchique définie au chapitre 2, nous connaissons le voisinage du conflit. En appliquant une méthode similaire au calcul de l_{min}, nous pouvons déterminer une polyligne l_{max} majorant le déplacement. Nous disposerions alors d'une zone définissant un déplacement minimal et un déplacement maximal dans laquelle devrait se trouver la courbe corrigée. Pour cela, les contours actifs conviennent mieux car ils permettent plus de déformations que les réseaux de barres. L'énergie externe en un point serait nulle lorsque le point est dans la zone et croîtrait très rapidement à l'extérieur.

3.5 Correction des auto-intersections

3.5.1 Faisabilité de la correction

L'opérateur de déplacement que nous avons mis en place peut aussi être utilisé pour la correction des auto-intersections. Cependant, cet opérateur ne permet pas de corriger tous les conflits. Par exemple, sur la figure 3.32, le conflit à gauche peut être corrigé de deux façons en fonction de la profondeur. Si la profondeur à l'extérieur est plus importante, la courbe est déformée pour élargir le sommet. Sinon, les segments en conflit doivent être supprimés. Seul le premier cas peut être traité avec notre opérateur de déplacement.

Dans le chapitre 1, nous avons vu les différents types d'auto-intersection. D'un côté, nous avons des auto-intersections franches qui sont en fait des artefacts dus à des problèmes

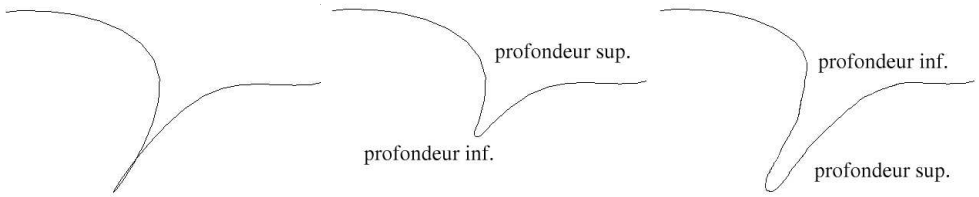

Fig. *3.32* — Généralisation d'une auto-intersection en fonction de la profondeur. A gauche : courbe originale, au centre : suppression de la boucle, à droite : élargissement.

numériques. Ces intersections ne peuvent pas être corrigées par déplacement puisqu'elles forment des boucles qui doivent être supprimées. De l'autre côté, nous avons des intersections visuelles et singulières qui peuvent être corrigées soit par déplacement, soit par suppression. Nous nous intéressons ici uniquement aux corrections par déplacement.

Le cas le plus simple est lorsque les deux segments en conflit sont définis sur des intervalles paramétriques $[t_{i_{min}}, t_{i_{max}}]$ et $[t_{j_{min}}, t_{j_{max}}]$ différents. Les points de contrôle à déplacer ne sont pas les mêmes pour chaque segment et les méthodes précédentes peuvent être appliquées. Si le conflit ne peut pas être défini sur deux segments disjoints, la méthode n'est plus valable puisque nous ne pouvons plus calculer un déplacement minimal de la même façon. Nous présentons donc dans le paragraphe suivant une définition du déplacement minimal à effectuer ainsi que les modifications à apporter à la méthode de déformation.

3.5.2 Correction d'une auto-intersection par déformation

3.5.2.1 Calcul du déplacement minimal

Le calcul des distances est fondé sur le critère 1. Le segment de courbe B-spline f est approché par une ligne polygonale s. Les points de s sont notés P_j. Un point P_{j_0} est en conflit avec un point P_j si la distance $\|P_{j_0}P_j\|$ est inférieure à ϵ_{vis} et si l'arc entre les deux points revient sur lui-même, c'est-à-dire, si les segments entre P_{j_0} et P_j forment un angle supérieur à π. Parmi les points en conflit avec P_{j_0}, le déplacement minimal ΔP_{j_0} à effectuer au point P_{j_0} est donné par le point P_{j_1} le plus proche de P_{j_0}. Il est orienté dans la direction du vecteur $\overrightarrow{P_{j_1}P_{j_0}}$ et de longueur $\epsilon_{vis} - \|P_{j_0}P_{j_1}\|$ (figure 3.33) sauf dans le cas où l'intersection est tangentielle, le vecteur étant alors orienté suivant la normale à la courbe. Ce principe peut être appliqué à tous les points de s définissant ainsi une suite de vecteurs d_{min} comme dans le paragraphe 3.2.3.

Deux exemples de déplacement minimal sont donnés sur la figure 3.34. Notons que le déplacement minimal est calculé pour chaque point indépendamment des autres déplacements. Par exemple, sur la figure 3.33, nous avons $\Delta P_{j_0} = -\Delta P_{j_1}$ et $\|\Delta P_{j_0}\| = \|\Delta P_{j_1}\| = \epsilon_{vis} - d$. Si chaque point est déplacé du vecteur correspondant, nous avons, après correction, $\|P_{j_0}P_{j_1}\| = \epsilon_{vis} + \|\Delta P_{j_0}\|$. Un déplacement plus petit pourrait être effectué. Ce-

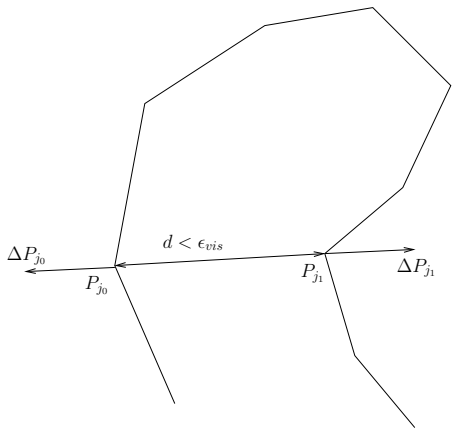

Fig. *3.33* — Calcul du déplacement minimal entre deux points.

pendant, la solution n'est pas unique puisque les points ne sont pas fixés, contrairement aux cas d'intersection traités précédemment où seule une des deux courbes est déformée.

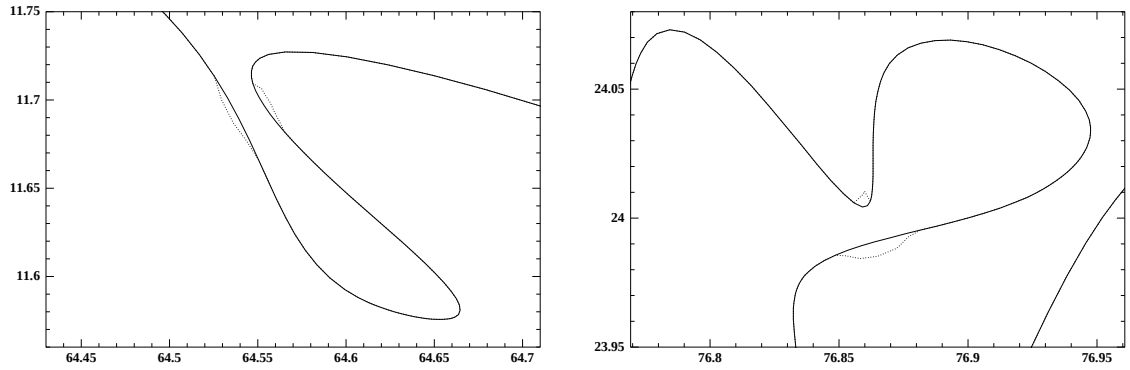

Fig. *3.34* — Déplacement minimal d'une courbe admettant une auto-intersection : courbe initiale en trait plein, déplacement minimal en pointillés.

3.5.2.2 Déformation de la courbe

Comme pour les intersections, les déplacements sont effectués en déplaçant les points de contrôle. La méthode géométrique que nous avons présentée au paragraphe 3.3.2 donne de mauvais résultats car les déformations à effectuer sont trop locales. Les corrections sont faites à l'aide des contours actifs. La méthode du paragraphe 3.4.2 peut être appliquée directement avec le déplacement minimal défini ci-dessus.

La solution obtenue est valide mais, comme nous avons vu, le déplacement minimal est plus grand que nécessaire. Un déplacement plus petit peut être défini. Pour cela, nous proposons de recalculer le déplacement minimal à chaque itération. Le deuxième intérêt est que, lorsque deux segments sont en conflit, ils sont tous les deux déplacés. En recalculant le déplacement

minimal à chaque étape, les deux segments peuvent être déplacés différemment et l'algorithme est arrêté lorsque la distance de lisibilité est garantie entre les segments. Sur la figure 3.35, nous donnons le résultat obtenu pour les conflits de la figure 3.34.

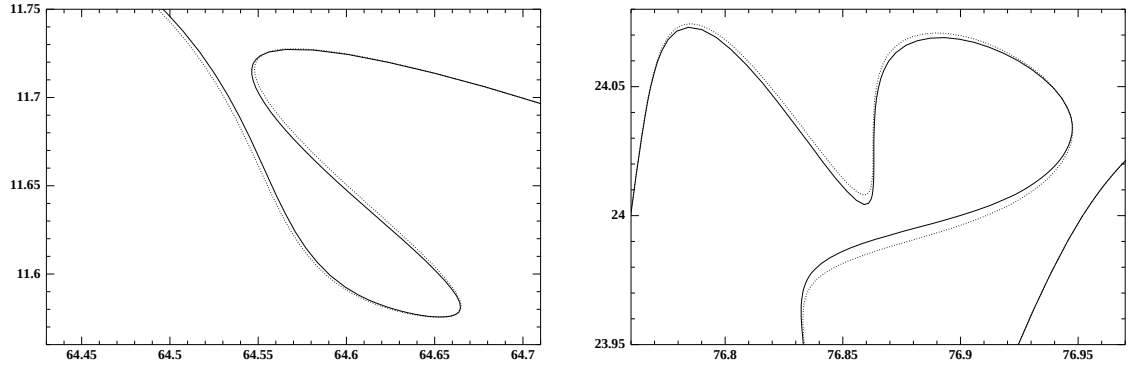

Fig. *3.35* — Correction d'une auto-intersection par déplacement : courbe initiale en trait plein, correction en pointillés.

3.6 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la correction des intersections entre deux isobathes par des méthodes de déformation. Nous avons d'abord présenté les stratégies utilisées en généralisation pour corriger les conflits entre isobathes. Nous avons également expliqué comment les contraintes cartographiques sont prises en compte. Notamment, la contrainte de sécurité nous oblige à déformer uniquement l'isobathe la plus profonde. La contrainte de lisibilité se traduit par la définition d'une ligne de déplacement minimal garantissant la résolution du conflit. Enfin, nous avons donné des critères de forme afin de mesurer les déformations effectuées. Pour cela, nous avons mesuré la norme du déplacement effectué et la courbure de ce déplacement.

Ensuite, nous nous sommes intéressés aux méthodes géométriques de généralisation de lignes et aux méthodes de déformation de courbes B-splines. Le problème de la plupart des méthodes existantes est qu'elles procèdent par minimisation des erreurs ou des conflits et ne permettent pas de garantir des contraintes comme les contraintes de sécurité et de lisibilité. Nous avons alors présenté une première méthode de correction composée de deux étapes. Nous calculons une première solution géométriquement par déplacement des points de contrôle puis améliorons ce résultat en traitant le polygone de contrôle comme un réseau de barres soumis à des forces. La courbe est alors déformée itérativement en modifiant les forces et en fixant progressivement les points de contrôle. Pour cela, la méthode nécessite la résolution d'un système linéaire à chaque itération.

Dans le paragraphe suivant, nous avons présenté une deuxième sorte de méthodes de correction. Il s'agit des modèles déformables consistant à associer notre problème d'intersection à

un problème physique de minimisation d'énergies. La contrainte de lisibilité est exprimée par des énergies externes déformant la courbe alors que la contrainte de forme est exprimée par une énergie interne tendant à limiter les déformations. Le système est alors déformé jusqu'à atteindre une position où l'énergie est minimale. Les principaux modèles rencontrés dans la littérature sont un modèle de poutres où les courbes sont assimilées à des poutres soumises à des tractions et des flexions et une méthode de contours actifs où le déplacement de la courbe est considéré comme un modèle déformable soumis à des énergies de tension et de courbure.

La méthode des réseaux de barres nous donne des solutions valides mais qui sont parfois à une distance importante de la courbe initiale. De plus, même si la forme est bien préservée, nous n'avons pas tenu compte du critère de conservation de la forme de la courbe. Nous avons alors présenté une deuxième méthode de correction fondée sur les contours actifs. L'énergie externe est fonction du déplacement minimal à effectuer pour corriger le conflit alors que l'énergie interne est donnée par la courbure du déplacement calculée de manière discrète le long de la courbe. Nous disposons de paramètres de forme permettant de régler le rapport entre les différentes énergies. Dans notre cas, l'énergie externe doit toujours être prépondérante afin de garantir la correction du conflit. Nous avons donc introduit une méthode réglant automatiquement le paramètre de forme, assurant ainsi la convergence de la méthode.

Les deux méthodes ont été testées et comparées sur des conflits détectés avec la méthode présentée dans le chapitre précédent. Les réseaux de barres sont plus rapides car ils demandent moins de calculs mais ils donnent de moins bons résultats que les contours actifs : en général, les déplacements effectués avec les snakes sont plus petits et les formes sont mieux préservées.

Nous avons également étudié le traitement des auto-intersections. Les courbes peuvent être déformées pour corriger les auto-intersections visuelles ou singulières. Pour cela, un nouveau déplacement minimal est défini et le modèle de cotours actifs précédent est appliqué.

Les contours actifs sont plus souples que les réseaux de barres et permettent d'ajouter facilement de nouvelles contraintes comme un déplacement maximal pour intégrer d'autres objets situés dans le voisinage. L'utilisation des contours actifs pourraient également être étendue à la gestion globale de conflits plus complexes où plusieurs courbes seraient déformées dans un même processus.

CHAPITRE 4

Exemple récapitulatif

4.1 Objectif

Dans ce chapitre, nous appliquons notre méthode de détection et de correction des conflits à des ensembles d'isobathes construites à partir de relevés bathymétriques. L'intérêt est de comparer nos résultats avec les résultats provenant de la généralisation manuelle des cartes par les cartographes du SHOM. Les isobathes obtenues par traitement manuel correspondent aux courbes figurant sur la carte finale où toutes les données sont prises en compte (sondes, traits de côte, ...).

Dans notre méthode, la stratégie de généralisation n'est pas prise en compte puisque nous ne disposons que des isobathes et du seul opérateur de déplacement. De plus, les corrections sont effectuées localement : un conflit est défini entre deux courbes et le voisinage n'est pas pris en compte lors de la déformation. Comme nous avons vu dans le chapitre précédent, la correction se fait en modifiant uniquement la courbe la plus profonde pour respecter la contrainte de sécurité.

L'environnement d'une courbe n'étant pas considéré, il est possible qu'en corrigeant un conflit, un autre conflit avec une courbe voisine soit créé. Afin de limiter ce problème, la correction se fait en partant des isobathes les moins profondes. Ainsi, si une isobathe est en conflit avec une isobathe moins profonde et une isobathe plus profonde, l'isobathe est d'abord déformée pour corriger le premier conflit puis l'isobathe plus profonde est déformée pour corriger le second conflit. De ce fait, les modifications apportées pour la correction du premier conflit sont prises en compte pour la correction du second conflit.

Nous présentons deux exemples où une zone complète est traitée. Ces zones sont délimitées par les rectangles sur les cartes de la figure 4.1. Nous partons des isobathes construites à partir des points de sondes et représentées par des polylignes. Ces données sont fournies par le SHOM. Les courbes B-splines sont construites par compression avec une précision inférieure à 10^{-2}. Dans certains cas, lorsque la géométrie des courbes est complexe, il est possible que la précision voulue ne soit pas atteinte et que de nouveaux conflits soient créés. Ensuite, la méthode du chapitre 2 est appliquée pour détecter les conflits. La correction se fait à l'aide des contours actifs.

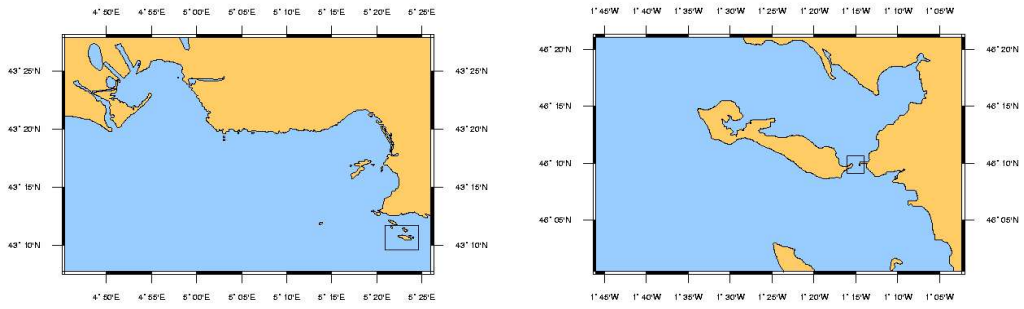

Fig. *4.1* — Extraits des cartes traitées 6767 à gauche et 7404 à droite.

4.2 Premier exemple

Nous présentons un premier exemple extrait de la carte 6767. Sur la figure 4.4 sont tracées les isobathes avant généralisation, représentées par des polylignes. Après construction des courbes B-splines et traitement des conflits, nous obtenons le résultat de la figure 4.5. 90 intersections ont été détectées. Même si l'approche est locale, certains conflits globaux entre plusieurs courbes ont été corrigés. Par exemple, dans les zones où plusieurs isobathes sont imbriquées, correspondant à des sommets, le déplacement des isobathes au centre est pris en compte lors du déplacement des isobathes à l'extérieur. En revanche, dans certains cas, de nouveaux conflits sont créés.

Certains problèmes sont dus également à une mauvaise définition des courbes. Sur la figure 4.2 à gauche, nous avons agrandi un extrait de la carte précédente. Nous voyons qu'il y a des intersections entre les isobathes qui sont liées à la définition des courbes sous forme de listes de points (le phénomène d'arcs brisés apparaît clairement). Ces problèmes sont accentués lors de la construction des courbes B-splines (figure 4.2 à droite) : dans certains cas, les données sont mal conditionnées et des oscillations apparaissent amplifiant les erreurs précédentes. Les déplacements à effectuer pour corriger le conflit sont alors importants. Comme nous l'avons signalé, les auto-intersections ne sont pas toutes prises en compte et des conflits pouvant apparaître lors de la correction restent à corriger (figure 4.3).

Nous avons affiché les isobathes obtenues après généralisation manuelle sur la figure 4.6 et l'extrait de la carte finale sur la figure 4.7. La généralisation manuelle se fait en tenant compte également des autres objets de la carte, notamment des sondes. De ce fait, certaines courbes sont agrégées ou supprimées pour avoir moins d'isobathes. Des courbes peuvent également être déplacées pour avoir suffisamment de place pour marquer les sondes. Nous voyons que les courbes obtenues après correction sont proches des isobathes de la carte. Lors de la généralisation manuelle, les isobathes sont lissées et les formes sont moins bien conservées que dans notre cas où nous avons toujours le même nombre de points mais le principal intérêt des méthodes de généralisation automatique est que les contraintes sont mieux respectées. En effet, si l'on compare avec les courbes initiales, la contrainte de sécurité n'est pas toujours

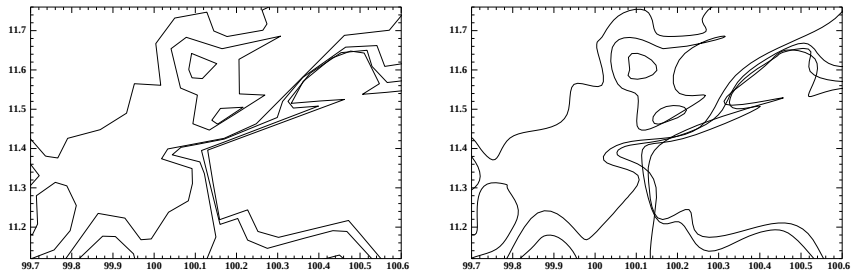

Fig. *4.2* — Définition des isobathes : à gauche polylignes, à droite B-splines obtenues par compression.

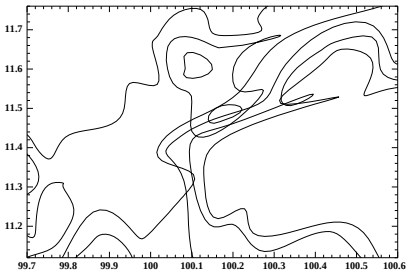

Fig. *4.3* — Correction par déplacement inadaptée : les courbes déplacées entrent en conflit avec d'autres courbes du voisinage.

respectée lorsque les courbes sont généralisées manuellement.

4.3 Deuxième exemple

Nous présentons sur la figure 4.8 un deuxième exemple extrait de la carte 7404. Nous avons détecté 41 intersections. Le résultat est donné sur la figure 4.9. Certains conflits ne pouvant pas être corrigés par déplacement n'ont pas été traités.

Comme dans le cas précédent, des conflits, notamment des auto-intersections ont été créés mais les intersections visuelles et les recouvrements entre les isobathes au centre de la figure ont bien été corrigées. Sur la figure 4.10, nous avons agrandi la partie centrale. En traitant ls conflits dans le sens des profondeurs croissantes, nous avons pu corriger les intersections entre toutes les courbes : la déformation appliquée à une courbe est bien prise en compte pour la correction du conflit suivant. Cependant, étant donné qu'il y avait quatre courbes en conflit, les déplacements appliqués aux courbes sont de plus en plus grands. De ce fait, la dernière courbe subit des déformations importantes. La forme de la courbe n'est donc pas préservée et des auto-intersections sont créées. De plus, en haut de la figure, nous voyons que certaines courbes, situées à gauche de l'isobathe fermée avant la correction sont passées à droite.

Les résultats obtenus sont comparés avec les données finales apparaissant sur les cartes

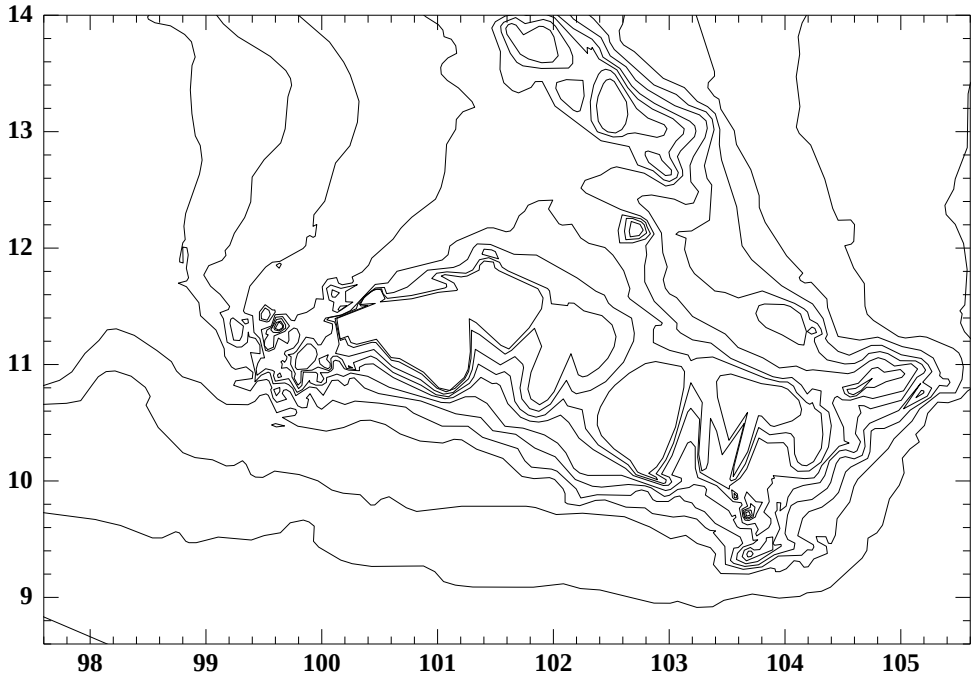

Fig. *4.4* — Extrait de la carte 6767 du SHOM : isobathes avant généralisation.

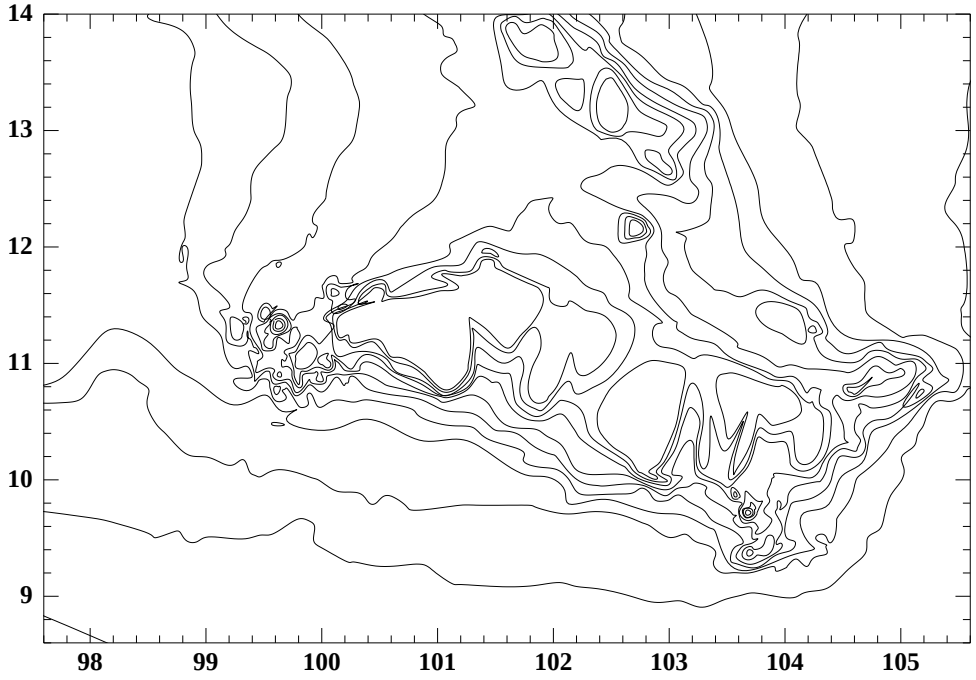

Fig. *4.5* — Traitement par déplacement des isobathes.

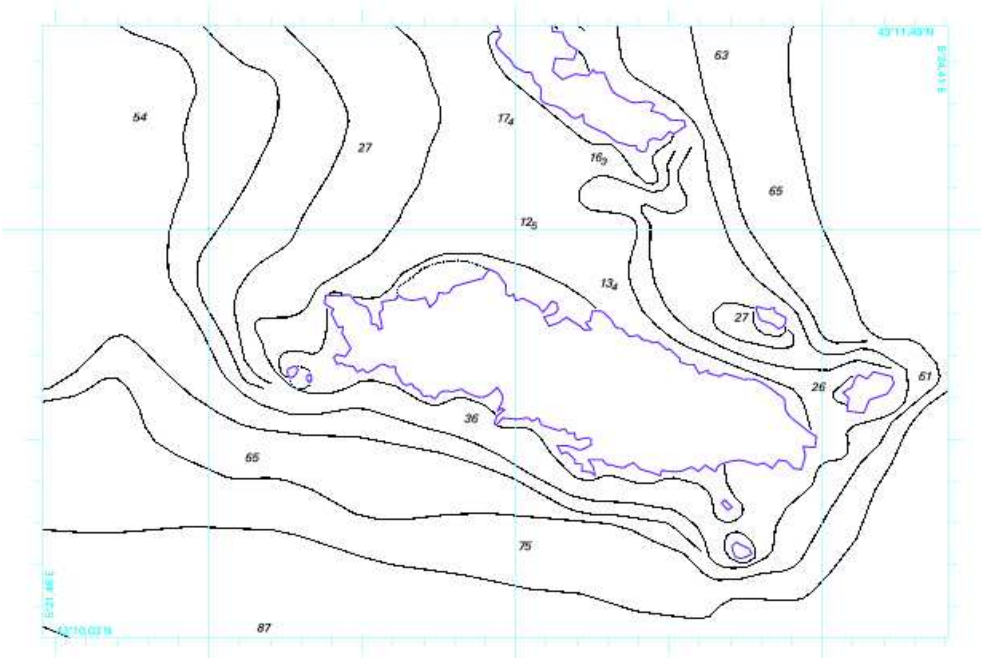

Fig. *4.6* — Isobathes après généralisation manuelle.

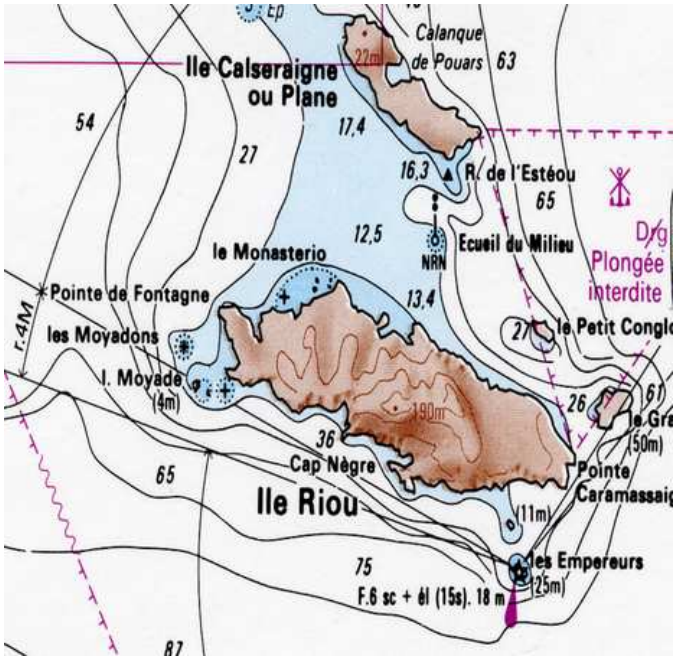

Fig. *4.7* — Extrait de la carte finale 6767 du SHOM.

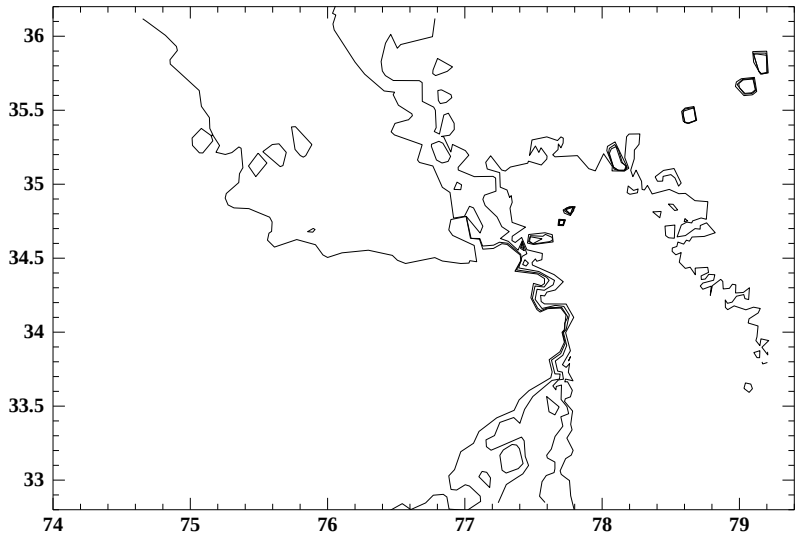

Fig. *4.8* — Extrait de la carte 7404 du SHOM : isobathes avant généralisation.

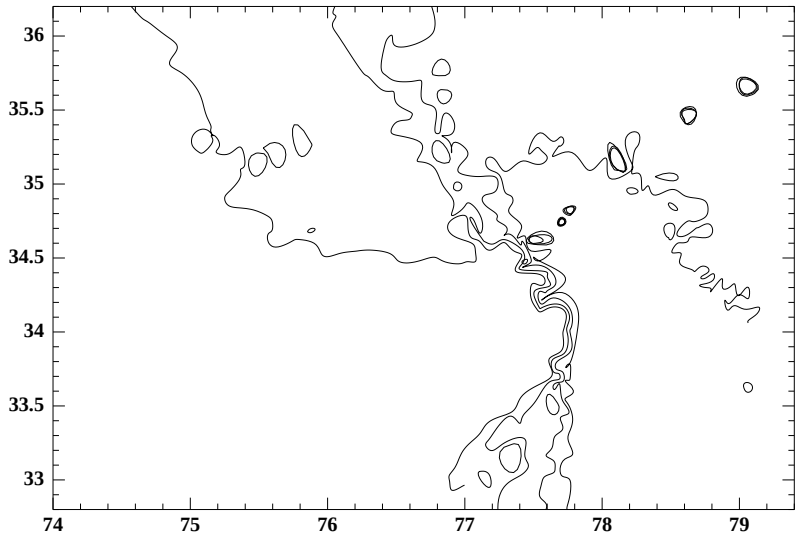

Fig. *4.9* — Traitement par déplacement des isobathes.

du SHOM. Sur la figure 4.11, nous avons représenté les isobathes après traitement des intersections et les isobathes apparaissant sur la carte finale après généralisation manuelle. Comme précédemment, lorsque les isobathes sont déplacées manuellement, le cartographe ne respecte pas toujours la contrainte de sécurité. La méthode de traitement des conflits que nous avons mise en place permet de mieux respecter les contraintes cartographiques. Cependant, toutes les intersections ne peuvent pas être corrigées par déformation et cette méthode doit s'intégrer dans un processus plus complexe. Pour cela, il faut que d'autres opérateurs soient

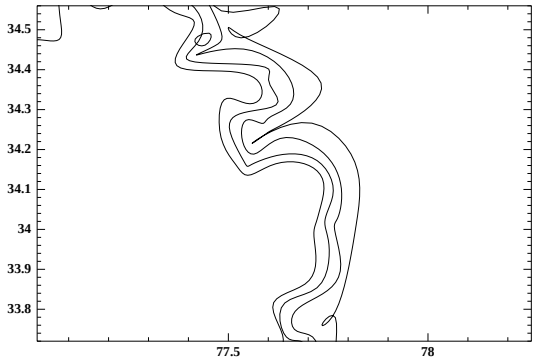

Fig. *4.10* — Correction séquentielle d'un ensemble de conflits.

définis (suppression, agrégation, lissage) et que ces opérateurs soient appliqués en fonction du contexte et des choix du cartographe.

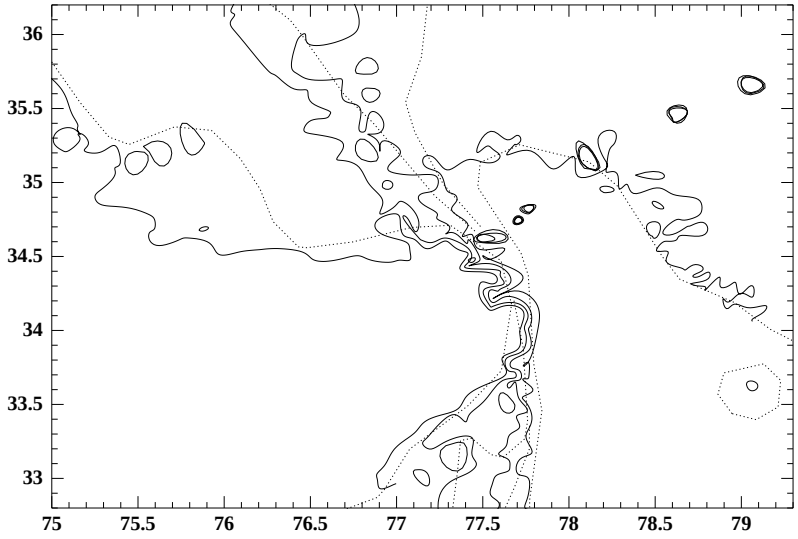

Fig. *4.11* — Isobathes après généralisation manuelle (en pointillés).

4.4 Conclusion

Notre méthode de détection et de correction des conflits a été appliquée à deux morceaux de carte. Même si les conflits sont corrigés localement, en effectuant les corrections en partant des isobathes les moins profondes, nous obtenons de bons résultats et la plupart des conflits sont supprimés. Cependant, d'autres conflits peuvent être créés. Pour éviter cela, il faut qu'une stratégie de généralisation soit mise en place afin de choisir l'opérateur de correction à appliquer. Ceci est d'autant plus important que la forme de la carte finale dépend des

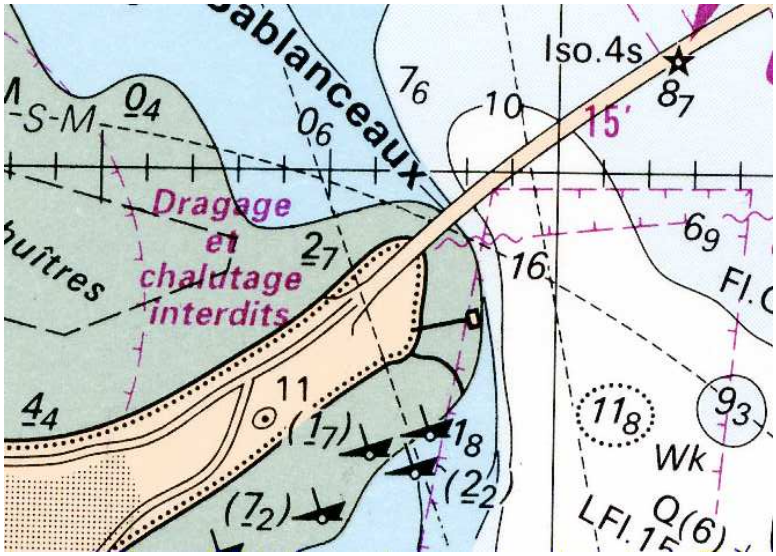

Fig. *4.12* — Extrait de la carte finale 7404 du SHOM.

opérateurs utilisés et de leur ordre d'application. Il faudrait également contrôler si, lors de la déformation, de nouveaux conflits ne sont pas créés.

Enfin, des problèmes apparaissent lors de la construction des courbes. Ils sont liés aux méthodes de construction et aux types de courbe utilisés pour représenter les isobathes. La représentation des courbes sous forme de polylignes ne permet pas d'éviter certains problèmes dus à leur caractère discret (arcs brisés, précision insuffisante à certains endroits). D'autres conflits apparaissent également lors de la construction des courbes B-splines à partir de ces polylignes. Ces problèmes sont inhérents à la méthode de construction utilisée. Une solution serait donc de construire les isobathes initiales directement sous forme de courbes B-splines sans passer par des polylignes. Malheureusement, nous ne disposons pas d'autres données sinon, les courbes pourraient par exemple être construites à partir d'un modèle numérique de terrain issu des relevés. Les contours actifs pourraient être utilisés pour cela, évitant ainsi certains problèmes de paramétrisation : une courbe initiale serait placée sur le modèle de terrain et déformée vers les points de même profondeur. Cela permettrait de réduire le nombre de conflits afin d'avoir moins de corrections à effetuer par la suite. Néanmoins, cette technique demande à mettre en place un mode de fonctionnement différent puisque la généralisation d'une carte ne se ferait plus en partant de courbes déjà construites mais en repartant à chaque fois des points initiaux.

Conclusion et perspectives

Dans ce mémoire, nous nous sommes intéressés à la détection des intersections entre courbes B-splines et à leur correction par des méthodes de déformation. Ces travaux sont appliqués à la généralisation des cartes marines.

Dans un premier chapitre, nous avons étudié différentes méthodes de détection des intersections spécifiques aux courbes paramétriques (algébriques, non linéaires et géométriques). Les méthodes algébriques et non linéaires sont surtout adaptées à la détection des intersections régulières de courbes. Les intersections singulières comme les tangences ou les recouvrements de courbes sont peu traitées. Un autre problème est le traitement des intersections de visualisation. Ces différents types d'intersections nécessitent des méthodes robustes. Pour cela, les méthodes géométriques sont les mieux adaptées.

Les méthodes géométriques se distinguent par le type de volume englobant et la méthode utilisée pour segmenter les courbes. Nous avons testé différentes méthodes sur des courbes B-splines. Il s'agissait de détecter des intersections visuelles sur de grands ensembles de données. Les méthodes les plus rapides et les plus fiables sont les méthodes utilisant des englobants grossiers. Les méthodes plus précises posent des problèmes pour le traitement des singularités. L'intérêt de ces tests est de mettre en évidence les critères à prendre en compte pour la détection des conflits dans un grand ensemble de données comme une carte marine.

Dans le deuxième chapitre, nous avons présenté une méthode de détection des intersections spécifique à la généralisation des cartes marines. Nous avons d'abord défini la généralisation cartographique et précisé les différentes contraintes propres aux cartes marines. Il s'agit principalement de la contrainte de sécurité nous interdisant de déplacer un point à une profondeur inférieure à sa profondeur réelle et de la contrainte de lisibilité permettant de distinguer les isobathes à l'échelle de la carte. Il y a donc conflit entre deux isobathes lorsque la distance de lisibilité n'est pas respectée. Le traitement de ces conflits doit se faire en deux étapes : d'abord, nous devons détecter les conflits (il s'agit, en fait d'intersections de visualisation) puis les corriger. La détection doit se faire avec des méthodes rapides et robustes étant donné que nous devons surtout détecter des intersections singulières et visuelles dans un grand ensemble de courbes.

La méthode que nous avons mise en place procède en deux étapes. Nous effectuons d'abord un partitionnement hiérarchique de la carte à l'aide d'un quadtree. Les courbes sont réparties dans les différentes cellules. Cela permet de limiter les tests d'intersection puisqu'une courbe ne peut être en conflit qu'avec les courbes de la même cellule. Pour assurer la robustesse et la rapidité de la méthode, la segmentation des courbes dans les cellules se fait sans calcul. La répartition se fait en étudiant le polygone de contrôle. Un segment de courbe dans une cellule est délimité par les points de contrôle définissant l'intersection entre la courbe et la cellule. De ce fait, nous nous contentons de comparer les coordonnées des points de contrôle par rapport à la cellule. Afin de tenir compte de la lisibilité, l'appartenance à la cellule est définie à une tolérance égale à la distance de lisibilité. De ce fait, les cellules se recouvrent et certaines intersections sont détectées plusieurs fois. La méthode permet également de limiter la place mémoire puisque les segments sont définis à partir des indices des nœuds délimitant les intervalles paramétriques. La décomposition de la carte est faite jusqu'à pouvoir conclure à une non intersection ou lorsqu'un niveau maximal est atteint.

Ensuite, dans chaque cellule, il faut calculer les intersections entre les segments. Pour cela, les segments sont approchés par des lignes polygonales à l'aide de schémas de subdivision (uniformes ou non suivant le type de courbes B-splines utilisées). Les intersections sont alors définies par les indices des nœuds délimitant les intervalles en conflit. Cela nous permet de connaître les indices des points de contrôle à modifier pour corriger les intersections. Lorsque toutes les intersections ont été détectées dans les cellules, il faut alors regrouper les intervalles entre eux pour éliminer les intersections détectées plusieurs fois et regrouper celles réparties dans plusieurs cellules.

La méthode a été testée sur des jeux de cartes fournis par le SHOM. Nous avons comparé les résultats obtenus en faisant varier la profondeur de l'arbre et la précision des approximations. Les meilleurs résultats sont obtenus pour des arbres de cinq ou six niveaux. Lorsque nous avons moins de niveaux, les cellules sont plus grandes et nous passons plus de temps à approcher les courbes et à calculer les intersections entre des lignes polygonales plus grandes. Au delà de six niveaux, les cellules sont trop petites et les calculs sont redondants. La précision des approximations influe surtout sur le temps de calcul étant donné que la ligne polygonale est définie avec beaucoup plus de points. Le nombre d'intersections varie également puisque nous détectons plus de conflit si l'approximation est mauvaise. Enfin, nous remarquons que dans le cas des courbes B-splines uniformes, l'utilisation de schémas de subdivision uniforme permet de réduire les temps de calcul.

La méthode mise en place a été validée sur des cas réels et donne de bons résultats. Néanmoins, des améliorations sont toujours possibles. Nous n'avons intégré que des critères de distance pour la définition des conflits. Des informations sémantiques sur la profondeur des isobathes, la fermeture ou l'ouverture des courbes ou la nature du fond peuvent être intégrées. Cela fournit de nouveaux critères de segmentation de la carte. Les courbes seraient alors réparties dans plusieurs couches correspondant à différentes classification. La détection des conflits se ferait uniquement entre les courbes d'une même couche. Cela permet également

de réduire la quantité de données avant de détecter les conflits. Par exemple, une isobathe correspondant à un creux peut être directement supprimée si son aire n'est pas suffisamment grande.

Dans le troisième chapitre, nous nous sommes intéressés à la correction des intersections par la déformation des isobathes. La correction doit se faire en tenant compte des contraintes présentées dans le chapitre 2. Nous avons donc expliqué la stratégie de généralisation habituellement utilisée et la prise en compte de ces contraintes. La contrainte de sécurité nous oblige à ne modifier que l'isobathe la plus profonde tandis que la deuxième isobathe reste fixe. La contrainte de lisibilité nous impose un déplacement minimal afin de garantir la suppression du conflit. Ces deux contraintes doivent être impérativement vérifiées pour qu'une correction soit validée. Afin de mesurer la qualité des corrections, nous avons également considéré la contrainte géomorphologique demandant à préserver la forme des courbes. Nous avons défini deux critères pour mesurer la qualité de la solution. En premier lieu, la courbe doit être proche de sa position initiale. Pour cela, nous mesurons la norme du déplacement effectué. Ensuite, les déformations doivent être limitées. Le deuxième critère est donc la norme de la courbure le long de la courbe de déplacement.

Une première classe de méthodes de correction sont les méthodes géométriques où les corrections sont effectuées en calculant un déplacement à chaque point. Deux approches différentes sont possibles. Les méthodes locales corrigent les conflits séparément. Le résultat final dépend de l'ordre dans lequel sont effectuées les corrections. Les méthodes globales résolvent l'ensemble des conflits en même temps en exprimant les contraintes sous forme d'équations et en les assemblant dans un système. En général, il y a beaucoup plus d'équations que d'inconnues. Le problème de ces méthodes est donc qu'elles ne permettent pas de garantir l'ensemble des contraintes. Nous présentons donc une méthode garantissant les contraintes fortes de sécurité et de lisibilité.

La méthode est fondée sur une approche géométrique. Nous calculons un déplacement à effectuer pour chaque point de contrôle en fonction du déplacement minimal. La solution est valide mais les déplacements peuvent être importants. Nous appliquons donc, dans un deuxième temps, un second algorithme de déformation considérant le polygone de contrôle comme un réseau de barres. Des forces ramenant les points de contrôle vers les points initiaux sont appliquées. Au fur et à mesure du déplacement, les points de contrôle sont fixés pour que la courbe reste toujours à une distance suffisante. Le processus est arrêté lorsque tous les points sont fixés. Cette méthode corrige bien les conflits mais la contrainte de forme n'est pas prise en compte. Nous présentons donc un autre modèle de correction.

Une deuxième classe de méthodes sont les modèles déformables. Il s'agit de modèles physiques où des énergies sont définies en fonction des contraintes. Le modèle se déforme alors automatiquement pour minimiser les énergies et atteindre une position stable. Nous avons mis en place une méthode de correction fondée sur les contours actifs. Le contour actif représente le déplacement appliqué à la courbe. L'énergie externe est fonction de la distance de déplacement minimal. Lorsque le déplacement est supérieur au déplacement minimal,

l'énergie est nulle et la contrainte de lisibilité est satisfaite. L'énergie interne est définie par la courbure du déplacement et correspond à la contrainte géomorphologique. Le contour actif est alors déformé pour minimiser l'énergie totale. Pour valider les contraintes fortes, nous définissons une énergie externe supérieure à l'énergie interne. Afin d'appliquer la méthode à tous les types d'intersections, nous proposons un réglage automatique du paramètre de forme fixant le rapport entre les énergies. Cela permet d'assurer la convergence du snake vers une solution valide tout en tenant compte de la forme de la courbe.

Nous avons appliqué ces méthodes sur des conflits détectés précédemment. Il apparaît que la méthode de réseau de barres est plus rapide mais donne de moins bons résultats suivant les critères que nous avons définis. Cela vient du fait que pour les réseaux de barres, les directions de déplacement des points de contrôle sont imposées et que certains points sont fixés alors que pour les contours actifs, les déplacements peuvent se faire dans toutes les directions. La méthode des contours actifs est ensuite étendue au traitement des auto-intersections visuelles et singulières.

Dans ce chapitre, nous traitons de l'intersection et de l'auto-intersection d'isobathes sans tenir compte de l'environnement. Les courbes situées dans le voisinage doivent être prises en compte afin de ne pas créer de nouveaux conflits. Le voisinage du conflit est facilement connu à partir du quadtree défini dans le chapitre 2. A partir des courbes voisines, nous pouvons définir un déplacement maximal de la même façon que nous avons défini le déplacement minimal. Les forces ou les énergies externes seraient alors définies de sorte à pénaliser le système lorsque le déplacement est trop important. Nous n'avons pas traité la correction des auto-intersections franches. Dans ce cas, les méthodes de déplacement précédentes ne peuvent pas être appliquées. Par contre, l'auto-intersection peut être corrigée en perturbant la courbe [Andersson et al., 1998] ou en étudiant la géométrie du polygone de contrôle [Kantorowitz et Schechner, 1993, Neagu et al., 2000].

La méthode de correction peut aussi être étendue au traitement des intersections de plusieurs courbes. La correction peut alors se faire en déformant toutes les courbes simultanément ou deux à deux jusqu'à avoir corriger tous les conflits. D'autres opérateurs de généralisation peuvent être définis à partir des méthodes de déformation présentées comme par exemple, un opérateur d'agrégation : le snake est placé sur la carte et est déformé pour adhérer au contour des segments à agréger. La position initiale du snake serait définie à partir d'un volume englobant les segments.

Concernant le réglage automatique des paramètres de forme, nous ne traitons que le paramètre de courbure β. Si les contours actifs sont utilisés pour définir d'autres opérateurs, il peut être nécessaire de régler le paramètre de tension α. Un problème intéressant serait donc d'intégrer α dans le calcul automatique des paramètres sachant que la définition des paramètres est propre à chaque opérateur puisque les contraintes sont différentes.

Enfin, un dernier point est que les méthodes présentées dans ce mémoire sont définies pour la construction de cartes mais peuvent être utilisées également pour la mise à jour de

ces cartes, par exemple, si des données source ont été modifiées ou ajoutées. La structure hiérarchique permet d'insérer de nouvelles courbes à la racine puis de les répartir dans les différentes cellules. Les intersections ne peuvent apparaître que dans les cellules feuilles où de nouveaux segments ont été ajoutés. En fonction des modifications effectuées, il est aussi possible de modifier la structure en ajoutant ou en supprimant des cellules.

Ces méthodes peuvent également être appliquées pour définir la trajectoire d'un objet afin d'éviter les collisions avec d'autres objets fixes ou mobiles dans des domaines comme les SIG ou la CAO. Ainsi, dans le cadre du développement de systèmes d'aide à la navigation [Fournier et al., 2003], elles peuvent être utilisées pour définir la route d'un bâtiment. Nous pourrions alors détecter les collisions éventuelles et modifier la route du navire en conséquence. Afin d'assurer la sécurité de la navigation, il faudrait éviter les zones peu profondes et passer à distance des autres navires. Les conflits seraient donc définis avec une précision ϵ_{vis} suffisamment large compte tenu de la taille du navire et de sa manœuvrabilité. La mise à jour de la trajectoire pourrait être faite régulièrement en fonction des déplacement des autres navires.

Nos méthodes peuvent aussi être appliquées en CFAO à la définition d'une trajectoire d'usinage, l'objectif étant de définir les différentes passes de l'outil en s'assurant que la distance entre la forme à usiner et la trajectoire est toujours supérieure à la largeur de l'outil. Cette largeur définirait la précision ϵ_{vis}. Pour ce type d'application, il serait aussi intéressant d'étendre les modèles présentés pour des courbes planes aux courbes de l'espace.

ANNEXE

A Les réseaux de barres

Les réseaux de barres ont été introduits pour la modélisation de courbes et surfaces par Léon dans [Léon et Trompette, 1995]. Les auteurs présentent une méthode de déformation des courbes et des surfaces B-splines où le polygone ou le polyèdre de contrôle est assimilé à un réseau de barres. Les déformations sont effectuées en modifiant les forces internes donnant la tension dans les barres ou les forces externes appliquées aux points de contrôle. L'intérêt de la méthode est que la condition d'équilibre s'exprime sous la forme d'un système linéaire.

Un réseau est constitué de $m+1$ points $(Q_i)_{i=0}^m$ et de $p+1$ segments $(b_j)_{j=0}^p$ reliant les points. Il est associé à un champ de $m+1$ vecteurs $(F_i^{ext})_{i=0}^m$ représentant les forces extérieures appliquées aux points du système. L'équilibre du réseau de barres est assujetti :

- aux forces extérieures appliquées aux points,
- aux liaisons entre les points et l'environnement,
- à la détermination d'une position d'équilibre entre les barres du réseau.

Le réseau est en équilibre lorsque la somme des forces appliquées en chaque point est nulle.

$$F_i^{ext} + \sum F_j^{int} = 0, \quad i = 0, \ldots, m \tag{A.1}$$

où F_j^{int} sont les forces internes appliquées au point Q_i.

Chaque force interne peut être remplacée par une densité de force définie par :

$$q = \frac{F^{int}}{l} \tag{A.2}$$

où l représente la longueur de la barre. Soit $\tilde{Q}$ la matrice diagonale $((p+1) \times (p+1))$ issue des densités de force suivant le principe

$$\tilde{Q}_{ij} = \begin{cases} q_j = \frac{F_j^{int}}{l_j} & \text{si } i = j \\ 0 & \text{sinon} \end{cases} \tag{A.3}$$

Les barres du réseau sont définies à l'aide d'une matrice de connectivité C de taille $((p+1) \times (m+1))$ détaillée par :

$$C_{kl} = \begin{cases} 1 & \text{si la barre } k \text{ prend son origine ou se termine au point } l \\ -1 & \text{si la barre } k \text{ prend son origine ou se termine au point } l \\ & \text{et que son autre extrémité est marquée par un 1} \\ 0 & \text{sinon} \end{cases} \tag{A.4}$$

Cette matrice est divisée en deux matrices C_f et C_l regroupant les colonnes des mf points fixes et des ml points libres. En CFAO, les points libres et fixes sont déterminés par l'utilisateur.

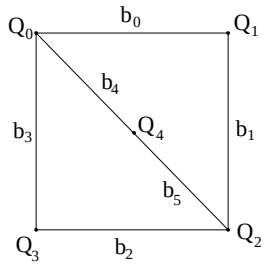

Fig. *A.1* — Exemple de réseau de barres.

Sur l'exemple de la figure A.1, les points au bord sont fixés et le point au centre est libre. Les matrices C_f et C_l correspondantes sont :

$$C_f = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad C_l = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ -1 \end{pmatrix} \tag{A.5}$$

L'équation (A.1) peut être écrite pour tous les points du système sous la forme matricielle :

$$\begin{aligned} C_l^t \tilde{Q} C_l x_l + C_l^t \tilde{Q} C_f x_f &= F_x^{ext} \\ C_l^t \tilde{Q} C_l y_l + C_l^t \tilde{Q} C_f y_f &= F_y^{ext} \\ C_l^t \tilde{Q} C_l z_l + C_l^t \tilde{Q} C_f z_f &= F_z^{ext} \end{aligned} \tag{A.6}$$

Les inconnues du système (A.6) sont les vecteurs x_l, y_l, z_l contenant les coordonnées des points libres. Les éléments q_i sont tous strictement positifs. Ce système possède une solution unique quelles que soient les forces externes ou la topologie du réseau [Léon et Trompette, 1995].

Dans le cadre de la modification de surfaces splines, nous partons d'une surface initiale où tous les points sont fixes et les inconnues sont les forces externes. L'équation (A.6) est résolue pour définir ces forces. Ensuite, une déformation peut être appliquée de deux façons : l'utilisateur peut modifier les forces externes ou la densité de forces internes.

ANNEXE B

Propriétés mécaniques des poutres

Ce chapitre fournit les résultats de base concernant le calcul des énergies de déformation d'une poutre soumise à des contraintes de traction ou de flexion. Des détails supplémentaires peuvent être obtenus dans [Imbert, 1995].

La généralisation cartographique déforme et déplace des objets en fonction de la place disponible. Le principe des méthodes mécaniques est de traiter les éléments comme des structures mécaniques soumises à des contraintes de traction et de fléchissement. Les éléments doivent alors être déformés afin d'équilibrer les contraintes internes et externes. La méthode présentée dans [Bader, 2001] est fondée sur la propriété suivante :

Propriété 6 *Un objet déformable est en équilibre si la somme de toutes les forces qui lui sont appliquées est nulle en tout point et si la somme des moments est nulle en tout point afin d'empêcher les mouvements par translation et rotation.*

Lorsqu'un objet est soumis à des forces ou des moments externes, des forces et des moments internes résultants apparaissent. Ces forces sont mesurées en fonction de leur distribution à l'intérieur de l'objet. Si l'objet est sectionné suivant une section S normale à l'axe z et que l'on prend un élément de surface ΔA (figure B.1), la quantité t définie par

$$t = \lim_{\Delta A \to 0} \frac{\Delta F}{\Delta A} \tag{B.1}$$

avec ΔF le résultant de toutes les forces agissant sur ΔA existe et est la contrainte interne. Elle se décompose en contrainte normale σ_z et en contraintes tangentielles τ_{zy} et τ_{zx}. σ_z représente l'intensité de traction ou de compression sur un élément de surface. τ_{zy} et τ_{zx} représentent l'intensité des forces tangentes à un élément de surface. Des contraintes normales et tangentielles peuvent également être calculées suivant les plans orthogonaux aux axes x et y.

L'application de forces provoque la déformation non uniforme de l'objet. Les déformations sont mesurées à l'aide des contraintes de déformation. On observe des déformations normales (dans ce cas, la contrainte de déformation ϵ représente l'élongation (ou la contraction) par

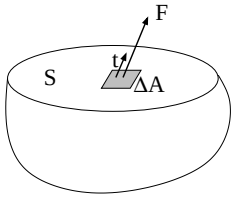

Fig. *B.1* — Définition des contraintes internes.

unité de longueur) et des changements d'angle originellement perpendiculaires (il s'agit de la contrainte tangentielle γ). Dans le cas où les déformations u et v se font suivant les axes x et y, les contraintes sont définies comme suit :

$$\begin{aligned} \epsilon_x &= \frac{\partial u}{\partial x} \\ \epsilon_y &= \frac{\partial v}{\partial y} \\ \gamma_{xy} &= \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \end{aligned} \tag{B.2}$$

Le rapport entre les contraintes internes et externes est donné par la loi de Hooke pour un matériau élastique :

$$\begin{pmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{pmatrix} = \frac{E}{1-\nu^2} \begin{pmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{pmatrix} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{pmatrix} \tag{B.3}$$

où E est le module d'Young définissant la rigidité du matériau et ν décrit la capacité du matériau à changer de volume.

Dans le cadre de la généralisation de lignes, chaque segment de polyligne est considérée comme une poutre soumise à des efforts de traction et de flexion. Il y a traction lorsque la poutre est soumise à une force alignée avec son axe. Dans le cas de la poutre, seules les contraintes normales sont prises en compte. La loi de Hooke se résume donc à

$$\sigma = E\epsilon \tag{B.4}$$

Par la suite, nous nous plaçons dans un plan xOy avec O l'origine située à l'extrémité de la poutre, x l'axe dirigé suivant l'axe de la poutre et y l'axe orthogonal à x. Les déformations sont notées u dans la direction x et v dans la direction y.

Lorsqu'une force F alignée suivant une direction ξ provoquant une déformation dans la même direction est appliquée, le travail U de F est défini par

$$U = \int_0^D F d\xi \tag{B.5}$$

où D est le déplacement final conséquent à l'application de la force. Le matériau étant élastique, la force est linéaire, comprise entre 0 et $\frac{P}{D}$ où P est la force maximale. De ce fait, le travail vaut

$$U = \int_0^D \frac{P}{D} \xi d\xi = \frac{1}{2} PD \tag{B.6}$$

La contrainte de traction normale induite sur un élément de surface est définie par $\sigma = \frac{dF}{dA}$ avec dA un élément de surface orthogonal à F. Nous avons donc

$$dF = \sigma dA \tag{B.7}$$

D'après (B.2), le déplacement élémentaire est donné par $dD = \epsilon d\xi$. Par conséquent, $dF dD = \sigma\epsilon\, d\xi\, dA$ soit

$$dF dD = \sigma\epsilon dV \tag{B.8}$$

où dV est un élément de volume. L'énergie de déformation dU créé par l'application d'une force dF provoquant un déplacement dD vaut :

$$dU = \frac{1}{2}\sigma\epsilon dV \tag{B.9}$$

L'énergie d'un corps soumis à une contrainte normale est donc donnée par

$$U = \int_V \frac{\sigma\epsilon}{2} dV \tag{B.10}$$

Dans le cas où la force F est appliquée suivant l'axe x de la poutre, les contraintes varient uniquement suivant x. L'énergie de déformation de F peut s'écrire

$$U = \int_A \int_L \frac{\sigma\epsilon}{2} dx dA = \int_L \frac{A}{2}\sigma\epsilon dx \tag{B.11}$$

En appliquant la loi de Hooke, le travail devient

$$U = \int_L \frac{AE}{2}\epsilon^2 dx = \int_L \frac{AE}{2}(\frac{du}{dx})^2 dx \tag{B.12}$$

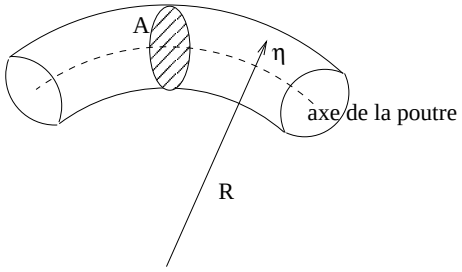

Fig. *B.2* — Poutre en flexion.

Considérons maintenant une poutre pouvant se déformer en flexion (figure B.2). La déformation en un point de la poutre situé à une distance η de l'axe de la poutre s'exprime par la relation

$$\epsilon = \frac{\eta}{R} \tag{B.13}$$

où R est le rayon de courbure de la poutre. Soit, en prenant la courbure égale à la dérivée seconde du déplacement v par rapport à x,

$$\epsilon = \eta\frac{d^2v}{dx^2} \tag{B.14}$$

D'où la contrainte :

$$\sigma = E\eta\frac{d^2v}{dx^2} \tag{B.15}$$

Le travail vaut donc

$$U = \frac{1}{2}\int_V E\eta^2(\frac{d^2v}{dx^2})^2dV = \frac{1}{2}\int_L E\frac{d^2v}{dx^2}dx\int_A \eta^2 dA \tag{B.16}$$

En notant

$$I = \int_A \eta^2 dA \tag{B.17}$$

le moment d'inertie, l'énergie de déformation d'une poutre soumise à une force de flexion est donc

$$U = \int_L \frac{EI}{2}(\frac{d^2v}{dx^2})^2dx \tag{B.18}$$

Bibliographie

[Andersson et al., 1998] Andersson, L.-E., Peters, T., et Stewart, N. (1998). Self-intersection of composite curves and surfaces. *Computer Aided Geometric Design*, 15(5) :507–527.

[Aziz et al., 1990] Aziz, N., Bata, R., et Bhat, S. (1990). Bézier surface/surface intersection. *IEEE Computer Graphics & Applications*, pages 50–58.

[Bader, 2001] Bader, M. (2001). *Energy minimization methods for feature displacement in map generalization.* PhD thesis, Universität Zürich.

[Barrault et Bader, 2002] Barrault, M. et Bader, M. (2002). Les algorithmes de déplacement. In Ruas, A., editor, *Généralisation et représentations multiples*, pages 287–302. Hermes.

[Beard, 1991] Beard, K. (1991). Constraints on rule formation. In Buttenfield, B. et McMaster, R. B., editors, *Map Generalization*, pages 121–135.

[Berglund et al., 2001] Berglund, T., Erikson, U., Jonsson, H., Mrozek, K., et Söderkvist, I. (2001). Automatic generation of smooth paths bounded by polygonal chains. In *International Conference on Computational Intelligence for Modelling Control and Automation.*

[Bobrich, 1996] Bobrich, J. (1996). *Ein neuer Ansatz zur kartographischen Verdrängung auf der Grundlage eines mechanischen Federmodells.* PhD thesis, Deutsche Geodätische Kommission, München.

[Bobrich, 2001] Bobrich, J. (2001). Cartographic displacement by minimization of spatial and geometric conflicts. In *20th International Cartographic Conference, Bejing*, volume 3, pages 2032–2042.

[Boehm, 1980] Boehm, W. (1980). Inserting new knots into B-spline curves. *Computer Aided Design*, 12(4) :199–201.

[Brigger et al., 2000] Brigger, P., Hoeg, J., et Unser, M. (2000). B-spline snakes : A flexible tool for parametric contour detection. *IEEE Transactions on Image Processing*, 9(9) :1484–1496.

[Brunet et al., 1993] Brunet, P., Navazo, I., et Vinacua, A. (1993). A modelling scheme for the approximate representation of closed surfaces. *Computing Suppl.*, 8 :75–90.

[Burghardt et Meier, 1997] Burghardt, D. et Meier, S. (1997). Cartographic displacement using the snakes concept. In Förstner, W. et Plümer, L., editors, *Semantic modeling for the acquisition of topographic information from images and maps*, pages 59–71.

[Cohen et al., 1980] Cohen, E., Lyche, T., et Riesenfeld, R. (1980). Discrete B-spline subdivision techniques in computer-aided geometric design and computer graphics. *Computer Graphics and Image Processing*, 14 :87–111.

[Cohen et Cohen, 1993] Cohen, L. et Cohen, I. (1993). Finite element methods for active contours models and ballons for 2D and 3D images. *IEEE Pattern Analysis and Machine Intelligence*, pages 1131–1147.

[Creac'h et al., 2000] Creac'h, D., Devogele, T., Quimerc'h, C., et Saux, E. (2000). Aide à la préparation des cartes marines, cahier des charges fonctionnel. Technical Report 00-0002, Institut de Recherche de l'Ecole Navale, Brest.

[Creach et Le Gac, 1995] Creach, D. et Le Gac, J.-C. (1995). Etude d'aide à la préparation des cartes marines. Projet de Fin d'Etude, ENSIETA - EPSHOM.

[Daniel, 1989] Daniel, M. (1989). Modélisation de courbes et surfaces par des B-splines. application à la conception et à la visualisation de formes. Thèse de Doctorat, Université de Nantes.

[Daniel, 1992] Daniel, M. (1992). A curve intersection algorithm with processing of singular cases : introduction of a clipping technique. In Lyche, T. et Schumaker, L., editors, *Mathematical Methods in Computer Aided Geometric Design II*, pages 161–170.

[Delingette, 1994] Delingette, H. (1994). Modélisation, déformation et reconnaissance d'objets tridimensionnels à l'aide de maillages complexes. Thèse de Doctorat, Ecole Centrale de Paris.

[Dietz, 1996] Dietz, U. (1996). B-spline approximation with energy constraints. In Hoschek, J. et Kaklis, P., editors, *Reverse engineering*, pages 229–239.

[Dubois, 2000] Dubois, V. (2000). Auto-intersection de courbes planes. Rapport de stage, D.E.A. Informatique, Université de Nantes.

[Duchêne et Regnauld, 2002] Duchêne, C. et Regnauld, N. (2002). Le modèle AGENT. In Ruas, A., editor, *Généralisation et représentations multiples*, pages 369–385. Hermes.

[Farin, 1992] Farin, G. (1992). *Courbes et surfaces pour la CGAO*. Masson, édition française.

[Fei, 2002] Fei, L. (2002). *A Method of Automated Cartographic Displacement - On the Relationship between Streets and Buildings*. PhD thesis, Universität Hannover.

[Figueiredo, 1996] Figueiredo, L.-H. (1996). Surface intersection using affine arithmetic. In Davis, W. A. et Bartels, R., editors, *Graphics Interface '96*, pages 168–175.

[Foley et al., 1995] Foley, J., van Dam, A., Feiner, S., Hugues, J., et Phillips, R. (1995). *Introduction à l'infographie*. Addison-Wesley, édition française.

[Fortune et Van Wyk, 1993] Fortune, S. et Van Wyk, C. (1993). Efficient exact arithmetic for computational geometry. In *Proceedings of the 9th ACM Symposium on Computational Geometry*, pages 163–172.

[Foufou, 1997] Foufou, S. (1997). Contribution à l'algorithmique des intersections de surface en algèbre des volumes. Thèse de Doctorat, Université Claude Bernard, Lyon I.

[Fournier et al., 2003] Fournier, S., Devogele, T., et Claramunt, C. (2003). A role-based multi-agent model for concurrent navigation systems. In *Proceedings of the* 6^{th} *AGILE conference*, pages 623–632.

[Goldenthal et Bercovier, 2003] Goldenthal, R. et Bercovier, M. (2003). Spline curve approximation and design by optimal control over the knots. Technical Report 2003-7, Leibniz Center.

[Gottschalk et al., 1996] Gottschalk, S., Lin, M., et Manocha, D. (1996). OBBTree : A hierarchical structure for rapid interference detection. In *Proceedings of the ACM Conference on Computer Graphics*, pages 171–180.

[Guilbert, 2002] Guilbert, E. (2002). Détection des conflits entre isobathes en généralisation cartographique. *Revue internationale de géomatique*, 12(4) :421–438.

[Guilbert et al., 2003] Guilbert, E., Saux, E., et Daniel, M. (2003). A hierarchical structure for locating intersections in large sets of B-spline curves. In Lyche, T., Mazure, M.-L., et Schumaker, L. L., editors, *Curves and Surfaces Design*, pages 205–214. Nashboro Press.

[Guilbert et al., 2004] Guilbert, E., Saux, E., et Daniel, M. (2004). Combining geometrical and mechanical displacements for suppressing intersections in cartographic generalization. In *WSCG Short Communications*, pages 87–94. Union Agency - Science Press.

[Harrie, 1999] Harrie, L. (1999). The constraint method for solving spatial conflicts in cartographic generalization. *Cartography and geographic information science*, 26(1) :55–69.

[Harrie et Sarjakoski, 2002] Harrie, L. et Sarjakoski, T. (2002). Simultaneous graphic generalization of vector data sets. *GeoInformatica*, 6(3) :233–261.

[Hlúšek, 2000] Hlúšek, R. (2000). Bézier curves intersection using relief perspective. In *WSCG Short Communications*.

[Ho et Cohen, 2000] Ho, C.-C. et Cohen, E. (2000). Surface self-intersection. In Lyche, T. et Schumaker, L. L., editors, *Mathematical methods for curves and surfaces : Oslo 2000*, pages 183–194.

[Hoffmann et Juhász, 2001] Hoffmann, M. et Juhász, I. (2001). Shape control of cubic B-spline and NURBS curves by knot modifications. In *Fifth International Conference on Information Visualisation*, pages 63–70.

[Højholt, 2000] Højholt, P. (2000). Solving space conflicts in map generalization : using a finite element method. *Cartography and geographic information science*, 27(1) :65–73.

[Hu et al., 1996] Hu, C.-Y., Maekawa, T., Sherbrooke, E., et Patrikalakis, N. (1996). Robust interval algorithm for curve intersections. *Computer-aided Design*, 28(6-7) :495–506.

[Huet, 1996] Huet, T. (1996). Généralisation de cartes vectorielles d'occupation des sols : une approche par triangulation coopérative pour la fusion des polygones. Thèse de Doctorat, Université de Toulouse 1.

[Imbert, 1995] Imbert, J. (1995). *Analyse des structures par éléments finis*. Cépaduès-éditions, troisième édition.

[Jacob et al., 2001] Jacob, M., Blu, T., et Unser, M. (2001). A unifying approach and interface for spline-based snakes. In *Proceedings of the SPIE International Symposium on Medical Imaging : Image Processing*, pages 340–347.

[Jiang et Nakos, 2004] Jiang, B. et Nakos, B. (2004). Line simplification using self-organizing maps. Technical report, University of Gävle.

[Kantorowitz et Schechner, 1993] Kantorowitz, E. et Schechner, Y. (1993). Managing the shape of planar splines by their control polygons. *Computer-Aided Design*, 25 :355–364.

[Kass et al., 1987] Kass, M., Witkin, A., et Terzopoulos, D. (1987). Snakes : active contour models. In *Proceedings of International Conference on Computer Vision*, pages 259–268.

[Kim et al., 1998] Kim, D.-S., Lee, S.-W., et Shin, H. (1998). A cocktail algorithm for planar Bézier curve intersections. *Computer-Aided Design*, 30(13) :1047–1051.

[Koparkar et Mudur, 1983] Koparkar, P.-A. et Mudur, S. (1983). A new class of algorithms for processing of parametric curves. *Computer-Aided Design*, 15(1) :407–429.

[Krishnan et al., 1998] Krishnan, S., Gopi, M., Lin, M., Manocha, D., et Pattekar, A. (1998). Rapid and accurate contact determination between spline models using shelltrees. *Computer Graphics Forum*, 17(3) :315–326.

[Lane et Riesenfield, 1980] Lane, J. et Riesenfield, R. (1980). A theoritical developpement for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2 :35–46.

[Larsen et al., 1995] Larsen, O., Radeva, P., et Martí, E. (1995). Guidelines for choosing optimal parameters of elasticity for snakes. In *Proceedings of International Conference on Computer Analysis and Image Processing*.

[Lasser, 1989] Lasser, D. (1989). Calculating the self-intersections of Bézier curves. *Computer in industry*, 10 :259–268.

[Lee, 1989] Lee, E. (1989). Choosing nodes in parametric curve interpolation. *Computer Aided Design*, 21(6) :363–370.

[Léon, 1991] Léon, J.-C. (1991). *Modélisation et construction de surfaces pour la CFAO*. Hermès.

[Léon et Trompette, 1995] Léon, J.-C. et Trompette, P. (1995). A new approach towards free-form surfaces control. *Computer Aided Geometric Design*, 12 :395–416.

[Lonergan et Jones, 2001] Lonergan, M. et Jones, C. (2001). An iterative displacement method for conflict resolution in map generalization. *Algorithmica*, 30 :287–301.

[Lutterkort et Peters, 1999] Lutterkort, D. et Peters, J. (1999). Smooth paths in a polygonal channel. In *Proceedings of the* 15^{th} *annual ACM Symposium on Computer Geometry*, pages 316–321.

[Lutterkort et Peters, 2000] Lutterkort, D. et Peters, J. (2000). Linear envelopes for uniform B–spline curves. In *Curves and Surfaces, St Malo*, pages 239–246.

[Maekawa et Patrikalakis, 1993] Maekawa, T. et Patrikalakis, N. (1993). Computation of singularities and intersections of offsets of planar curves. *Computer-Aided Geometric Design*, 10(5) :407–429.

[Manocha et Demmel, 1994] Manocha, D. et Demmel, J. (1994). Algorithms for intersecting parametric and algebraic curves I : Simple intersections. *ACM Transactions on Graphics*, 13(1) :73–100.

[Manocha et Demmel, 1995] Manocha, D. et Demmel, J. (1995). Algorithms for intersecting parametric and algebraic curves II : multiple intersections. *Graphical models and image processing : GMIP*, 57(2) :81–100.

[Meegama et Rajapakse, 2003] Meegama, R. et Rajapakse, J. (2003). NURBS snakes. *Image Vision Comput.*, 21 :551–562.

[Meek et al., 2003] Meek, D., Ong, B., et Walton, D. (2003). Constrained interpolation with rational cubics. *Computer Aided Geometric Design*, 20 :253–275.

[Michelucci et Moreau, 1995] Michelucci, D. et Moreau, J.-M. (1995). Lazy arithmetic. Technical report, E.M.S.E., Saint-Étienne.

[Müller et al., 1995] Müller, J. C., Weibel, R., Lagrange, J. P., et Salgé, F. (1995). Generalization : State of the art and issues. In *GIS and Generalization - Methodology and Practice*, pages 3–17. Taylor & Francis.

[Neagu et al., 2000] Neagu, M., Calcoen, E., et Lacolle, B. (2000). Bézier curves : Topological convergence of the control polygon. In *Mathematical Methods in CAGD : Oslo 2000*, pages 347–354.

[Neagu et Lacolle, 1999] Neagu, M. et Lacolle, B. (1999). Computing the combinatorial structure of arrangements of curves using polygonal approximations. Technical Report RR 1015-M, IMAG, Grenoble.

[Nickerson, 1988] Nickerson, B. (1988). Automated cartographic generalization for linear features. *Cartographica*, 25(3) :15–66.

[Nicolas, 1995] Nicolas, A. (1995). Contribution à l'étude de l'intersection de surfaces gauches. Thèse de Doctorat, Université de Nantes.

[OHI, 1988] OHI (1988). Spécifications de l'OHI pour les cartes marines et règlement de l'OHI pour les cartes internationales. n° 250-VI-1988 OHI.

[Pernot et al., 2003] Pernot, J.-P., Guillet, S., Léon, J.-C., Falcidieno, B., et Giannini, F. (2003). A new approach to minimisation for shape control during free-form surface deformation. In *Proceedings of the 29th Design Automation Conference*.

[Peters et Wu, 2001] Peters, J. et Wu, X. (2001). Optimized refinable surface enclosures. Technical report, University of Florida.

[Pottmann et al., 2002] Pottmann, H., Leopoldseder, S., et Hofer, M. (2002). Approximation with active B-spline curves and surfaces. In *Proceedings of Pacific Graphics 02*, pages 8–25. IEEE Computer Society.

[Preparata et Shamos, 1985] Preparata, F. et Shamos, M. (1985). *Computational Geometry : an Introduction*. Springer-Verlag.

[Radeva et al., 1995] Radeva, P., Serrat, J., et Martí, E. (1995). A snake for model-based segmentation. In *International Conference on Computer Vision (ICCV'95)*, pages 816–821.

[Ruas et Bianchin, 2002] Ruas, A. et Bianchin, A. (2002). Echelle et niveau de détail. In Ruas, A., editor, *Généralisation et représentation multiple*, pages 25–44. Hermes.

[Ruas et Libourel, 2002] Ruas, A. et Libourel, T. (2002). Introduction. In Ruas, A., editor, *Généralisation et représentation multiple*, pages 17–22. Hermes.

[Sabin, 2000] Sabin, M. (2000). Computer aided geometric design. an elementary and superficial course. http ://www.damtp.cam.ac.uk/user/na/people/Malcolm/mas.html.

[Samet, 1990] Samet, H. (1990). *Applications of spatial data structures*. Addison Wesley.

[Saux, 1999] Saux, E. (1999). Lissage de courbes par des B-splines. Application à la compression et à la généralisation cartographique, Thèse de Doctorat, Université de Nantes.

[Saux et Daniel, 1999] Saux, E. et Daniel, M. (1999). Data reduction of polygonal curves using B-splines. *Computer Aided Design*, 31 :507–515.

[Saux et al., 2002] Saux, E., Thibaud, R., Devogele, T., Béra, R., et Guilbert, E. (2002). Généralisation des cartes marines. In Ruas, A., editor, *Généralisation et représentations multiples*, pages 303–318. Hermes.

[Sederberg et Nishita, 1990] Sederberg, T. et Nishita, T. (1990). Curve intersection using Bézier clipping. *Computer Aided Design*, 22(9) :538–549.

[Sederberg et Parry, 1986] Sederberg, T. et Parry, S. (1986). Comparison of three curve intersection algorithms. *Computer-Aided Design*, 18(1) :58–64.

[Sederberg et al., 1989] Sederberg, T., White, S., et Zundel, A. (1989). Fat arcs : a bounding region with cubic convergence. *Computer Aided Geometric Design*, 6 :205–218.

[Sester, 2000] Sester, M. (2000). Generalization based on least squares adjustment. In *International Archives of Photogrammetry and Remote Sensing*, volume 33.

[SHOM, 1984] SHOM (1984). Conception, confection, présentation des cartes marines. Document interne.

[Ware et Jones, 1998] Ware, J. et Jones, C. (1998). Conflict reduction in map generalization using iterative improvement. *Geoinformatica*, 2(4) :383–407.

[Weibel, 1991] Weibel, R. (1991). Amplified intelligence and rule-based systems. In Buttenfield, B. et McMaster, R., editors, *Map Generalization : Making Rules for Knowledge Representation*, pages 172–186.

[Weibel et Dutton, 1999] Weibel, R. et Dutton, G. (1999). Generalising spatial data and dealing with multiple representations. In *Geographical Information Systems*. Wiley.

[Williams et Shah, 1992] Williams, D. et Shah, M. (1992). A fast algorithm for active contours and curvature estimation. *CVGIP : Image Understanding*, 55(1) :14–26.

[Wilson et al., 2003] Wilson, I., Ware, J., et Ware, J. (2003). Reducing graphic conflict in scale reduced maps using a genetic algorithm. In *Fifth Workshop on Progress in Automated Map Generalization*.

[Yang et al., 2004] Yang, H., Wang, W., et Sun, J. (2004). Control point adjustment for B-spline curve approximation. *Computer-Aided Design*, 36(7) :639–652.

[Zhang et al., 2001] Zhang, C., Zhang, P., et Cheng, F. (2001). Fairing spline curves and surfaces by minimizing energy. *Computer-Aided Design*, 33 :913–923.

Détection et correction des intersections entre courbes B-splines. Application à la généralisation cartographique

Cette thèse présente une méthode de détection et de correction des intersections visuelles et singulières entre courbes B-splines adaptée à la généralisation des cartes marines. Dans une première partie, nous nous intéressons à la détection des intersections. La méthode proposée effectue d'abord un partitionnement du plan. Les courbes sont réparties dans les cellules sans calcul numérique. Le partitionnement est donc rapide et robuste. Ensuite, les intersections sont calculées à l'aide de schémas de subdivision. La deuxième partie concerne la correction des conflits par déformation respectant les contraintes cartographiques. Nous présentons une première méthode où le polygone de contrôle est assimilé à un réseau de barres déformé par l'application de forces externes. Une deuxième méthode est ensuite présentée où le déplacement est représenté par un snake soumis à des énergies définies en fonction des conflits. Les paramètres de forme sont réglés automatiquement.

Mots clés : *courbes B-splines, intersection, quadtree, réseau de barres, contours actifs, snake, généralisation cartographique.*

Detection and correction of intersections between B-spline curves applied to cartographic generalisation

In this thesis, a method for locating and correcting visual and singular intersections between B-spline curves for cartographic generalisation is presented. In a first stage, we focus on intersection detection. A hierarchical segmentation of the plane is done and the curves are shared in the cells without any computation, so that the segmentation is fast and reliable. Intersections are computed using a subdivision scheme. The second part deals with correction by deforming the curves respecting cartographic constraints. A first method is introduced where the control polygon is associated with a cable network deformed by the application of external forces. A second method is then presented where the displacement is associated with a snake submitted to energies related to the conflicts. Shape parameters are set automatically.

Keywords : *B-spline curves, intersection, quadtree, cable network, active contours, snake, cartographic generalisation.*