UNIVERSITÉ DE LIMOGES

ÉCOLE DOCTORALE

FACULTÉ DES SCIENCES ET TECHNIQUES

Année : 2017 Thèse N° X

Thèse

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline : Mathématiques

présentée et soutenue par

Pierre BONNELIE

le 13 février 2017

Déformations libres de contours pour l'optimisation de formes et application en électromagnétisme

Thèse dirigée par Paul Armand et Stéphane Bila

JURY :

Jacques-Arthur WEIL	Professeur, Université de Limoges	Président
François JOUVE	Professeur, Université Paris 7, Paris Diderot	Rapporteur
Frédéric MESSINE	Maître de conférences HDR, E.N.S.E.E.I.H.T. Toulouse	Rapporteur
Paul ARMAND	Professeur, Université de Limoges	Membre du jury
Didier AUROUX	Maître de Conférence, Université Nice Sophia Antipolis	Membre du jury
Stéphane BILA	Chargé de recherche HDR, Université de Limoges	Membre du jury
Fabien CAUBET	Maitre de conférences, Université de Limoges	Membre du jury
Olivier RUATTA	Maitre de conférences, Université de Limoges	Membre du jury

Table des matières

Table des figures

Liste des tableaux

Introduction générale

L'objectif de cette thèse est de tester une méthode de déformation par contours libres sur des problèmes d'optimisation de formes. Tous les travaux effectués ont été réalisés en dimension deux où les formes sont des parties du plan et leurs frontières sont des courbes fermées. Celles-ci sont paramétrées par des courbes de Bézier par morceaux, qui sont des courbes polynomiales donc entièrement définies par un nombre fini de coefficients que l'on appelle points de contrôle. C'est en déplaçant ces points que l'on fait évoluer les formes et, dans le cas d'un problème d'optimisation, il faut les déplacer dans une direction de descente afin de converger vers un optimum local. Grâce aux propriétés qu'offrent les courbes de Bézier, il est facile de détecter certains comportements pendant l'évolution des formes. Si au cours de la déformation, une forme a tendance à se diviser en deux, on peut repérer cette portion de courbe et séparer la forme en deux composantes connexes simplement par des opérations sur les points de contrôle. De même si deux formes se rapprochent l'une de l'autre, on dispose d'une méthode pour regrouper leurs points de contrôle et fusionner les deux formes en une. On a appelé *flip* cette méthode qui permet de modifier la topologie des formes de façon dynamique, c'est-à-dire pendant un algorithme d'optimisation.

Plusieurs problèmes d'optimisation de formes ont été considérés afin de tester notre méthode de déformation par contours libres. Le premier problème vient du domaine de l'électromagnétisme et est un travail réalisé en collaboration avec l'équipe MINACOM du laboratoire XLIM. Le problème est la conception d'un composant électrique, un filtre micro-ondes passe-bande. Jusqu'ici, plusieurs méthodes d'optimisation ont été utilisées afin d'améliorer la performance de ces composants. La première est l'optimisation paramétrique qui consiste à changer les dimensions des structures qui composent le filtre. Une seconde méthode est l'optimisation par gradient topologique qui considère le problème comme la recherche de la répartition optimale de matière à l'intérieur du filtre. Notre méthode, de nature à la fois géométrique et topologique, offre une alternative aux méthodes précédentes et le but de notre travail était de voir si elle conduisait à des filtres plus performants.

Nous nous sommes ensuite intéressé à un deuxième problème qui est la détection d'objets immergés dans un fluide. On peut formuler ce problème en un problème d'optimisation de formes où les formes optimales consistent en ces objets immergés. A l'aide d'un algorithme d'optimisation de formes, on est capable de progressivement localiser et déterminer la forme des obstacles inclus dans un plus grand domaine. Nous avons pu appliquer notre méthode de déformation pour détecter jusqu'à deux objets et avons un article publié sous le nom de *Flip procedure in geometric approximation of multiple-component shapes - Application to multiple-inclusion detection* dans le journal *SMAI - Journal of computational mathematics*.

Une troisième application pour notre méthode est les problèmes de trajectoires

optimales. Ce sont des problèmes d'optimisation de formes particuliers car les formes ici sont des trajectoires, des courbes ouvertes reliant deux points de l'espace. Etant donnés deux points A et B, on recherche un chemin reliant ces deux points et minimisant une certaine fonction coût. Un exemple concret de ce genre de problèmes est celui de la navigation où l'on souhaite dépenser le moins d'énergie possible. Un second exemple, théorique lui, est la recherche de chemins pour les méthodes de résolution de systèmes par homotopie. Nous avons utilisé notre méthode utilisant les courbes de Bézier pour approcher les trajectoires sur ces deux exemples particuliers.

La thèse est composée de trois chapitres. Le premier est un chapitre général présentant l'optimisation de formes en donnant cinq méthodes utilisées aujourd'hui pour résoudre ces problèmes. La première méthode est l'optimisation géométrique classique remontant à Hadamard basée sur une représentation explicite des formes et la déformation de la frontière. La deuxième méthode est celle des lignes de niveaux qui se différencie de la précédente par une représentation implicite des formes. La troisième méthode que nous exposons est celle utilisant la théorie de l'homogénéisation où l'idée n'est plus de déplacer une frontière mais de trouver une répartition optimale de matière. La quatrième est la méthode par le gradient topologique qui fait évoluer les formes en créant des trous infinitésimaux ou bien en ajoutant d'infimes quatités de matière. La cinquième méthode que nous avons choisie est une famille d'algorithmes évolutionnaires qui sont de nature très différente par rapport aux quatres méthodes précédentes car visant une optimisation globale. Le deuxième chapitre est le chapitre décrivant notre méthode de déformation par contours libres où nous présentons les courbes de Bézier dans un premier temps, puis la méthode de déformation agissant sur les points de contrôle dans un second temps et enfin le flip affectant la topologie des formes. Le troisième chapitre rassemble les trois problèmes particuliers d'optimisation de formes que l'on a mentionné précédemment. Nous commençons par présenter le problème de conception de filtres micro-ondes passe-bande, puis nous présentons le problème de la détection d'objets immergés et nous terminons avec deux exemples de problèmes de trajectoires optimales.

Chapitre 1 :

Présentation de l'optimisation de formes et des différentes méthodes de résolution

Un problème d'optimisation de formes est défini par trois données : un modèle, un critère et un ensemble de formes admissibles. Le modèle est, la plupart du temps, une équation aux dérivées partielles qui décrit l'état u d'un système physique. La forme Ω à optimiser est le plus souvent un domaine de $\mathbb{R}^2$ ou $\mathbb{R}^3$ où est définie cette équation mais elle peut aussi être seulement inclue dans ce domaine comme par exemple une partie de sa frontière. Le critère ou fonction objectif $\mathcal{J}(\Omega)$ représente une grandeur physique que l'on cherche soit à minimiser soit à maximiser. Par exemple on peut chercher à minimiser le poids d'une structure ou bien minimiser la compliance (travail des forces extérieures) d'une structure élastique soumise à une force. Dans la suite du chapitre, on suppose qu'on cherchera toujours à minimiser $\mathcal{J}$ de sorte à ne pas systématiquement préciser "minimiser ou maximiser". Notons que cela n'est pas restrictif puisque tout problème de maximisation peut se reformuler en un problème de minimisation en considérant l'oppposé du critère. Dans la plupart des cas, le critère ne dépend pas directement de la forme mais indirectement, à travers l'état $u(\Omega)$ du système. L'optimisation se fait sur un ensemble restreint $\mathcal{O}_{ad}$ de formes admissibles non-seulement pour tenir compte de contraintes (par exemple une contrainte de volume à ne pas dépasser) mais aussi pour assurer l'existence d'une forme optimale car dans la grande majorité des cas les problèmes d'optimisation de formes sont mal posés (dans le sens où il n'y a même pas existence d'une forme optimale).

Les méthodes de résolution sont réparties en trois catégories qui sont l'optimisation paramétrique, l'optimisation géométrique et l'optimisation topologique. Précisons que les termes anglais *shape optimization*, que l'on aurait tendance à traduire en *optimisation de formes* en français, désigne en fait l'optimisation géométrique. L'expression *sizing optimization* correspond à l'optimisation paramétrique et l'expression *topology optimization* à l'optimisation topologique. L'optimisation paramétrique est considérée comme la plus simple car les variables d'optimisation sont en général quelques paramètres qui définissent la forme. Par exemple on peut chercher l'épaisseur optimale d'une membrane élastique tendue soumise à un chargement. Dans ce cas le contour de la forme est fixé, seule la hauteur ou l'épaisseur varie au cours de l'optimisation. L'optimisation géométrique, elle, autorise des changements plus prononcés de la forme puisqu'elle consiste à déplaçer sa frontière. Les formes obtenues sont donc plus riches qu'en optimisation paramétrique mais cette méthode ne permet pas toujours d'optimiser la topologie. La troisième catégorie est l'optimisation topologique qui produit les formes les plus générales possibles puisqu'elles modifient la topologie des formes en autorisant l'apparition de trous.

Nous allons présenter cinq méthodes d'optimisation dans ce chapitre. Nous commençons avec l'optimisation géométrique des formes par variation de la frontière. Cette méthode fait partie des premières méthodes d'optimisation de formes et consiste à représenter la forme de façon explicite en échantillonnant sa frontière en plusieurs points. Le domaine est nécessairement maillé lorsqu'on utilise la méthode des éléments finis

pour résoudre l'équation du problème et la frontière de la forme est donc explicitement représentée par une courbe linéaire par morceaux qui fait partie du maillage. La forme évolue en déplaçant les sommets du maillage dans une direction calculée à partir de la fonction objectif. Le principal défaut de cette première approche est l'impossibilité d'introduire des changements topologiques. La seconde méthode que nous présentons est l'optimisation de formes par la méthode des lignes de niveau et permet, elle, de gérer très facilement les changements topologiques. On la classe dans la famille de l'optimisation géométrique car elle consiste toujours à déplacer la frontière des formes. La différence avec la première méthode est dans la représentation des formes, où ici une forme est donnée de façon implicite par une fonction ϕ dont la ligne de niveau zéro coincide avec la frontière de la forme. Le domaine est toujours discrétisé et la forme est stockée sous forme d'un tableau de valeurs de ϕ pour chaque élément du maillage. Chaque maille avec une valeur positive indique que cette maille se trouve à l'intérieur de la forme et chaque maille dont la valeur est négative est à l'extérieur. Puisque les calculs sont exécutés sur la fonction ϕ, les changements topologiques sont naturellement gérés par cette méthode : un changement de topologie correspond à un changement de signe de la fonction ϕ. Cependant les expériences menées montrent que la méthode des lignes de niveau n'est pas capable de créer des trous à l'intérieur des formes (mais elle sait les faire disparaitre). Afin de modifier assurément la topologie des formes, des méthodes topologiques d'optimisation de formes ont été développées et nous en présentons deux, à commencer par la méthode d'homogénéisation. En tant que méthode topologique, la méthode d'homogénéisation considère les problèmes d'optimisation de formes comme des problèmes de répartition de matière dans un domaine. Il s'agit de trouver la répartition de matière $\chi(x)$ en décidant pour chaque point x du domaine que $\chi(x) = 1$ s'il y a de la matière ou $\chi(x) = 0$ sinon. Cependant cette formulation donne lieu à un problème mal posé dans le sens où il n'existe pas de solution. Le principe de la méthode d'homogénéisation est d'élargir l'espace des formes dites classiques, représentables par une fonction caractéristique, à un espace de formes appelées formes composites afin d'avoir un résultat d'existence et d'unicité. Cette technique d'optimisation n'a pour l'instant été appliquée qu'à l'optimisation de structures en mécanique. Il existe d'autres méthodes topologiques pouvant être appliquées à une plus grande famille de problèmes comme la méthode d'optimisation par gradient topologique qui est la quatrième technique d'optimisation que nous présentons. Le principe du gradient topologique en optimisation de formes est d'insérer directement des trous dans les formes. Afin de déterminer les endroits où il est avantageux de retirer de la matière, on dispose du développement asymptotique topologique de la fonction objectif. Nous concluons ce chapitre avec une dernière méthode d'optimisation radicalement différente des précédentes qui utilise les algorithmes évolutionnaires. Elle vise une optimisation globale des problèmes d'optimisation de formes et parcourt l'espace des formes admissibles de façon stochastique

en suivant la théorie d'évolution de Darwin. Les algorithmes évolutionnaires ont l'avantage d'être applicables à un plus grand ensemble de problèmes que les méthodes locales classiques notamment car aucune condition de régularité du critère n'est imposée, mais n'ont aucune garantie de converger.

1.1 Optimisation géométrique par variation de frontière

L'optimisation géométrique de formes consiste à modifier la forme en déplaçant sa frontière. C'est une méthode qui, a priori, offre des formes plus variées qu'en optimisation paramétrique où l'on fait varier quelques paramètres. Cette méthode remonte à l'article pionnier de J. Hadamard [36] en 1907 où il propose de déformer Ω en déplaçant chaque point de sa frontière le long de la normale. Le cadre théorique est dû à F. Murat et J. Simon [48, 49] où ils paramètrent les formes par des difféomorphismes et définissent une notion de différentiabilité par rapport au domaine Ω. Il est ainsi possible de caractériser les formes optimales par des conditions d'optimalité et de calculer le gradient du critère pour mettre en place une méthode numérique d'optimisation.

On suit la présentation de l'optimisation géométrique faite dans [24]. L'idée est de munir l'ensemble $\mathcal{O}_{ad}$ des domaines admissibles d'une structure métrique pour étudier les problèmes de continuité et d'une structure différentielle pour étudier les problèmes de dérivabilité. Notons $W^{1,\infty}(\mathbb{R}^N;\mathbb{R}^N)$ l'espace des fonctions lipschitziennes ϕ de $\mathbb{R}^N$ dans $\mathbb{R}^N$ telles que ϕ et $\nabla\phi$ sont uniformément bornées sur $\mathbb{R}^N$. On munit cet espace de la norme

$$\|\phi\| = \sup_{x\in\mathbb{R}^N} (\mid \phi(x) \mid_{\mathbb{R}^N} + \mid \nabla\phi(x) \mid_{\mathbb{R}^N\times\mathbb{R}^N})$$

où $\mid . \mid_{\mathbb{R}^N}$ est la norme euclidienne vectorielle et $\mid . \mid_{\mathbb{R}^N\times\mathbb{R}^N}$ est la norme euclidienne matricielle. L'espace $(W^{1,\infty}(\mathbb{R}^N;\mathbb{R}^N), \|.\|)$ est alors un espace de Banach. On définit ensuite l'espace de difféomorphismes

$$\mathcal{T} = \{T \text{ tel que } (T - Id) \in W^{1,\infty}(\mathbb{R}^N;\mathbb{R}^N) \text{ et } (T^{-1} - Id) \in W^{1,\infty}(\mathbb{R}^N;\mathbb{R}^N)\}.$$

On choisit les difféomorphismes de la forme $T = Id + \theta$ avec $\theta \in W^{1,\infty}(\mathbb{R}^N;\mathbb{R}^N)$ de norme strictement inférieure à 1. Comme l'illustre la Figure 1.1, chaque forme admissible Ω est obtenue par déformation du domaine initial Ω_0 par un difféomorphisme θ

$$\Omega := (Id + \theta)(\Omega_0) = \{x + \theta(x), x \in \Omega_0\}.$$

Une fois que les formes sont paramétrées par les difféomorphismes θ, on peut définir la

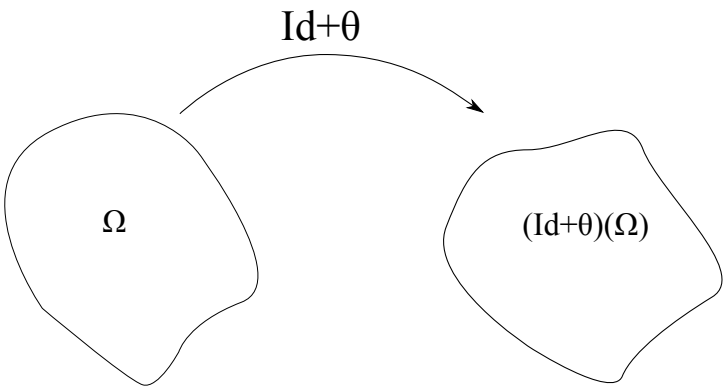


FIGURE 1.1 – Transformation du domaine Ω par le difféomorphisme $Id + \theta$.

différentiabilité de la fonction coût en Ω_0 à partir de la dérivation par rapport à θ : le critère $J(\Omega)$ est dit différentiable par rapport au domaine en Ω_0 si la fonction $\theta \to J((Id+\theta)(\Omega_0))$ est différentiable en 0 dans l'espace de Banach $W^{1,\infty}(\mathbb{R}^N;\mathbb{R}^N)$. On note $J^{'}(\Omega_0)(\theta)$ la dérivée de J en Ω_0 dans la direction θ.

Par exemple, pour les critères du type $J(\Omega) = \int_\Omega f(x)dx$, la dérivée en Ω_0 dans la direction θ vaut

$$J^{'}(\Omega_0)(\theta) = \int_{\partial\Omega_0} \theta(x) \cdot n(x) f(x) ds$$

où $n(x)$ est la normale extérieure en $x \in \partial\Omega_0$. Pour les fonctions objectifs faisant intervenir une fonction $u(\Omega)$ dépendant du domaine, comme par exemple dans le cas de la compliance $J(\Omega) = \int_\Gamma g(x)u(\Omega, x)dx$ avec $\Gamma \subset \partial\Omega$ ou bien dans le cas d'un critère de moindres carrés $J(\Omega) = \int_\Omega \mid u(\Omega, x) - u_0(x) \mid^2 dx$ pour atteindre une cible u_0, l'expression de la dérivée nécessite plus de travail et fait intervenir l'état adjoint p du système. Il est alors nécessaire de résoudre, en plus du problème direct, le problème adjoint pour obtenir la dérivée et cela à chaque itération de l'algorithme d'optimisation. Cependant dans le cas particulier de la minimisation de la compliance, le problème est dit auto-adjoint car l'état adjoint p est égal, au signe près, à l'état u. Ainsi il suffit de résoudre uniquement le problème direct, allégeant alors le temps de calcul.

Les algorithmes d'optimisation sont des algorithmes itératifs, ce qui signifie qu'ils commencent avec une forme initiale donnée par l'utilisateur et cette forme est progressivement améliorée à chaque itération de la boucle d'optimisation. La forme courante est légèrement déformée en utilisant ce qu'on appelle une direction de descente. La méthode du gradient est un cas particulier d'algorithme de descente où la direction est donnée par le gradient de la fonction objectif. Dans le cas de l'optimisation géométrique cela siginifie qu'à l'itération $k+1$, la forme Ω_k est déformée en Ω_{k+1} de la façon suivante

$$\Omega_{k+1} = (Id + \theta_k)(\Omega_k)$$

où θ_k est la direction de descente choisie de sorte que $\mathcal{J}(\Omega_k + 1) < \mathcal{J}(\Omega_k)$. Si l'on possède l'expression analytique du gradient de forme

$$\mathcal{J}^{'}(\Omega)(\theta) = \int_{\partial\Omega} d(\Omega)\theta \cdot n ds$$

où $d(\Omega)$ est une fonction dépendant de l'état $u(\Omega)$ du problème direct et de l'état $p(\Omega)$ du problème adjoint, alors en prenant la direction particulière

$$\theta = -d(\Omega)n$$

la dérivée de forme vérifie $\mathcal{J}^{'}(\Omega)(\theta) < 0$. Ainsi pour un certain $\epsilon > 0$ suffisamment petit, l'approximation au premier ordre de $\mathcal{J}((Id + \epsilon\theta)(\Omega))$ assure

$$\mathcal{J}((Id + \epsilon\theta)(\Omega)) - \mathcal{J}(\Omega) \approx \epsilon\mathcal{J}^{'}(\Omega)(\theta) < 0.$$

Une telle direction $\theta = -d(\Omega)n$ entraine donc une diminution du critère pour peu qu'on effectue un petit pas ϵ dans cette direction.

Un algorithme de descente suivant le gradient possède la structure suivante :

Algorithme du gradient

- Choisir une forme initiale Ω_0
- Tant que le critère d'arrêt n'est pas vérifié,
  - Résoudre le problème direct et le problème adjoint
  - Calculer la direction de descente $\theta = -d(\Omega)n$
  - Déformer la forme courante $\Omega = (Id + \epsilon\theta)(\Omega)$

La première étape est de choisir une forme de départ Ω_0. La forme finale obtenue dépend fortement de cette forme initiale car l'algorithme du gradient est une méthode locale (beaucoup d'algorithmes d'optimisation ont l'inconvénient d'être des méthodes locales) : la forme retournée par l'algorithme est proche d'un minimum local mais on n'a aucun moyen de savoir si c'est un minimum global. On peut seulement changer de forme initiale pour tenter de trouver un meilleur minimum local. Il existe des méthodes d'optimisation globale comme les algorithmes génétiques mais ceux-ci font partie d'une famille bien différente, appelée algorithmes évolutionnaires, et sont présentés à la fin de ce chapitre.

Une fois la forme initiale choisie, l'algorithme entre dans la boucle d'optimisation. Le critère d'arrêt peut prendre plusieurs formes, la plus simple étant de fixer un nombre maximum d'itérations. Une autre condition d'arrêt peut être de calculer la différence entre les deux dernières évaluations de la fonction objectif. Lorsque cet écart est suffisament

petit, les prochaines itérations n'offrent qu'une amélioration négligeable du critère et il vaut mieux stopper l'algorithme. Dans le même principe, on peut calculer la norme du gradient de forme et terminer l'exécution dès cette quantité est inférieure à une valeur seuil.

Chaque itération nécessite de résoudre le problème direct et le problème adjoint pour être capable de calculer la direction de descente θ. On résoud ces deux problèmes par la méthode des éléments finis [40]. On ne va pas présenter cette dernière en détail mais simplement en donner le principe. Le but est de chercher une approximation de la solution $u(\Omega)$ de l'équation au dérivées partielles dans un espace de dimension finie. Cet espace est défini à l'aide d'un maillage du domaine Ω constitué de triangles ou de quadrangles en dimension deux et des tétrahèdres en dimension trois par exemple. La Figure 1.1 montre le maillage d'une console en dimension deux. Une fois le domaine maillé on définit l'espace où l'on va chercher une approximation de la solution $u(\Omega)$ et on en choisit une base, en général polynomiale. Cette approximation est alors une combinaison linéaire des éléments de cette base. Pour déterminer les coefficients de cette combinaison linéaire, l'équation est résolue à chaque noeud du maillage de sorte à définir un problème d'interpolation.

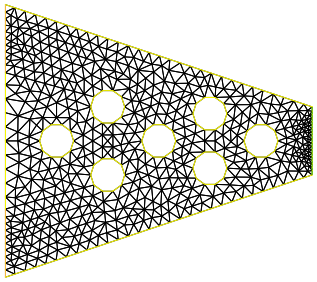


FIGURE 1.2 – Exemple de maillage d'une console en deux dimensions.

Le maillage du domaine Ω fournit une représentation de la forme où sa frontière est une courbe linéaire par morceaux. Cette dernière apparait en effet dans le maillage par un ensemble de sommets reliés par des arêtes. On peut se satisfaire d'une telle représentation par des polynômes de degré un mais on préfère en général une frontière plus régulière et donc utiliser de polynômes de plus haut degré par exemple. Par un changement de variable, ce sont les coefficients des polynômes qui deviennent les variables du problème d'optimisation. Pour mettre à jour la forme courante on doit alors calculer une variation de chaque coefficient de la paramétrisation polynomiale et pour cela, on dispose de deux méthodes. On peut soit calculer directement les dérivées partielles du critère par rapport aux coefficients des polynômes soit appliquer le gradient géométrique en des points de la frontière et interpoler leurs déplacements. La forme est toujours maillée parce qu'il est

nécessaire de résoudre l'équation du modèle par la méthode des éléments finis mais l'on dispose d'une meilleure représentation par des courbes polynomiales de degré plus grand que un (en général les courbes sont cubiques). On verra au chapitre suivant que notre méthode suit ce procédé en utilisant des courbes polynomiales appelées courbes de Bézier. De plus, on proposera un algorithme simple permettant de modifier dynamiquement la topologie de la forme pendant l'optimisation ce qui permet de palier le plus gros défaut de cette première méthode d'optimisation, à savoir son impossibilité à pouvoir changer la topologie.

1.2 Optimisation topologique par la méthode des lignes de niveau

Contrairement à la méthode géométrique précédente qui représente de façon explicite la frontière des formes, l'optimisation de formes par la méthode des lignes de niveau utilise une représentation implicite ce qui lui donne l'avantage de pouvoir facilement gérer les changements de topologie. La forme Ω est représentée par une fonction dont la ligne de niveau zéro correspond à la frontière $\partial\Omega$. La méthode des lignes de niveau n'est pas à proprement parler une méthode d'optimisation mais est une technique de suivi d'interface développée par Osher et Sethian en 1988 [51]. L'interface séparant deux milieux est représentée par la ligne de niveau zéro d'une fonction ϕ. Utilisée dans le cadre de l'optimisation de formes, la méthode des lignes de niveau offre une nouvelle façon de décrire les formes et de suivre leurs évolutions. La correspondance entre une forme Ω et la fonction ligne de niveau ϕ est décrite de la façon suivante

$$\begin{cases} \phi(x) < 0 & \text{si } x \notin \Omega \\ \phi(x) = 0 & \text{si } x \in \partial\Omega \\ \phi(x) > 0 & \text{si } x \in \text{int } \Omega \end{cases}$$

La notion de dérivée de forme provenant des méthodes géométriques d'optimisation s'associe bien à la méthode des lignes de niveau puisqu'elle donne une déformation de la frontière $\partial\Omega$. En utilisant cette dérivée de forme comme direction de déplacement, la méthode des lignes de niveau permet de calculer une nouvelle fonction ϕ de sorte que sa ligne de niveau zéro coincide avec la frontière déplacée.

Dans le cas d'un problème d'optimisation de formes en dimension deux, le graphe de la fonction ϕ est une surface dans $\mathbb{R}^3$ (voir Figure 1.3). L'intersection avec le plan d'équation $z = 0$ donne la fontière du domaine Ω. Un des avantages d'utiliser la méthode des lignes de niveau est que les changements topologiques sont possibles et naturellement gérés car les trous apparaissent ou disparaissent selon les variations de la fonction ϕ. Nous présentons

dans un premier temps la méthode des lignes de niveau et ensuite comment l'utiliser dans un contexte d'optimisation de formes.

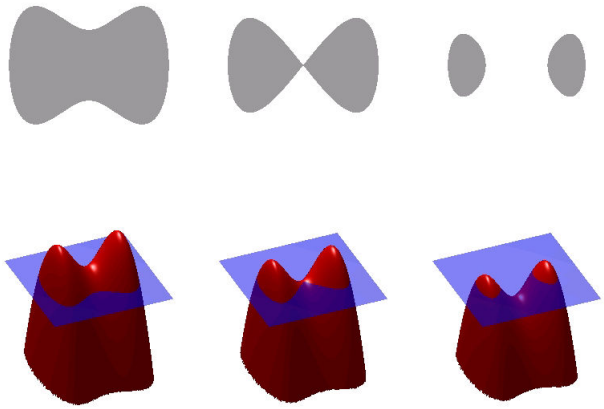


FIGURE 1.3 – Représentation d'une forme en dimension deux par la méthode des lignes de niveau.

1.2.1 Méthode des lignes de niveau

La méthode des lignes de niveau est une technique développé par s. Osher et J.A. Sethian en 1988 [51]. Elle permet de suivre une interface Γ entre deux milieux qui se déplace selon une vitesse F le long de la normale. Au lieu de représenter explicitement par des points sur l'interface et de suivre le déplacement en mettant à jour leur position, l'idée est de considérer l'interface comme la ligne de niveau zéro d'une fonction ϕ. Avant d'utiliser la méthode des lignes de niveau en optimisation, elle fut utilisée dans beaucoup disciplines comme par exemple la segmentation d'image, l'infographie et la mécanique des fluides [60].

De façon générale, considérons une hypersurface $\Gamma(t)$ de dimension N se propageant le long de sa normale n à vitesse F, fonction scalaire de la position $x(t) \in \Gamma(t)$ (voir Figure 1.4). L'interface $\Gamma(t)$ est interprétée comme la ligne de niveau zéro d'une fonction ϕ

$$\Gamma(t) = \{x \in \mathbb{R}^N, \phi(x,t) = 0\}.$$

Le but est d'obtenir une équation décrivant l'évolution de la fonction ϕ au cours du temps t. Soit $x(t) \in \Gamma(t)$, on a alors

$$\phi(x(t),t) = 0.$$

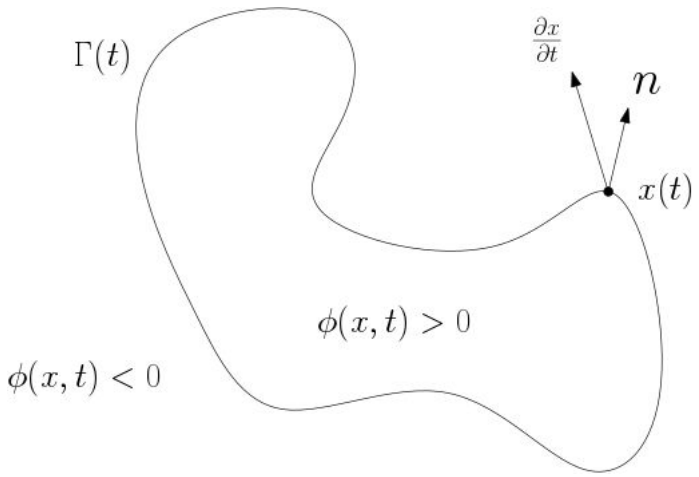


FIGURE 1.4 – Interface evolutive $\Gamma(t)$ se propageant à vitesse F le long de sa normale n.

En dérivant par rapport à t, cela donne

$$\frac{\partial \phi}{\partial t} + < \nabla \phi, \frac{\partial x}{\partial t} >= 0. \tag{1.1}$$

On peut reformuler cette équation en remarquant que, par définition de la vitesse F, $F(x(t)) =< \frac{\partial x}{\partial t}, n >$. De plus, le vecteur $\nabla\phi$ est perpendiculaire à $\Gamma(t)$ donc le vecteur normal s'écrit $n = \frac{\nabla\phi}{|\nabla\phi|}$. En tenant compte de ces égalités, l'équation (1.1) devient

$$\frac{\partial \phi}{\partial t} + F \mid \nabla \phi \mid = 0.$$

C'est une équation standard de Hamilton-Jacobi qui peut être résolue numériquement par la méthode des différences finies ou avec des techniques d'ordres plus élevés [51] sur un maillage structuré. Un point important à noter est que, bien que l'équation d'évolution ne soit définie que sur l'interface, il faut l'étendre au domaine entier pour mettre à jour ϕ sur tout le domaine. Cela signifie que le champ de vitesse F doit être étendu à tout le domaine. La méthode optimale de résolution de cette équation, appelée "Narrow Band Level Set Method", ne résoud l'équation que dans une bande entourant l'interface. Il existe une méthode plus rapide, appelée "Fast Marching", cependant elle requiert que la vitesse F ne change pas de signe. On renvoie à [60] pour plus d'informations sur ces méthodes.

1.2.2 Application à l'optimisation de formes

Les premiers travaux sur l'optimisation par la méthode des lignes de niveau ont été publiés en 2000 [62, 23]. Etant donné que la méthode des lignes de niveau considère la déformation d'une interface sous l'effet d'un champ de vitesse, on voit que la dérivée de forme provenant des méthodes géométriques d'optimisation est particulièrement bien

adaptée ici. En utilisant le gradient de forme on est capable de trouver une direction de descente θ, celle-ci est ensuite utilisée comme champ de vitesse dans l'équation d'Hamilton-Jacobi afin de modifier la forme. Notons que si la plupart des travaux utilisent la dérivée de forme pour déplacer la frontière, ils en existent aussi qui utilisent la dérivée topologique [5, 14]. Cette derivée mesure la variation du critère $\mathcal{J}$ par rapport à l'insertion d'une petite sphère, remplie de vide, à l'intérieur de la forme. L'intérêt d'utiliser un gradient topologique est de forcer le changement de topologie lorsque la méthode des lignes de niveau n'y arrive pas avec le gradient géométrique de forme. La forme retournée en fin d'algorithme est donc moins dépendante de la forme initiale et donc évite de trouver un certain nombre de minima locaux.

Pour commencer l'algorithme d'optimisation, il faut construire une fonction ligne de niveau correspondant à la forme initiale choisie Ω_0. Il faut donc faire en sorte que la ligne de niveau zéro coincide avec la frontière $\partial\Omega$. Une façon est de choisir ϕ comme la fonction de distance signée $\phi(x, t = 0) = d(x, \partial\Omega)$ ce qui permet de commencer l'algorithme avec une fonction ϕ régulière. Ce procédé de "réinitialisation" est répété au cours de l'algorithme pour eviter que ϕ devienne trop plate ou trop pentue. Pour réinitialiser ϕ à une fonction distance on résoud l'équation d'Hamilton-Jacobi suivante

$$\begin{cases} \frac{\partial\phi}{\partial t} + sign(\phi_0)(\mid \nabla\phi \mid -1) &= 0 \\ \phi(x, t = 0) &= \phi_0(x) \end{cases}$$

La solution de cette équation est la distance signée à la ligne de niveau $\{\phi_0(x) = 0\}$ qui correspond à la frontière de la forme courante.

Combinée avec l'optimisation de formes, la méthode des lignes de niveau demande de résoudre l'équation d'évolution d'Hamilton-Hacobi afin de calculer une nouvelle forme. Un problème d'optimisation de formes sous-entend qu'il y a une équation modélisant le système à résoudre. On a donc ici, a priori, deux maillages : un pour la résolution de l'équation décrivant l'état du système et un pour la mise à jour de la fonction ϕ. Un maillage adaptatif est préféré pour la résolution de l'équation du système alors que les méthodes de résolution numérique pour la mise à jour de ϕ utilisent des maillages structurés. Il est alors nécessaire de basculer, à chaque itération, entre les différentes représentation de la forme en remaillant [35]. Dans ce cas, l'utilisation d'un maillage adaptatif permet de représenter avec précision la frontière de la forme et la résolution du problème direct sera d'autant meilleure (Figure 1.5). Cependant le remaillage entraine un coup certain, d'autant plus que cette opération est lancée à chaque itération de l'algorithme d'optimisation. Une approche différente consiste à utiliser un seul maillage structuré pour les deux équations et de représenter la forme par une densité de matière. A chaque maille est associée une valeur de densité : 1 si la maille est à l'intérieur de la forme,

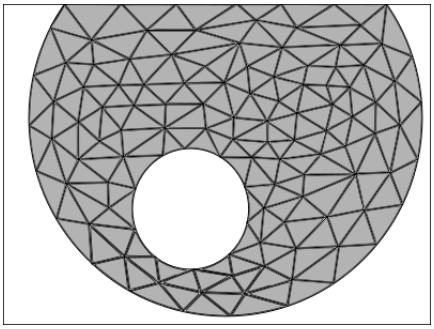


FIGURE 1.5 – Maillage adaptatif utilisé pour résoudre l'équation décrivant l'état du système.

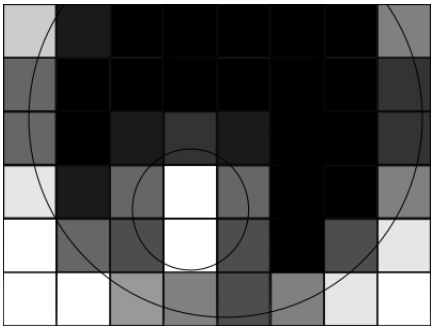


FIGURE 1.6 – Représentation de la forme par des valeurs de densité de matière.

0 si elle est à l'extérieur et une valeur intermédiaire $0 < \theta < 1$ si elle intersecte la frontière (Figure 1.6). Cette méthode permet donc de ne pas avoir à remailler systématiquement mais la frontière n'est pas lisse et les zones de densités intermédiaires posent des problèmes lorsqu'il faut calculer des quantités locales au niveau de la frontière [66].

Comme on l'a précisé plus haut, l'équation d'évolution n'est définie que pour la frontière de la forme. Or il est nécessaire de la résoudre sur tout le domaine de travail afin de conserver une régularité à la fois sur ϕ et sur le maillage. Une méthode d'extension du champ de vitesse est de le définir comme constant dans la direction de la normale. L'avantage est de conserver ϕ comme une fonction de distance signée [62] et donc d'assurer sa régularité. Cependant cela empêche de créer des trous à l'intérieur de la forme. Une seconde méthode profite du fait que, parfois, la dérivée de forme utilise des quantités définies partout. C'est le cas de la compliance notamment [52, 6]. L'avantage est que cette méthode autorise les changements topologiques, mais il est toutefois nécessaire de réinitialiser ϕ comme fonction de distance signée afin de garantir sa régularité.

1.3 Optimisation topologique par la méthode d'homogénéisation

Pour présenter l'optimisation topologique par la méthode de l'homogénéisation, on se place dans le cas de l'optimisation des structures en mécanique. Cette méthode est radicalement différente des deux précédentes qui sont basées sur le déplacement de la frontière des formes. Ici on optimise la distribution de matière χ qui est la fonction caractéristique de la forme : $\chi(x) = 1$ si x est un point de la forme et $\chi(x) = 0$ si x est en dehors de la forme. Ce type de problème est cependant mal posé dans le sens où il n'y a pas de solution. Ceci s'explique par le fait qu'on peut toujours améliorer le critère en retirant de la matière dans des zones de plus en plus petites (autrement dit en formant des trous dans la forme de plus en plus petits). On construit ainsi des suites minimisantes qui tendent vers des formes avec des trous infiniment petits. La limite d'une telle suite n'est pas une forme classique, c'est-à-dire qu'il n'existe pas de fonction caractéristique pouvant représenter cette limite. L'idée est alors d'élargir l'espace des formes en prenant en compte ces formes limites qu'on appelle formes composites. Ces formes sont caractérisées par deux variables $\theta(x)$ et $A(x)$. La première est une généralisation de $\chi(x)$ puisque $\theta(x)$ est la densité de matière en un point x et prend ses valeurs dans tout l'intervalle $[0, 1]$. La seconde variable $A(x)$ représente la microstructure de la forme composite ou la forme des trous. La théorie de l'homogénisation intervient en transformant le problème initial en un problème équivalent qualifié de relaxé ou homogénéisé pour lequel on dispose d'un théorème d'existence et d'unicité de la solution. Elle permet aussi de construire un algorithme numérique pour le calcul de forme optimale. Cependant la forme obtenue à la fin d'un tel algorithme est une forme composite où chaque élément du maillage a une valeur de densité $\theta(x) \in [0, 1]$. Or en pratique on ne sait construire que des formes classiques $\theta(x) \in \{0, 1\}$. C'est pourquoi on fait subir à la forme un étape de pénalisation visant à pénaliser les densités intermédiaires afin d'obtenir des valeurs très proches de 1 ou de 0.

On suit la présentation faite dans [24, 2] où l'auteur se place dans le cas de l'élasticité. Dans un premier temps on pose le problème défini sur l'espace des formes classiques et ensuite on présente la méthode d'homogénéisation dans le cas de strucutres périodiques et les matériaux composites optimaux (laminés séquentiels).

1.3.1 Position du problème original

On dispose d'une membrane élastique dans un domaine Ω. En chaque point $x \in \Omega$ on place un matériau α ou β, le premier étant le plus rigide. On note χ la fonction

caractéristique de la phase α

$$\chi(x) = \begin{cases} 1 \text{ si } x \text{ est un point dans la phase } \alpha \\ 0 \text{ si } x \text{ est un point dans la phase } \beta \end{cases}$$

L'épaisseur h_χ de la membrane en un point x est donnée par

$$h_\chi(x) = \alpha\chi(x) + \beta(1 - \chi(x)).$$

Le déplacement vertical de la membrane est la solution unique $u_\chi(x) \in H_0^1(\Omega)$ de

$$\begin{cases} -\text{div}(h_\chi \nabla u_\chi) &= f \text{ dans } \Omega \\ u_\chi &= 0 \text{ sur } \partial\Omega \end{cases}$$

où $f \in L^2(\Omega)$ est la force appliquée. On souhaite minimiser la fonction objectif suivante

$$\mathcal{J}(\chi) = \int_\Omega j(x, u_\chi(x))dx$$

où $j(x, u_\chi(x))$ est soit la compliance $j(u) = fu$ soit un critère de moindres carrés $j(u) =\mid u - u_0 \mid^2$ pour atteindre un déplacement cible $u_0 \in L^2(\Omega)$. On ajoute une contrainte de volume sur la phase α :

$$0 \leq \int_\Omega \chi(x)dx = V_\alpha \leq \mid \Omega \mid .$$

Le problème d'optimisation de formes est alors

$$\inf_{\chi \in \mathcal{O}_{ad}} \mathcal{J}(\chi)$$

avec $\mathcal{O}_{ad} = \{\chi \in L^\infty(\Omega; \{0, 1\}), \int_\Omega \chi(x)dx = V_\alpha\}$.

Sans contraintes supplémentaires sur les formes admissibles, la fonction objectif ne peut pas atteindre son minimum. Il est en effet toujours avantageux d'introduire des zones de matériau β de plus en plus petites pour diminuer le critère. Le minimum est alors atteint par passage à la limite qui n'est pas une forme classique décrite par une fonction caractéristique mais une forme dite composite, généralisée ou homogénéisée constituée par un mélange des deux phases. L'idée est alors de relaxer le problème en élargissant l'espace des formes admissibles de sorte à inclure ces formes composites. On remplace la fonction caractéristique χ par la fonction de densité de matière θ à valeurs dans l'intervalle $[0, 1]$, représentant toujours la phase α. Une seconde variable A^* est nécessaire pour décrire une forme composite qui est le tenseur de sa loi de comportement effectif correspondant à la microstructure de la forme. On fait appel à la théorie de l'homogénéisation pour relaxer

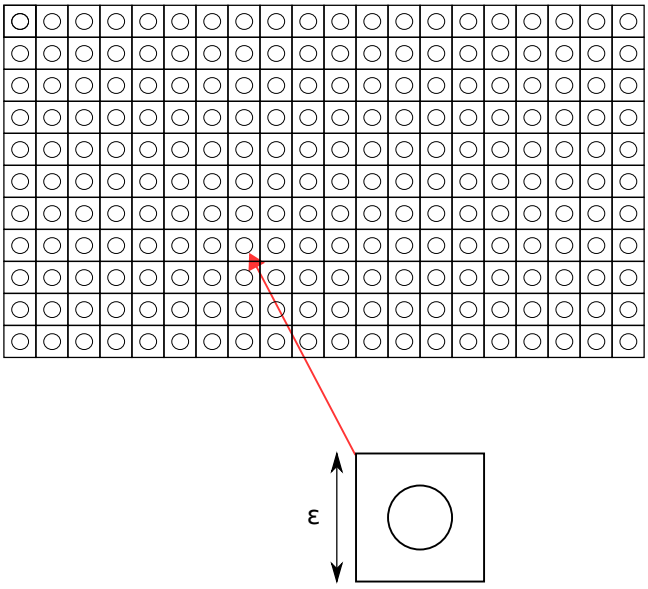


FIGURE 1.7 – Une strucutre périodique en dimension deux.

ou généraliser le problème d'origine en un problème équivalent dans le sens où toute forme optimale du problème relaxé est la limite, au sens de l'homogénéisation, d'une suite minimisante de formes classiques.

1.3.2 Principe de l'homogénéisation

On présente maintenant le principe de l'homogénéisation appliquée au problème d'élasticité précédent. On renvoie toujours à [24, 2] pour des références sur le sujet. La théorie de l'homogénéisation étudie les méthodes de moyennisation dans les équations aux dérivées partielles. Partant d'une équation définie sur un domaine très hétérogène où les coefficients varient rapidement en fonction de la variable d'espace x, l'homogénéisation cherche à établir une autre formulation du problème avec des coefficients moyennés ou homogénéisés. On se restreint ici à l'homogénéisation des structures périodiques mais il existe une théorie de l'homogénéisation dans le cas non-périodique.

Une forme possède une structure périodique lorsqu'elle est constituée d'une cellule de base $Y = [0,1]^N$ répétées dans les N directions d'espace (voir Figure 1.7). La taille de la cellule Y est très petite devant celle de la structure et l'on note $0 < \epsilon << 1$ le rapport entre ces deux tailles.

Le tenseur A est une fonction périodique de période 1 dans chaque direction d'espace e_i :

$$\forall y \in Y, \forall i \in \{1, ..., N\}, A(y + e_i) = A(y).$$

Ainsi la fonction $x \mapsto A(\frac{x}{\epsilon})$ est périodique de période ϵ. On distingue l'échelle microscopique de l'échelle macroscopique car au niveau de la cellule, la fonction A oscille

très rapidement contrairement au niveau global de la structure où les variations de A sont atténuées.

Le problème est alors

$$\left\{ \begin{array}{rcl} -\mathrm{div}(A(\frac{x}{\epsilon})\nabla u_\epsilon) & = & f \text{ dans } \Omega \\ u_\epsilon & = & 0 \text{ sur } \partial\Omega \end{array} \right.$$

qui admet une solution unique $u_\epsilon \in H_0^1(\Omega)$ si $f \in L^2(\Omega)$. Résoudre ce problème par une méthode numérique est irréalisable en pratique car il faut utiliser une discrétisation du domaine où les éléments sont de taille au plus ϵ. On souhaite alors moyenner les propriétés de la structure en définissant un second problème où le tenseur A est un tenseur homogénéisé A^* et calculer une approximation de u_ϵ sur un maillage de taille raisonnable. Pour cela la théorie de l'homogénéisation utilise une technique de développement asymptotique à deux échelles qui est une analyse asymptotique de l'équation précédente lorsque ϵ tend vers zéro. On écrit la solution u_ϵ en série de puissances de ϵ :

$$u_\epsilon = \sum_{i=0}^{+\infty} u_i(x, \frac{x}{\epsilon})\epsilon^i.$$

La variable (lente) x est liée à l'échelle macroscopique et la variable (rapide) $y = \frac{x}{\epsilon}$ à l'échelle microscopique. L'équation du problème homogénéisé est de la forme

$$\left\{ \begin{array}{rcl} -\mathrm{div}(A^*\nabla u) & = & f \text{ dans } \Omega \\ u & = & 0 \text{ sur } \partial\Omega \end{array} \right.$$

où les coefficients de la matrice A^* sont donnés par une formule implicite. Un théorème ([24, Théorème 7.5]) assure que u_ϵ est proche de la solution u lorsque ϵ est petit et on dit que $A(\frac{x}{\epsilon})$ converge au sens de l'homogénéisation vers A^*.

Comme on l'a dit précédemment, en passant du problème original au problème relaxé, on change la variable χ fonction caractéristique de la forme en la fonction de densité θ et on ajoute le tenseur A^* comme variable du problème d'optimisation. Une forme est donc représentée par un couple (θ, A^*) où l'ensemble admissible pour A^* consiste en tous les matériaux composites périodiques fabriqués par mélange des phases α et β en proportions respectives θ et $1-\theta$ (cet ensemble est noté G_θ dans l'ouvrage de référence utilisé). Le problème d'optimisation relaxé se formule donc par

$$\inf_{(\theta,A^*)\in\mathcal{O}_{ad}^*} \mathcal{J}(\theta, A^*) = \int_\Omega j(x, u(x))dx$$

où $\mathcal{O}_{ad}^* = \{(\theta, A^*) \in L^\infty(\Omega; [0,1] \times \mathbb{R}^{N^2}), A^*(x) \in G_{\theta(x)}, \int_\Omega \theta(x)dx = V_\alpha\}$ est l'ensemble

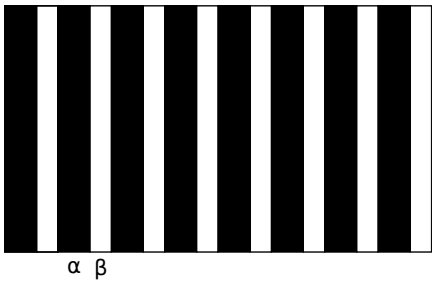


FIGURE 1.8 – Laminé simple consitué par deux matériaux.

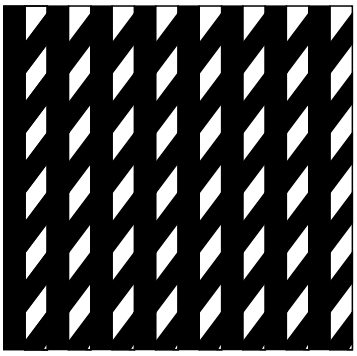


FIGURE 1.9 – Laminé séquentiel de rang deux.

admissible homogénéisé et u la solution du problème homogénéisé. Ce problème possède une solution optimale ([24, Théorème 7.19]) et l'on connait une formule explicite décrivant la microstructure optimale. On montre de plus qu'on peut se restreindre à un sous-ensemble $L_\theta \subset G_\theta$, appelés laminés séquentiels, qui ont des propriétés macroscopiques optimales. Ce sont des matériaux composites fabriqués en empilant de façon successive des couches de phase α et β en proportions respectives θ et $1-\theta$. La Figure 1.8 montre un laminé de rang un dit laminé simple et la Figure 1.9 un laminé séquentiel de rang deux construit de façon récursive à partir d'un laminé simple où l'on ajoute du matériau α dans une direction non-colinéaire à celle du laminé simple. On peut continuer ainsi et construire des laminés de rang n caractérisés par n directions de lamination et n proportions. En pratique on se limite aux laminés simples car ils possèdent la microstructure optimale ([24, Théorème 7.21]).

D'un point de vue pratique cela signifie que l'on a deux variables pour chaque élément du maillage. La première est toujours la densité de matière θ et la seconde est l'angle ϕ de lamination (pour un problème en dimension deux et deux angles en dimension trois). L'angle ϕ remplace le tenseur A^* en pratique car la microstructure optimale est celle des laminés simples définis par l'unique direction de lamination. La méthode d'homogénéisation offre aussi l'avantage de pouvoir dériver par rapport aux variables θ et A^* qui sont continues contrairement au problème original où la variable χ ne prend que des valeurs discrètes. On peut alors utiliser des algorithmes numériques de type descente

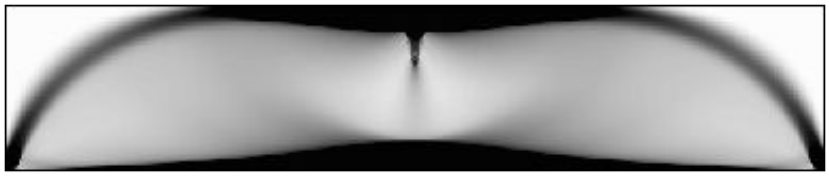


FIGURE 1.10 – Forme composite optimale.

FIGURE 1.11 – Forme classique obtenue après pénalisation.

suivant le gradient. Un algorithme d'optimisation par méthode d'homogénéisation se termine souvent par une phase de pénalisation qui consiste à projeter la forme composite obtenue sur l'espace des formes classiques. Une fois que l'algorithme a convergé vers une forme composite optimale, on continue les itérations tout en pénalisant les zones de densités intermédiaires. Un exemple de pénalisation est le suivant :

$$\theta_{pen} \leftarrow \frac{1 - \cos(\pi\theta)}{2}$$

de sorte que si $0 < \theta < 0.5$ alors $\theta_{pen} < \theta$ et si $0.5 < \theta < 1$ alors $\theta_{pen} > \theta$. La Figure 1.10 montre une forme composite obtenue par la méthode d'homogénéisation sur un problème de poutre optimale. Les zones foncées correspondent aux endroits où la densité est proche de 1 et les zones claires aux endroits où la densité est proche de zéro. On remarque qu'il y a une grande majorité de zones grises correspondant à des valeurs de densité plus proches de 0.5. La forme obtenue après pénalisation est donnée en Figure 1.11 où les zones grises ont effectivement disparues.

1.4 Optimisation par la méthode du gradient topologique

Comme son nom l'indique l'optimisation par gradient topologique est une méthode d'optimisation topologique des formes. Contrairement à la méthode d'homogénéisation qui travaille dans un espace de formes relaxées non-réalisables en pratique, ici l'espace des formes est l'espace classique où une forme est représentée par sa fonction caractéristique $\chi = 0$ ou 1. La méthode d'optimisation par gradient topologique modifie les formes en retirant de la matière (autrement dit en insérant des trous) aux endroits où cela

fait diminuer le plus le critère ou inversement en ajoutant de la matière. Pour cela on dispose d'un développement asymptotique topologique de la fonction objectif indiquant sa variation lors d'un retrait de matière autour d'un point x. Cette idée d'insérer successivement des trous dans la forme provient du fait que les formes optimales en mécanique sont souvent constituées de beaucoup de trous microscopiques. La méthode du gradient topologique n'a pas seulement été appliquée à la mécanique mais aussi à l'électromagnétisme, à la localisation de fissures ou d'inclusions, au traitement d'image, à la mécanique des fluides ...

Le début de la méthode d'optimisation par gradient topologique remonte à l'article en optimisation de structures d'Eschenauer et Schumacher [26] de 1994 et à sa thèse [57]. Les auteurs ont appelé leur technique "bubble method" car leur algorithme d'optimisation comportait une phase où des petits trous circulaires étaient insérés dans la forme en suivant la sensibilité topologique du critère. La boucle d'optimisation consistait en une phase d'optimisation géométrique et cette phase d'optimisation topologique. La phase d'optimisation géométrique permettait de modifier aussi la frontière des trous nouvellement créés.

L'optimisation par gradient topologique repose sur le calcul du développement asymptotique topologique du critère $\mathcal{J}$. La fonction objectif est de la forme $\mathcal{J}(\Omega) = j(u_\Omega)$ où u_Ω est la solution d'une équation aux dérivées partielles sur le domaine Ω et l'on note $B(x, \epsilon)$ la boule de rayon ϵ centrée en $x \in \Omega$. Le développement asymptotique topologique de $\mathcal{J}$ au point x est

$$\mathcal{J}(\Omega \setminus B(x, \epsilon)) = \mathcal{J}(\Omega) + f(\epsilon)g(x) + o(f(\epsilon))$$

avec $\lim_{\epsilon \to 0} f(\epsilon) = 0$, $f(\epsilon) > 0$. La fonction g est le gradient topologique ou dérivée topologique de $\mathcal{J}$ et donne sa variation lorsqu'on ajoute un trou infinitésimal en x.

Après les premiers travaux de 1994, Sokolowski et Zochowski [64] ont généralisé la méthode d'Eschenauer et Schumacher, spécialement conçue pour la minimisation de la compliance, à des fonctions objectifs arbitraires dans le cas d'équations elliptiques et en élasticité dans le plan. Masmoudi a ensuite adapté la méthode de l'adjoint (utilisée pour calculer le gradient géométrique) pour la dérivée topologique en employant la méthode appelée "truncation technique" dans le cas d'une condition de Dirichlet homogène pour des trous circulaires. Les auteurs de [30] se sont intéressés au calcul du gradient topologique pour des fonctions objectifs générales et des trous non nécessairement circulaires dans le cas de l'élasticité linéaire avec condition aux limites de Neumann sur le bord du trou. Considérer des trous de forme abitraire a permi d'utiliser la méthode du gradient topologique pour les problèmes de détection de fissures [8] ainsi que pour des problèmes de traitement d'image [44]. La méthode du gradient topologique a été appliquée à plusieurs

autres problèmes comme l'équation de Poisson [33], l'équation de Navier-Stokes [7], l'équation de la chaleur et l'équation d'onde [9] et l'électromagnétisme [55, 47].

Le déroulement d'un algorithme d'optimisation de formes par gradient topologique consiste à calculer le gradient g par la méthode de l'adjoint et retirer la matière des mailles x où $g(x)$ est négatif et d'en rajouter là où il est le plus positif. L'avantage offert par cette méthode est qu'il suffit de résoudre le problème direct et le problème adjoint une fois pour obtenir g en chaque élément du maillage. Il est fréquent de converger en seulement quelques itérations comme le montre par exemple l'article [8] traitant un problème de détections de fissures où une seule itération permet de localiser toutes les fissures.

1.5 Optimisation globale par les algorithmes évolutionnaires

Nous terminons ce chapitre par la méthode d'optimisation par les algorithmes évolutionnaires. Ces algorithmes ont pour objectif une optimisation globale et sont à différencier des techniques précédentes. Dans les quatres méthodes que l'on vient de présenter, l'évolution des formes est gouvernée par un gradient géométrique ou topologique de la fonction objectif. Ici c'est totalement différent puisque la modification des formes est dirigée par deux principes de la théorie de Darwin. L'algorithme entretient un ensemble fini de formes qu'il met à jour en suivant la sélection naturelle d'une part et les variations aveugles ou non-dirigées d'autre part. Le procédé de sélection naturelle est modélisé par la fonction objectif qui mesure la qualité des formes ou leurs performances. Les variations aveugles elles considèrent les formes comme des chromosomes et consistent en des mutations des gènes et des croisements entre plusieurs individus. Les algorithmes évolutionnaires sont très couteux en terme de temps de calcul mais possèdent plusieurs avantages. Ils visent en effet une optimisation globale et leur champ d'application est plus grand que les méthodes classiques d'optimisation. Il n'y a en effet pas de condition de régularité sur le critère et ils peuvent être définis sur des espaces de recherche non-standard comme les espaces de listes ou de graphes.

L'idée d'utiliser les principes du Darwinisme pour la résolution de problèmes remonte aux années 1950. A cette époque trois algorithmes ont été développés et sont apparus dans les années 1960 : aux Etats-Unis, Fogel [29, 28] développa la programmation évolutionnaire, Holland [38] puis Goldberg [32] créèrent les algorithmes génétiques et en Allemagne Rechenberg et Schwefel [56, 59] inventèrent les stratégies d'évolution. C'est plus tard dans les années 1990 qu'on inventa la famille des algorithmes évolutionnaires qui regroupe ces trois branches. Plus tard une quatrième catégorie, appelée programmation génétique, intégra la famille des algorithmes évolutionnaires.

L'idée commune à ces quatres variations des algorithmes évolutionnaires est de suivre la théorie d'évolution de Darwin selon laquelle une population d'individus évolue selon deux principes. Le premier est celui de sélection naturelle, conséquence de l'environnement qui exerce une pression sur les individus et entraine la disparition des moins performants. Le second est le principe des variations aveugles ou variations non-dirigées du patrimoine génétique qui s'expriment à travers des mutations et des croisements entre individus. Ce principe d'évolution peut se formuler en un problème d'optimisation où la fonction objectif $\mathcal{J}$ mesure la performance de chaque individu. Ainsi il est plus cohérent de formuler ce problème d'optimisation en un problème de maximisation car l'on souhaite maximiser la performance. (**Remarque :** dans les sections précédentes on avait supposé que le problème d'optimisation était formulé en un problème de minimisation).

La terminologie suivante spécifique aux algorithmes évolutionnaires est à mettre en parallèle avec celle de l'optimisation de formes. On a déjà évoqué la fonction objectif $\mathcal{J}$ qui est remplacée par le terme *fonction performance* ou *fonction d'adaptation*. L'espace des formes admissibles $\mathcal{O}_{ad}$ est l'*espace de recherche*, noté E, où chaque élément est appelé *individu*. Le terme de *population* désigne un ensemble de p individus et est notée P_i. La transition d'une population P_i à une population nouvelle P_{i+1} constitue une itération de l'algorithme et on parle de *génération* de la population P_{i+1}.

Partant d'une population initiale P_0, un algorithme évolutionnaire consiste à générer successivement des populations d'individus dont la performance finit par converger vers une valeur maximale. Le processus de génération d'une population P_{i+1} commence par une première étape de sélection des individus les plus performants parmi la population courante P_i. Cette sélection définit un ensemble d'individus que l'on nomme *parents*. Parmi ces parents une deuxième étape se charge d'appliquer les opérateurs de variations non-dirigées que sont les mutations et les croisements afin d'engendrer des *enfants*. Une dernière phase termine la génération de la nouvelle population P_{i+1} en sélectionnant des individus parmi la population courante P_i et les enfants. Le caractère stochastique des algorithmes évolutionnaires est présent dans chaque phase de génération d'une population, aussi bien au niveau de la sélection naturelle des individus qu'au niveau des opérations de croisements et de mutations. Il est cependant possible d'utiliser des méthodes déterministes de sélection afin de conserver la performance d'une population. La structure générale suivie par un algorithme évolutionnaire est la suivante :

1. Initialisation de la population initiale P_0 par tirage aléatoire de p individus
2. Evaluation de chaque individu de P_0 par la fonction performance $\mathcal{J}$
3. (Boucle principale) Génération de la population P_{i+1} à partir de la population précédente P_i
   (a) Sélection des meilleurs individus (parents) au sens de $\mathcal{J}$

(b) Application des opérateurs de variations (mutation et croisement) aux parents pour générer de nouveaux individus (enfants)

(c) Evaluation des enfants par $\mathcal{J}$

(d) Remplacement de la population P_i : définition de la population P_{i+1} par sélection naturelle des enfants et/ou parents

Le critère d'arrêt de la boucle principale peut être lorsque l'on n'améliore plus la fonction performance après plusieurs tours ou lorsque l'on atteint une valeur souhaitée. A cause de la nature stochastique des algorithmes évolutionnaires, la performance n'est pas forcément améliorée à chaque itération. Il existe cependant des stratégies dites "élitistes" qui assurent que le meilleur individu d'une nouvelle population P_{i+1} est plus performant que le meilleur individu de la population précédente P_i.

Un algorithme évolutionnaire a pour but l'optimisation globale sur tout l'espace de recherche et sa réussite dépend dans sa capacité à maintenir une grande diversité génétique parmi la population. La diversité des individus est à favoriser dès l'initialisation et doit être maintenue pendant une longue période. C'est seulement vers la fin que la diversité génétique doit se réduire afin d'assurer la convergence. On dit qu'il y a convergence lorsque la diversité génétique est nulle, c'est-à-dire lorsque les individus de la population courante sont quasimment identiques. On oppose souvent les deux termes d'exploration et d'exploitation dans le domaine des algorithmes évolutionnaires. L'exploration permet à un algorithme d'effectuer une optimisation globale car elle favorise la diversité génétique de la population courante. Un algorithme trop accès sur l'exploration peut cependant échouer à localiser les régions d'individus performants et nuire à la convergence. La phase d'exploitation d'un algorithme évolutionnaire consiste elle à générer des individus proches des individus les meilleurs et restreint donc l'optimisation à un niveau plus local. L'exploitation est favorable à la convergence de l'algorithme mais l'on ne doit pas négliger l'exploration de l'espace de recherche afin d'éviter la convergence prématurée. Un algorithme évolutionnaire doit donc trouver le bon équilibre entre l'exploration de l'espace de recherche et l'exploitation des individus performants.

L'algorithme précédent donne les différentes étapes d'une méthode évolutionnaire et leurs enchainements mais oublie un aspect essentiel et commum à tout algorithme évolutionnaire qui est la représentation des individus. L'espace de recherche initial n'est souvent pas bien adapté pour définir des opérateurs de mutation et de croisement et l'on a besoin d'utiliser une représentation codée des individus pour laquelle il est simple d'appliquer ces opérateurs de variations. Ce procédé correspond à un changement de variable qui part de l'espace de recherche initial (espace phénotypique) vers un espace de représentation des individus (espace génotypique). Des exemples d'espaces génotypiques courament employés sont l'ensemble des chaines binaires $\{0,1\}^m$, un sous-ensemble de

vecteurs à coordonnées réelles $\prod_{i=1}^{m}[a_i, b_i]$ ou l'espace des arbres. Dans le cas d'un problème d'optimisation de formes, l'espace des chaines binaires est souvent utilisé et est directement lié au maillage. Une forme est représentée par un tableau T avec $T[i] = 1$ ou 0 selon que la maille numéro i est à l'intérieur ou en dehors de la forme.

Les opérateurs de variations aveugles que sont les mutations et les croisements sont dépendants de l'espace de représentation des individus. On distingue ces deux opérations non seulement par le fait qu'une mutation n'agit que sur un seul individu (c'est un opérateur unaire) contrairement à un croisement qui requiert au moins deux individus (c'est un opérateur binaire ou n-aire) mais aussi parce que les mutations favorisent l'exploration tandis que les croisements visent l'exploitation. Une mutation et un croisement ont chacun une probabilité p_m et p_c d'être appliqués à chaque individu. Pour donner des exemples, on se place dans le cas où l'espace génotypique choisi est l'ensemble des chaines binaires. Un exemple de mutation est le "bit-flip" qui consiste à échanger un bit avec une faible probabilité. L'opération s'arrête dès qu'un certain nombre (en général un) de bits ont été modifiés. Un exemple de croisement entre deux individus consiste à déterminer une position k dans la chaine binaire, séparant chaque chaine en deux moitiés et à définir ensuite deux nouveaux individus construits en échangeant ces moitiés :

$$\begin{aligned}(x_1, ..., x_m) &\longrightarrow (x_1, ..., x_k, y_{k+1}, ..., y_m)\\ (y_1, ..., y_m) &\longrightarrow (y_1, ..., y_k, x_{k+1}, ..., x_m)\end{aligned}$$

Ce type de croisement est appelé le croisement à un point et se généralise au croisement à n points. Une autre méthode de croisement est le croisement uniforme où chaque bit d'un enfant provient du premier parent ou du second avec probabilité 0.5.

Les opérateurs de variations non-dirigées constitue un des deux principes sur lesquels s'appuient les algorithmes évolutionnaires. Le second principe est celui de sélection naturelle et apparait dans l'étape de sélection et l'étape de remplacement. Ces deux étapes sont indépendantes du choix de représentation des individus, contrairement aux opérateurs de variations aveugles. La différence entre la sélection et le remplacement est qu'un individu peut être sélectionné plusieurs fois, donnant lieu à plusieurs parents identiques, alors que pour l'étape de remplacement un individu est soit sélectionné soit il disparait définitivement. Ces deux étapes utilisent des méthodes communes pour choisir des individus et ne sont basées que sur les valeurs de performance. Une méthode pour sélectionner peut être déterministe où l'on ne garde simplement que les meilleurs individus. Cela permet de conserver à coup sûr les individus les plus performants mais la diversité génétique offerte par les individus éliminés est perdue. Des méthodes probabilistes sont aussi employées comme le tirage par roulette ou la sélection par tournoi, afin de laisser une chance à ces individus moins performants. Le tirage par roulette consiste à associer

une probabilité de sélection proportionnelle à la performance. Les individus performants sont donc plus susceptibles d'être choisis que les individus les moins performants mais ces derniers ont quand même une chance d'être sélectionnés. La sélection par tournoi consiste à répéter plusieurs épreuves où N individus sont tirés au sort et mis en compétition ; le vainqueur étant le plus performant est sélectionné.

Un couple de procédures de sélection et remplacement est appelé un moteur d'évolution. Un algorithme évolutionnaire est défini par le choix d'un moteur d'évolution et un espace de représentation des individus. Les quatres familles d'algorithmes évolutionnaires évoquées au début de cette section sont justement définis par des moteurs d'évolution et des espaces génotypiques particuliers. Les algorithmes génétiques travaillent dans l'espace des chaines binaires et utilisent les mutations de type "bit-flip", les croisements uniformes et le tirage par roulette ; les stratégies d'évolution travaillent elles sur les vecteurs à coordonnées réelles et utilisent des sélections déterministes ; la programmation évolutionnaire concerne les automates à états finis et n'utilisent pas d'opérateurs de croisement ; la programmation génétique concerne la construction automatique de programmes et représente les individus par des arbres d'expressions logiques.

L'optimisation évolutionnaire a été utilisée en optimisation de formes sur des problèmes de mécanique (on renvoie aux références mentionnées dans [24]) et d'électromagnétisme [42]. Les aglorithmes génétiques sont le plus souvent utilisés car la représentation par chaine binaire est bien adaptée pour ces problèmes. Dans le domaine d'optimisation de structures en mécanique, il y a quelques précautions à prendre lorsque l'on souhaite appliquer les opérateurs de croisements. Les deux exemples de croisements donnés précédemment ne sont pas adaptés car ils produisent des formes non-admissibles, produisant des formes avec des parties non connectées. Il y a deux approches pour remédier à ce problème. La première est de remplacer le vide par un matériau très mou ce qui permet d'utiliser les croisements à n points et les croisements uniformes puisque toutes le structures sont admissibles. La seconde est de définir des opérateurs de croisements spécifiques (croisement diagonal et croisement par blocs) respectant cette contrainte topologique.

On termine en évoquant la complémentarité entre les agorithmes évolutionnaires et les méthodes de recherche locales. Les méthodes d'optimisation de formes déterministes locales et les méthodes d'optimisation évolutionnaires globales ne sont pas à mettre en compétition car elles n'ont pas le même but. Ces deux types d'optimisation sont de plus complémentaires et sont couplées dans la pratique pour donner des méthodes plus efficaces notamment avec les algorithmes génétiques hybrides [25]. Un algorithme génétique en tant qu'algorithme évolutionnaire est utilisé pour échantillonner l'espace de recherche et donc recouvrir le plus possible cet espace avec des représentants. Chaque

individu d'une population correspond à une région à explorer et est utilisé comme point initial pour la méthode de recherche locale. Cette dernière permet de calculer le meilleur individu dans chacun des voisinnages des individus initiaux. Cette stratégie tire profit de la complémentarité des méthodes globales et locales car un algorithme génétique sait localiser rapidement la région de l'optimum global mais a du mal à affiner la solution. Et, par définition, une méthode locale saura calculer le meilleur individu dans cette région.

Chapitre 2 :

Courbes de Bézier pour l'optimisation de formes

L'idée d'utiliser des courbes paramétrées pour représenter la frontière des formes n'est pas nouvelle en optimisation de formes. Dans des articles d'optimisation géométrique de formes de 1980, les auteurs ont déjà envisagé et mis en pratique ce procédé permettant d'avoir une représentation explicite et régulière de la frontière. Selon l'article de revue [39] les paramétrisations du type B-Spline sont les plus utilisées. Ce sont des courbes polynomiales définies par des points de contrôle (les coefficients des polynômes) et par un ensemble de noeuds. S'il on ne travaille pas avec une telle représentation mais qu'on se limite à celle donnée par le maillage, les formes ne sont décrites que par les sommets et les arêtes qui délimitent l'intérieur et l'extérieur des formes. Autrement dit on a seulement une paramétrisation linéaire par morceaux de la frontière. Cependant lorsqu'il s'agît de réaliser les formes en pratique, par exemple pour usiner une pièce automobile, les machines qui découpent les pièces sur des plaques métalliques suivent des trajectoires qui sont justement des courbes du type B-Splines. C'est donc une idée pertinente de considérer en amont du processus de réalisation des formes dont la frontière est définie par de telles courbes. Les B-Splines font partie d'une famille plus générale de courbes rationnelles appelées Non-Uniform Rational B-Splines (NURBS) qui sont très utilisées en conception assisté par ordinateur (CAO) car elles permettent de représenter des courbes et des surfaces très complexes à partir d'un jeu de points de contrôle. Leur caractère intuitif est un atout dont les utilisateurs de logiciels de CAO profitent. Il n'est en effet pas nécessaire de connaître la théorie de ces objets mathématiques pour créer les formes voulues : notre intuition suffit pour déterminer à la fois les points de contrôle à déplacer et la direction du déplacement afin de modifier la portion de courbe ou surface désirée.

Pour la méthode proposée ici, nous utilisons des courbes B-Splines particulières qui sont les courbes de Bézier. Le caractère intuitif des Bézier est d'autant plus grand que celui des NURBS ou des B-Splines car le lien entre les points sur la courbe et les points de contrôle est plus simple. Si une courbe B-Spline est définie à partir de points de contrôle et de noeuds, une courbe de Bézier n'utilise que des points de contrôle dans sa définition. Chaque point de la courbe est une combinaison convexe des points de contrôle où les coefficients de la combinaison sont calculés par les polynômes de Bernstein. Une courbe de Bézier est donc un simple polynôme écrit dans la base particulière des polynômes de Bernstein et les coordonnées dans cette base sont les points de contrôle. Le choix de cette base est justement fait pour que les points de contrôle donne une idée de l'allure de la courbe, d'où le caractère intuitif des Bézier. La Figure 2.1 montre une courbe de Bézier cubique définie par quatre points de contrôle. On pourrait étendre la notion de points de contrôle en disant que ce sont les coefficients du polynôme sans préciser la base dans laquelle il est écrit (et donc *point de contrôle* serait interchangeable avec *coefficient du polynôme*). Ainsi on peut se demander à quoi ressemble les points de contrôle de cette même courbe, cette fois-ci exprimée dans la base monomiale. La réponse est illustrée par

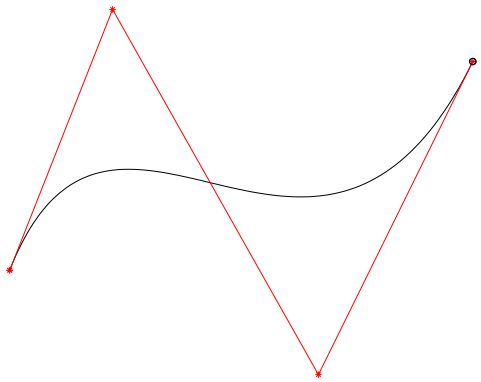


FIGURE 2.1 – Une courbe de Bézier avec ses points de contrôle.

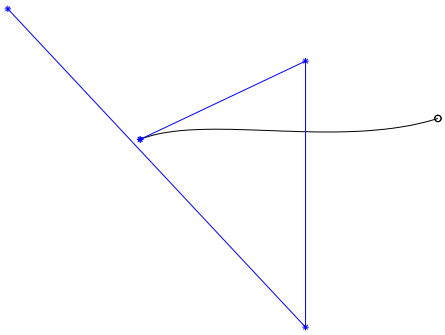


FIGURE 2.2 – La même courbe avec les coefficients (points de contrôle) de la base monomiale.

la Figure 2.2 et on se rend compte qu'il est très difficile de faire le lien entre les points de contrôle et la courbe lorsqu'on exprime le polynôme dans la base monomiale.

Dans ce chapitre nous introduisons dans un premier temps les courbes de Bézier et les courbes de Bézier par morceaux en énonçant quelques-unes de leurs propriétés. Ensuite nous présentons comment les utiliser dans un contexte d'optimisation pour représenter et modifier les formes ainsi qu'une méthode de changement topologique.

2.1 Courbes de Bézier

Dans tout le chapitre on se limite à présenter les courbes de Bézier planes puisque l'on s'est intéressé à des problèmes d'optimisation de formes en dimension deux. Etant donnés $d+1$ points $P_0, ..., P_d$ de $\mathbb{R}^2$, la courbe de Bézier $B([P_0, ..., P_d])$ est l'application définie sur $[0,1]$ à valeurs dans $\mathbb{R}^2$ par

$$B([P_0, ..., P_d])(t) = \sum_{i=0}^{d} P_i b_{i,d}(t) \tag{2.1}$$

où $b_{i,d}(t)$ sont les polynômes de Bernstein

$$b_{i,d}(t) = \binom{d}{i} t^i (1-t)^{d-i}.$$

Dans la suite, on notera $B([P_0, ..., P_d], t)$ plutôt que $B([P_0, ..., P_d])(t)$ pour désigner un point sur la courbe. Les points $P_0, ..., P_d$ sont les *points de contrôle* de la courbe de Bézier. Ils sont généralement reliés successivement par des segments pour former le *polygone de contrôle* qui est une approximation linéaire par morceaux de la courbe. La Figure2.3 montre cinqs courbes de Bézier avec leurs polygones de contrôle. Chaque courbe est obtenue en ajoutant un point de contrôle supplémentaire.

La définition explicite (2.1) est utilisée dans un cadre théorique pour démontrer des propriétés sur les courbes mais n'est pas la meilleure méthode pour calculer des points de la courbe pour des valeurs du paramètre. La méthode qu'utilisait Paul de Faget de Casteljau offre une alternative numériquement plus stable. A partir des points de contrôle P_i et une valeur du paramètre $t \in [0; 1]$, une suite de points est calculée par des interpolations linéaires successives. Le pseudo code de l'algorithme est donné par l'algorithme 1. On se rend mieux compte de l'algorithme avec un dessin comme le montre la Figure 2.4. Dans cet exemple la courbe de Bézier est cubique donc l'algorithme calcule d'abord les points P_0^1, P_1^1, P_2^1 puis P_0^2, P_1^2 et enfin P_0^3 qui est le point sur la courbe.

Algorithm 1 Algorithme de Casteljau

Input: $\mathbf{P_0}, ..., \mathbf{P_d}$, t
Output: $\mathbf{P_0^d}$
  for $k = 1$ **to** d **do**
    for $i = 0$ **to** $d - k$ **do**
      $\mathbf{P_i^k} = (1-t)\mathbf{P_i^{k-1}} + t\mathbf{P_{i+1}^{k-1}}$
    end for
  end for

En pratique on utilise des courbes cubiques car elles sont riches d'un point de vue géométrique. Elles permettent en effet de créer des courbes avec des points d'inflexion, des points de rebroussement. La Figure 2.5 montre les différents cas couverts par les paramétrisations de Bézier cubiques. Les deux cas de la seconde ligne dans la figure ne sont pas intéressants en pratique car l'on veut éviter les auto-intersections et les singularités. Cependant on souhaite avoir des points d'inflexion et comme il n'est pas possible d'en obtenir avec des courbes quadratiques on doit aller jusqu'au degré trois.

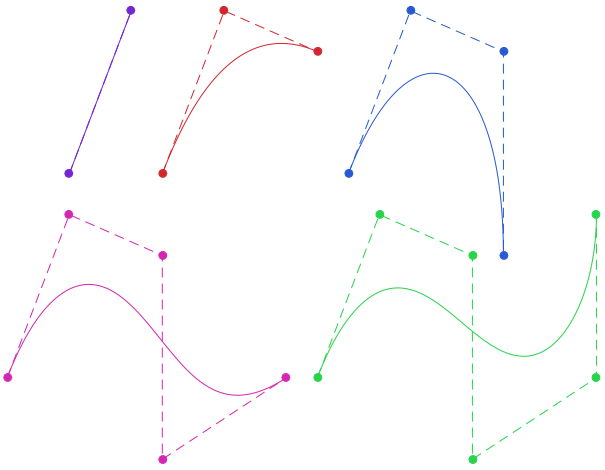


FIGURE 2.3 – Exemples de courbes de Bézier de degré 1 jusqu'à 5.

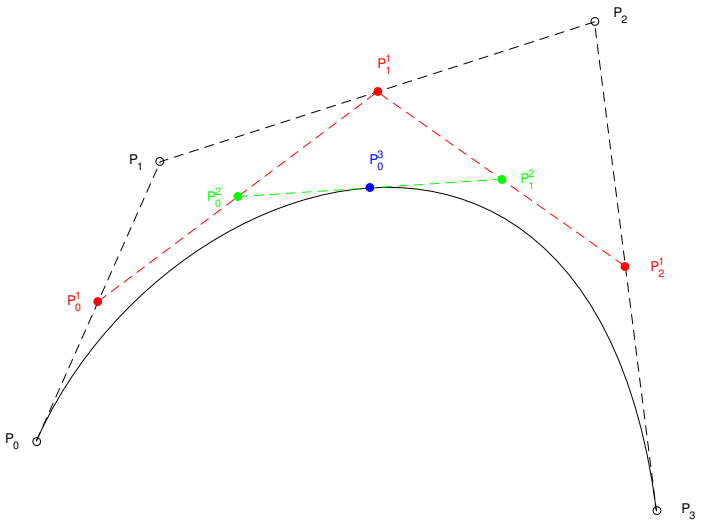


FIGURE 2.4 – Illustration de l'algorithme de Casteljau avec les points intermédiaires.

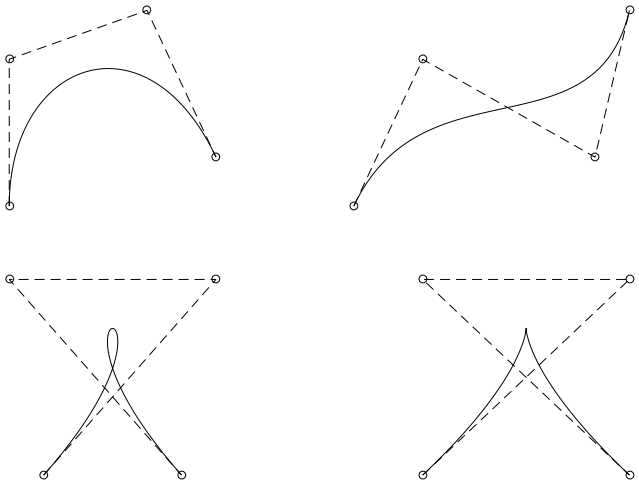


FIGURE 2.5 – Ensemble des situations géométriques offertes par les courbes de Bézier cubiques.

2.2 Propriétés des courbes de Bézier

Cette section énumère quelques propriétés des courbes de Bézier qu'on peut trouver dans [27] avec beaucoup d'autres.

— invariance par transformation affine : appliquer une fonction affine f sur les points de contrôle puis calculer la courbe revient au même que de calculer d'abord la courbe et d'ensuite lui appliquer la transformation f.

$$\forall t \in [0,1], \qquad B([f(P_0),...,f(P_d)],t) = f(B([P_0,...,P_d],t))$$

Il est alors judicieux d'appliquer une transformation affine (par exemple une rotation) sur les points de contrôle lorsqu'on souhaite minimiser le temps de calcul, puisque la transformation f ne sera appelée que $d+1$ fois.

— définition sur un intervalle $[a,b]$: on a présenté les courbes de Bézier avec une paramétrisation définie sur l'intervalle $[0,1]$ mais on peut aussi les définir sur tout intervalle $[a,b]$ $(a < b \in \mathbb{R})$ en considérant l'application suivante

$$\begin{array}{rcccc} t & : & [a,b] & \to & [0,1] \\ & & u & \mapsto & \frac{u-a}{b-a} \end{array}$$

En composant avec cette application t, toute courbe de Bézier peut être définie sur $[a,b]$ de la façon suivante

$$u \mapsto B([P_0,...,P_d],t(u))$$

— propriété de l'enveloppe convexe : la propriété de partition de l'unité pour les polynômes de Bernstein, i.e. $\sum_{i=0}^{d} b_{i,d}(t) = 1$, $\forall t \in [0,1]$, et le fait qu'ils sont tous positifs sur l'intervalle $[0,1]$ entraînent que tout point d'une courbe de Bézier est une combinaison convexe des points de contrôle

$$B([P_0,...,P_d],t) = \sum_{i=0}^{d} P_i b_{i,d}(t)$$

Ainsi toute courbe de Bézier est contenue dans l'enveloppe convexe de ses points de contrôle. C'est une propriété très pratique lorsqu'on souhaite déterminer si

deux courbes s'intersectent. Si tel est le cas alors une condition nécessaire est que les enveloppes convexes de leurs points de contrôle respectifs s'intersectent. En pratique, on utilise la contraposée : si les enveloppes convexes n'ont pas d'intersection alors on est sûr que les courbes ne s'intersectent pas. En revanche si les enveloppes convexes s'intersectent on ne peut pas conclure et il faut regarder plus en détail. Ce procédé est appelé *filtre géométrique* et consiste à utiliser des représentations grossières, plus simples que les courbes afin d'éliminer rapidement les cas où on est sûr qu'il n'y a pas d'intersection. Les objets étant géométriquement plus simples, les algorithmes de détection d'intersection pour ces objets sont plus rapides.

— interpolation des points extrêmes : une courbe de Bézier ne passe en général pas par ses points de contrôle sauf pour ses points extrêmes P_0 et P_d correspondant aux valeurs du paramètre $t = 0$ et $t = 1$. Ainsi il est facile de choisir des points de contrôle pour définir des courbes de Bézier par morceaux continues. Si on a deux jeux de points de contrôle $P_0, ..., P_d$ et $Q_0, ..., Q_e$ on peut définir une courbe par morceaux continue sous réserve d'avoir choisi P_d égal à Q_0. On peut aller plus loin en cherchant à créer un raccordement de classe $\mathcal{C}^1$ en utilisant le fait que les dérivées en $t = 0$ et $t = 1$ valent

$$\begin{array}{rcl} \frac{d}{dt}_{t=0} B([P_0, ..., P_d], t) & = & d\overrightarrow{P_0P_1} \\ \frac{d}{dt}_{t=1} B([P_0, ..., P_d], t) & = & d\overrightarrow{P_{d-1}P_d} \end{array}$$

La condition nécessaire et suffisante pour un raccordement $\mathcal{C}^1$ en $P_d = Q_0$ est alors

$$\frac{d}{dt}_{t=1} B([P_0, ..., P_d], t) = \frac{d}{ds}_{s=0} B([Q_0, ..., Q_e], s)$$

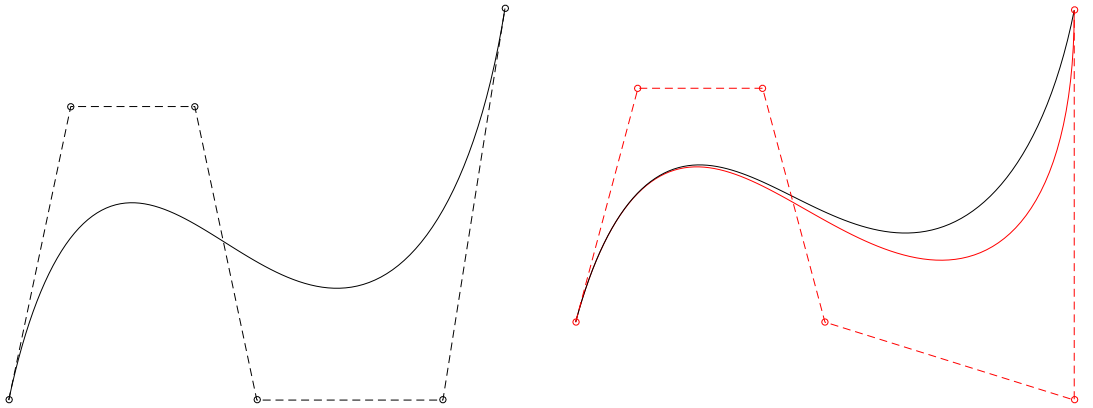
i.e.

$$d\overrightarrow{P_{d-1}P_d} = e\overrightarrow{Q_0Q_1}$$

— symétrie : inverser l'ordre des points de contrôle ne change pas la courbe. En effet, en utilisant le fait que $b_{i,d}(t) = b_{d-i,d}(1-t)$, on montre que

$$B([P_d, ..., P_0], t) = B([P_0, ..., P_d], 1-t).$$

Inverser l'ordre des points de contrôle revient à composer avec la fonction $t \mapsto 1-t$ donnant une reparamétrisation (de Bézier) de la courbe, qui est alors parcourue en sens inverse.



(a) Une courbe de Bézier de degré 5.

(b) La courbe après déplacement du cinquième point de contrôle.

FIGURE 2.6 – Illustration du contrôle pseudo-local des courbes de Bézier.

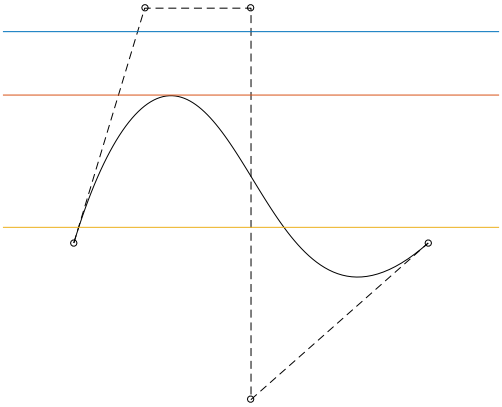


FIGURE 2.7 – Illustration de la propriété de diminution de la variation.

— contrôle pseudo-local : chaque polynôme de Bernstein possède un unique maximum en $t = \frac{i}{n}$. Ainsi en bougeant le point de contrôle P_i, la portion de courbe proche du point de paramètre $t = \frac{i}{n}$ sera la plus impactée (voir Figure 2.6(a) et 2.6(b). Cette propriété montre une fois de plus le côté intuitif des courbes de Bézier.

— diminution de la variation : en deux dimensions, une courbe de Bézier et une droite possèdent moins d'intersections que son polygone de contrôle avec cette même droite (voir Figure 2.7). En dimension trois on a le même résultat avec des plans.

Toutes ces propriétés seront utilisées pour construire un cadre algorithmique permettant d'exploiter les courbes de Bézier pour l'optimisation de formes.

2.3 Courbes de Bézier par morceaux

Pour représenter des courbes géométriquement plus riches avec des paramétrisations de Bézier, une solution serait d'utiliser plus de points de contrôle afin d'augmenter le nombre de degrés de liberté. Cependant cela signifie que le degré de la courbe augmente et il est connu que travailler avec des polynômes de haut degré entraine l'apparition d'oscillations. C'est le cas lorsqu'on cherche à interpoler des points or c'est une technique que l'on utilise dans la méthode de déformation (voir Section 2.4.2). En plus d'oscillations, les calculs sont moins stables numériquement au fur et à mesure que le degré augmente (ceci est dû au mauvais conditionnement de la matrice Vandermonde pour la base de Bernstein [46]). Un autre inconvénient d'utiliser une seule et même courbe de Bézier de haut degré est qu'on ne peut pas réellement modifier localement la courbe puisqu'en bougeant un point de contrôle, c'est théoriquement toute la courbe qui est impactée. Cependant la propriété du contrôle pseudo-local montre que les changements sont plutôt localisés. Pour pallier ces inconvénients, l'idée classique est d'utiliser une définition par morceaux de la courbe, c'est-à-dire raccorder plusieurs courbes de petits degré les unes à la suite des autres. On parle alors de *patch* pour désigner une courbe ou ses points de contrôle. Le fait que la courbe est globalement continue implique que les patches consécutifs partagent des points de contrôle entre eux aux points de raccordement.

Une courbe de Bézier par morceaux est construite comme suit. Soient N courbes de Bézier définies par les points de contrôle $P_0^1, ..., P_{d_1}^1$ pour le premier patch, $P_0^2, ..., P_{d_2}^2$ pour le deuxième, ... et $P_0^N, ..., P_{d_N}^N$ pour le N-ième et vérifiant les conditions de continuité suivantes

$$P_{d_i}^i = P_0^{i+1}, \qquad \forall i \in \{1, ..., N-1\}$$

. On définit la courbe de Bézier par morceaux (continue) par l'application notée $B([P_0^1, ..., P_{d_1}^1], ..., [P_0^N, ..., P_{d_N}^N])$ allant de $[0, 1]$ dans $\mathbb{R}^2$ valant

$$B([P_0^1, ..., P_{d_1}^1], ..., [P_0^N, ..., P_{d_N}^N], t) = B([P_0^i, ..., P_{d_i}^i], Nt - i + 1)$$

avec $i = \lfloor Nt \rfloor + 1$ si $t \neq 1$ et $i = N$ si $t = 1$.

L'expression dans la définition d'une courbe par morceaux signifie que pour tout $t \in [0; \frac{1}{N}]$, la portion de la courbe par morceaux correspond à la courbe de Bézier $B([P_0^1, ..., P_{d_1}^1])$, puis pour tout $t \in [\frac{1}{N}; \frac{2}{N}]$, la courbe de Bézier $B([P_0^2, ..., P_{d_2}^2])$ et ainsi de suite. La Figure 2.8 montre une courbe de Bézier par morceaux construite à partir d'un patch cubique, un patch de degré 5 et un dernier patch de degré 4. Lorsque tous les patches sont de même degré $d = d_1 = ... = d_N$, on dit que la courbe de Bézier par morceaux est uniforme en degré et l'on note $\mathcal{B}_{N,d}$ l'ensemble des courbes de Bézier par morceaux uniformes de degré d composées de N patches. Ce sont ces espaces qu'on utilise

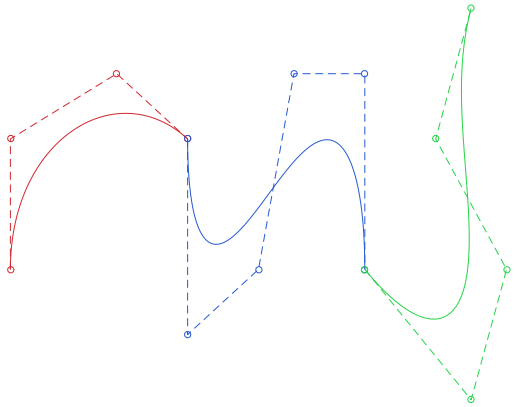


FIGURE 2.8 – Une courbe de Bézier par morceaux composée de trois patches.

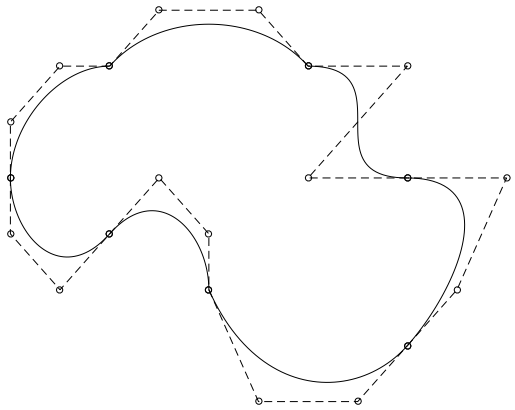


FIGURE 2.9 – Un exemple de forme représentée par une courbe de Bézier par morceaux fermée.

en pratique puisque l'on travaille avec des patches cubiques pour représenter les frontières des formes. La Figure 2.9 montre à quoi ressemble une forme utilisée en optimisation lorsqu'elle est représentée par notre méthode avec des courbes de Bézier.

2.4 Application à l'optimisation de formes

L'utilisation des courbes de Bézier, ou plus précisemment des courbes de Bézier par morceaux, en optimisation de formes consiste à représenter les formes par leur frontière qui est donc paramétrer par une courbe de Bézier. On travaille avec des courbes de Bézier dans le plan constituées par des patches cubiques ($d = 3$) ce qui permet à la fois d'avoir une richesse géométrique suffisante et d'éviter d'avoir des polynômes de trop grand degré, comme expliqué dans la section sur les courbes de Bézier par morceaux. Pour obtenir des formes plus complexes, on laisse alors le degré fixe mais on augmente le nombre de patches N pour avoir plus de degré de liberté. En représentant les formes par des courbes de Bézier, les véritables variables du problème d'optimisation sont alors les points de contrôle (ou plus rigoureusement leurs coordonnées). Le problème d'optimisation de

formes est donc transformé en un problème d'optimisation paramétrique, c'est à dire que l'ensemble des "formes" admissibles est un sous-ensemble de $\mathbb{R}^n$ pour un certain $n \in \mathbb{N}$. Comme la plupart des algorithmes d'optimisation le requiert, il faut trouver un moyen de calculer le gradient du critère. S'il est possible dans certain cas de calculer directement les dérivées partielles par rapport aux coordonnées des points de contrôle $\frac{\partial \mathcal{J}}{\partial P_j^i}$, en général on ne peut pas. Dans ce cas, on verra qu'on peut utiliser le gradient de forme provenant de l'optimisation géométrique de formes pour calculer le gradient du critère par rapport aux points de contrôle.

Dans cette section on commence par présenter ce qu'est l'espace des formes tout en montrant l'intérêt d'utiliser des paramétrisations polynomiales. Ensuite on traitera de la déformation des courbes de façon pratique qui est utilisée dans tout algorithme d'optimisation de formes. Enfin on propose deux méthodes permettant de modifier dynamiquement le nombre de patches d'une courbe ainsi qu'un procédé (algorithme de *flip*) capable de changer la topologie d'une forme.

2.4.1 Espace des formes

Avant de s'intéresser à l'espace des formes, on présente l'idée générale de l'utilisation des courbes de Bézier en optimisaiton de formes. On dispose d'un ensemble de formes admissibles noté $\mathcal{C}$ où chaque élément de cet ensemble est une courbe fermée représentant la frontière d'une forme du plan. On note toujours la fonction objectif par $\mathcal{J} : \mathcal{C} \to \mathbb{R}$ que l'on souhaite minimiser. Parmi les formes appartenant à l'ensemble $\mathcal{C}$ on choisit un sous-ensemble $\mathcal{B} \subset \mathcal{C}$ de courbes de Bézier par morceaux qui est un espace vectoriel sur $\mathbb{R}$ de dimension finie et l'on considère la restriction du critère $\mathcal{J}_{|\mathcal{B}}$ à cet espace. En tant que $\mathbb{R}-$espace vectoriel, $\mathcal{B}$ est donc isomorphe à $\mathbb{R}^n$ pour un certain $n \in \mathbb{N}$. En notant $\phi : \mathbb{R}^n \to \mathcal{B}$ cet isomorphisme, on optimise le critère $\mathcal{J}_{|\mathcal{B}} \circ \phi : \mathbb{R}^n \to \mathbb{R}$, ce qui transforme le problème d'optimisation de formes en un problème d'optimisation paramétrique. On dispose en réalité d'une suite d'espaces $\mathcal{B}_N \subset \mathcal{C}$, croissante pour l'inclusion, vérifiant $\lim_{N\to\infty} \mathcal{B}_N = \mathcal{C}$. En résolvant alors le problème d'optimisation dans l'espace $\mathcal{B}_N$, on utilise la solution comme point de départ pour la résolution du même problème dans l'espace $\mathcal{B}_{N+1}$.

On présente maintenant cet espace des formes $\mathcal{C}$. On suit la présentation faite dans [67] en commençant par définir les courbes représentant les frontières des formes.

Définition 1 *Une courbe (plane paramétrée) est une application continue*

$$m : [a, b] \to \mathbb{R}^2$$

où $a < b \in \mathbb{R}$.

On dit que la courbe est fermée si $m(a) = m(b)$.
Une courbe m *est une courbe de Jordan si elle est fermée et n'a pas d'auto-intersection i.e.* $m(t) = m(s)$ *si et seulement si* $t = s$ *ou* $\{t, s\} = \{a, b\}$.
Une courbe m *est de classe* $\mathcal{C}^1$ *par morceaux si* m *a partout des dérivées finies à gauche et à droite qui coincident sauf en un nombre fini de points.*
L'image d'une courbe m *est l'ensemble* $m([a, b])$ *que l'on note* $\mathcal{R}_m$.

On utilise dans la suite des courbes définies sur l'intervalle $[0, 1]$, comme c'était le cas dans la présentation des courbes de Bézier. Les courbes de Jordan sont celles qui nous intéressent car d'après le théorème du même nom, toute courbe de Jordan sépare le plan en deux parties connexes, l'une étant bornée et l'autre non, chacune des parties ayant pour frontière la courbe de Jordan. La partie bornée constitue donc la partie intérieure de la forme. Parmi les courbes de Jordan on considère les plongements, i.e. les immersions injectives propres et l'on note

$$\mathcal{C} = \{\ m\ :\ [0,1]\ \rightarrow\ \mathbb{R}^2 \text{ immersion injective propre vérifiant } m(0) = m(1)\}.$$

Le problème avec l'ensemble $\mathcal{C}$ est qu'on a pas de représentant unique pour décrire une frontière d'une forme : pour toute courbe $m \in \mathcal{C}$ on peut trouver une autre courbe $m' \in \mathcal{C}$ telle que $m \neq m'$ et $\mathcal{R}_m = \mathcal{R}_{m'}$. Par exemple pour tout difféomorphisme r de $[0, 1]$ dans $[0, 1]$, la courbe $m \circ r$ est une reparamétrisation du même objet géométrique $\mathcal{R}_m$. Le groupe $Diff([0, 1])$ des difféomorphismes de $[0, 1]$ dans lui-même agit sur l'espace vectoriel $\mathcal{C}$ ce qui conduit au quotient $\mathcal{C} \Big/ Diff([0, 1])$. Cet ensemble quotient est une variété [11] où chaque élément est une orbite $m.Diff([0, 1])$ et représente une frontière $\mathcal{R}_m$ d'une forme. On n'a cependant pas de système de représentants pour calculer sur ces courbes, ce qui n'est pas le cas lorsqu'on se retreint aux courbes de Bézier.

Parmi les courbes appartenant à l'ensemble $\mathcal{C}$, on considère les courbes de Bézier fermées $\mathcal{C}^1$ par morceaux. Dans la section précédente on a déjà introduit la notation $\mathcal{B}_{N,d}$ pour désigner les courbes composées de N patches de degré d et on définit ici l'ensemble suivant

$$\mathcal{B}^f_{N,d} = \{\gamma \in \mathcal{B}_{N,d} \text{ telle que } \gamma(0) = \gamma(1)\}.$$

On montre que $\mathcal{B}^f_{N,d}$ est un $\mathbb{R}$-espace vectoriel de dimension $2Nd$ en tant que sous-espace vectoriel des courbes planes fermées. Pour déterminer sa dimension il suffit de compter les points de contrôle, qui sont les coefficients des polynômes écrits dans la base de Bernstein. Pour toute courbe $\gamma \in \mathcal{B}^f_{N,d}$, elle possède N patches de degré d soit *a priori* $N(d + 1)$ points de contrôle. Or le fait que γ est continue et fermée impose que chaque

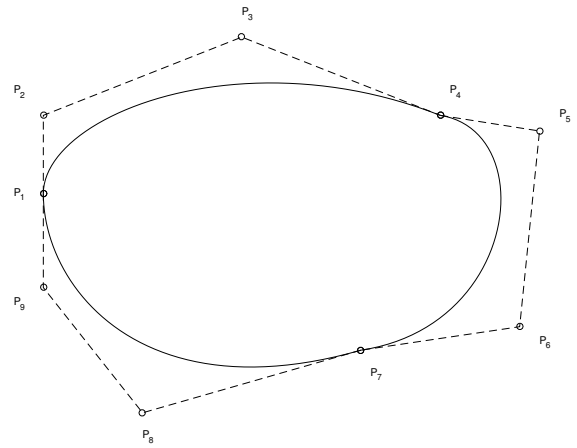


FIGURE 2.10 – Une courbe de Bézier par morceaux fermée composée de trois patches cubiques.

patch partage ses deux points de contrôle extrêmes avec ses patches voisins donc il y a en réalité $N(d+1) - N = Nd$ points de contrôle pour décrire γ. Le fait qu'on travaille avec des courbes à valeurs dans $\mathbb{R}^2$ permet de conclure que la dimension est bien $2Nd$. On note ϕ cet isomorphisme associant à Nd points de contrôle sa courbe de Bézier

$$\begin{array}{rccc} \phi & : & (\mathbb{R}^2)^{Nd} & \to & \mathcal{B}^f_{N,d} \\ & & (P_1, ..., P_{Nd}) & \mapsto & \phi(P_1, ..., P_{Nd}) \end{array}$$

avec $\phi(P_1, ..., P_{Nd})$ étant la courbe de Bézier par morceaux dont les N patches sont $\{P_1, ..., P_{d+1}\}, \{P_{d+1}, ..., P_{2d+1}\}, ..., \{P_{(N-1)d+1}, ..., P_{Nd}, P_1\}$.

Par exemple lorsqu'on travaille avec des courbes composées de trois patches cubiques, donc dans $\mathcal{B}^f_{3,3}$, les trois patches sont $\{P_1, P_2, P_3, P_4\}, \{P_4, P_5, P_6, P_7\}$ et $\{P_7, P_8, P_9, P_1\}$. La Figure 2.10 donne un exemple de courbe appartenant à $\mathcal{B}^f_{3,3}$.

L'avantage de travailler avec des courbes polynomiales est qu'il existe très peu de reparamétrisations qui sont encore polynomiales. Etant donné une courbe $\gamma \in \mathcal{B}^f_{n,d}$, on peut toujours composer avec l'application $t \mapsto 1 - t$ pour obtenir la même image $\mathcal{R}_\gamma$ simplement parcourue dans le sens inverse, donc il y a toujours au moins deux représentants polynomiaux dans une orbite. Cependant comme on l'a vu dans la partie énumérant certaines propriétés des courbes de Bézier, cette reparamétrisation consiste simplement à inverser les points de contrôle. Ces deux représentants sont donc sensiblement les mêmes. Mais il existe d'autres cas où une courbe $\gamma \in \mathcal{B}^f_{n,d}$ possède des reparamétrisations polynomiales qui diffèrent du simple changement d'orientation. Il y a une méthode pour les courbes de Bézier appelée *élévation du degré* [27] permettant d'ajouter des points de contrôle à une courbe sans modifier son image. La nouvelle courbe, bien que de degré d en réalité, possède alors plus de $d+1$ points de contrôle. Cette technique permet d'augmenter le nombre de degrés de liberté tout en laissant la

courbe inchangée. Dans notre cas cela signifie qu'il existe des courbes $\gamma \in \mathcal{B}^f_{n,d}$ dont les patches peuvent avoir des degrés strictement inférieurs à d et donc qu'on peut trouver dans la même orbite que γ d'autres paramétrisations polynomiales utilisant moins de points de contrôle. Cependant l'ensemble des courbes de $\mathcal{B}^f_{n,d}$ utilisant des patches de degré strictement inférieur à d est "petit", car ces courbes utilisant moins de Nd points de contrôle sont donc dans un sous-espace vectoriel de $\mathcal{B}^f_{n,d}$ qui est donc de mesure nulle. Il est donc très peu probable lorsqu'on tire au hasard une courbe dans $\mathcal{B}^f_{n,d}$ qu'elle utilise des patches de degré strictement inférieur à d. Ainsi dans la très grande majorité des cas, pour toute courbe $\gamma \in \mathcal{B}^f_{n,d}$, elle est la seule représentante polynomiale dans son orbite (on ne compte pas la paramétrisation qui consiste à parcourir $\mathcal{R}_\gamma$ dans le sens inverse).

On possède alors un ensemble de formes $\mathcal{C} \Big/ Diff([0,1])$ où on n'a pas de système de représentants pour chaque orbite, sauf si on se restreint aux courbes polynomiales. On propose alors une méthode d'approximation d'une courbe $m \in \mathcal{C}$ par une courbe polynomiale de $\mathcal{B}^f_{N,d}$, qui consiste à évaluer m en Nd points et calculer une courbe polynomiale qui les interpole. On commence par définir l'application d'évaluation ψ suivante

$$\begin{array}{rccc} \psi & : & \mathcal{C} & \to & (\mathbb{R}^2)^{Nd} \\ & & m & \mapsto & (m(t_1), ..., m(t_{Nd})) \end{array}$$

où $\mathbf{t} = (t_1, ..., t_{Nd})$ est la subdivision régulière de l'intervalle $[0,1]$ en Nd points i.e.

$$t_i = \frac{i-1}{Nd}, \text{ pour tout } i \in \{1, ..., Nd\}.$$

On souhaite maintenant interpoler ces Nd par une paramétrisation de Bézier, c'est-à-dire déterminer ses points de contrôle. On introduit alors l'application d'interpolation suivante

$$\begin{array}{rccc} H_{N,d} & : & (\mathbb{R}^2)^{Nd} & \to & (\mathbb{R}^2)^{Nd} \\ & & (M_1, ..., M_{Nd}) & \mapsto & (P_1, ..., P_{Nd}) \end{array}$$

où les points de contrôle $P_1, ..., P_{Nd}$ définissent la courbe $\gamma = \phi(P_1, ..., P_{Nd})$ telle que

$$\psi(\gamma) = (M_1, ..., M_{Nd}).$$

Il faut montrer que l'application $H_{N,d}$ est bien définie en prouvant l'existence et l'unicité de $(P_1, ..., P_{Nd})$. On va d'abord montrer que ce problème d'interpolation consiste en N problèmes d'interpolation faisant intervenir la même matrice, du fait du choix particulier de la subdivision $\mathbf{t}$. On fixe un vecteur $(M_1, ..., M_{Nd}) \in (\mathbb{R}^2)^{Nd}$ et en utilisant la définition d'une courbe de Bézier par morceaux, on remarque

$$\begin{array}{rcccl} M_1 & = & \gamma(t_1) & = & B([P_1,...,P_{d+1}],0) \\ M_2 & = & \gamma(t_2) & = & B([P_1,...,P_{d+1}],\frac{1}{d}) \\ ... & & & & \\ M_d & = & \gamma(t_d) & = & B([P_1,...,P_{d+1}],\frac{d-1}{d}) \\ M_{d+1} & = & \gamma(t_{d+1}) & = & B([P_1,...,P_{d+1}],1) \end{array}$$

Les $d+1$ premiers points $(M_1,...,M_{d+1})$ sont entièrement déterminés par le premier patch $\{P_1,...,P_{d+1}\}$. De la même façon on montre que

$$\begin{array}{rcccl} M_{d+1} & = & \gamma(t_{d+1}) & = & B([P_{d+1},...,P_{2d+1}],0) \\ M_{d+2} & = & \gamma(t_{d+2}) & = & B([P_{d+1},...,P_{2d+1}],\frac{1}{d}) \\ ... & & & & \\ M_{2d} & = & \gamma(t_{2d}) & = & B([P_{d+1},...,P_{2d+1}],\frac{d-1}{d}) \\ M_{2d+1} & = & \gamma(t_{2d+1}) & = & B([P_{d+1},...,P_{2d+1}],1) \end{array}$$

i.e. les points $(M_{d+1},...,M_{2d+1})$ sont entièrement déterminés par le deuxième patch $\{P_{d+1},...,P_{2d+1}\}$, et ainsi de suite. Il y a des équations redondantes entre chaque système du fait que les patches partagent leurs points de contrôle extrêmes avec leurs deux voisins et on peut écrire de deux façons

$$P_{(k+1)d+1} = B([P_{kd+1},...,P_{(k+1)d+1}],1) = B([P_{(k+1)d+1},...,P_{(k+2)d+1}],0) \ .$$

Le problème d'interpolation original à Nd équations est équivalent à N problèmes du type suivant : étant donnés $d+1$ points $N_1,...,N_{d+1} \in \mathbb{R}^2$ distincts deux à deux, trouver les points $Q_1,...,Q_{d+1} \in \mathbb{R}^2$ tels que

$$\left\{ \begin{array}{rcl} B([Q_1,...,Q_{d+1}],0) & = & N_1 \\ B([Q_1,...,Q_{d+1}],\frac{1}{d}) & = & N_2 \\ ... & & \\ B([Q_1,...,Q_{d+1}],\frac{d-1}{d}) & = & N_d \\ B([Q_1,...,Q_{d+1}],1) & = & N_{d+1} \end{array} \right.$$

C'est un problème d'interpolation classique où la matrice B_d du système est une matrice de Vandermonde écrite dans la base de Bernstein

$$B_d \;=\; \begin{bmatrix} b_{0,d}(0) & \dots & b_{d,d}(0) \\ b_{0,d}(\frac{1}{d}) & \dots & b_{d,d}(\frac{1}{d}) \\ \dots & \dots & \dots \\ b_{0,d}(\frac{d-1}{d}) & \dots & b_{d,d}(\frac{d-1}{d}) \\ b_{0,d}(1) & \dots & b_{d,d}(1) \end{bmatrix} .$$

Ceci conclue l'existence et l'unicité des points $(P_1, ..., P_{Nd})$ et le fait que l'application $H_{N,d}$ est bien définie.

On possède maintenant un moyen d'associer à toute courbe $m \in \mathcal{C}$ une courbe de Bézier par morceaux de la façon suivante

$$m \xrightarrow{\psi} \psi(m) \xrightarrow{H} H(\psi(m)) \xrightarrow{\phi} \phi(H(\psi(m))) .$$

Quand on restreint ψ à l'espace $\mathcal{B}^f_{N,d}$ on a un un isomorphisme entre les espaces vectoriels $\mathcal{B}^f_{N,d}$ et $(\mathbb{R}^2)^{Nd}$ par composition d'isomorphismes

$$\psi|_{\mathcal{B}^f_{N,d}} = (\phi \circ H)^{-1}$$

et on a le diagramme commutatif suivant

$$\begin{array}{ccc} (\mathbb{R}^2)^{Nd} & \xrightarrow{\phi} & \mathcal{B}^f_{N,d} \\ & \nwarrow^{H_{N,d}} & \downarrow \psi|_{\mathcal{B}^f_{N,d}} \\ & & (\mathbb{R}^2)^{Nd} \end{array}$$

L'espace $(\mathbb{R}^2)^{Nd}$ sert pour désigner deux espaces différents ici. Le premier, à gauche dans le diagramme, représente l'espace des points de contrôle et le second, en bas à droite, désigne les Nd-uplets de points échantillonnés sur une courbe. C'est pourquoi on préfère introduire les notations $\mathcal{P}_{N,d}$ pour désigner l'espace des points de contrôle et $\mathcal{M}_{N,d}$ celui des points appartenant à la courbe. Le diagramme précédent devient ainsi plus lisible

$$\begin{array}{ccc} \mathcal{P}_{N,d} & \xrightarrow{\phi} & \mathcal{B}^f_{N,d} \\ & \nwarrow^{H_{N,d}} & \downarrow \psi|_{\mathcal{B}^f_{N,d}} \\ & & \mathcal{M}_{N,d} \end{array}$$

2.4.2 Déformation d'une courbe

Dans la section précédente on a montré comment approcher une courbe $m \in \mathcal{C}$ par une courbe de Bézier par morceaux $\gamma = (\phi \circ H_{N,d} \circ \psi)(m)$. On s'intéresse maintenant à la

déformation de cette courbe γ puisque les algorithmes d'optimisaion de formes consistent à déplacer itérativement la frontière. A chaque itération on calcule une déformation $\delta\gamma$ appartenant à l'espace tangent $T_\gamma \mathcal{B}^f_{N,d}$ de $\mathcal{B}^f_{N,d}$ au point γ et l'on met à jour la forme courante par

$$\gamma \leftarrow \gamma + \alpha\delta\gamma$$

où α est un réel, appelé *pas de déformation.*

Or $\mathcal{B}^f_{N,d}$ étant un espace vectoriel, son tangent en tout point $T_\gamma \mathcal{B}^f_{N,d}$ est identifiable à $\mathcal{B}^f_{N,d}$ donc le diagramme de la section précédente existe avec $T_\gamma \mathcal{B}^f_{N,d}$

$$\begin{array}{ccc} \mathcal{P}_{N,d} & \xrightarrow{\phi} & T_\gamma \mathcal{B}^f_{N,d} \\ & \nwarrow^{H_{N,d}} & \downarrow \psi|_{\mathcal{B}^f_{N,d}} \\ & & \mathcal{M}_{N,d} \end{array}$$

Puisqu'il s'agit de déformations de courbe $\delta\gamma \in T_\gamma \mathcal{B}^f_{N,d}$ on peut l'évaluer par ψ pour obtenir un vecteur de déformations

$$\psi(\delta\gamma) = (\delta M_1, ..., \delta M_{Nd})$$

et calculer un vecteur de déformations des points de contrôle

$$H(\delta M_1, ..., \delta M_{Nd}) = (\delta P_1, ..., \delta P_{Nd}).$$

Le fait de disposer du même diagramme lorsqu'on travaille avec une déformation d'une courbe est utile car en pratique, dans un algorithme d'optimisation géométrique de formes, la déformation $\delta\gamma$ est évaluée en quelques points de la frontière (les sommets du maillage) et l'on déplace légèrement ces points dans chaque direction de déformation. C'est-à-dire qu'en pratique les algorithmes manipulent les formes dans l'espace $\mathcal{M}_{N,d}$. Dans notre cas où l'on fait le choix de paramétrer la frontière par des courbes de Bézier, on ne travaille pas dans l'espace $\mathcal{M}_{N,d}$ mais $\mathcal{P}_{N,d}$ puisque l'on ne garde en mémoire qu'un tableau de points de contrôle pour représenter la forme. Et lorsqu'on on souhaite mettre à jour la forme en déplaçant sa frontière en suivant une direction de descente (donnée par exemple par le gradient de forme), on doit calculer une déformation de ces points de contrôle. Un algorithme typique a donc la forme suivante

La partie "déformation" spécifique aux courbes de Bézier dans un algorithme d'optimisation de formes est l'étape d'interpolation $H_{N,d} \times \delta M$ qui permet de passer des déformations des points de la courbe aux déformations des points de contrôle. Cette opération d'interpolation est illustrée par les deux Figures 2.11 et 2.12 où la forme est

Algorithm 2 Algorithme simple de descente

Input: P : variable contenant les points de contrôle d'une forme initiale, α : pas de déformation

Output: P

for $k = 1$ **to** n **do**

$\delta M \leftarrow$ échantilloner une direction de descente

$\delta P \leftarrow H_{N,d} \times \delta M$

$P \leftarrow P + \alpha \delta P$

end for

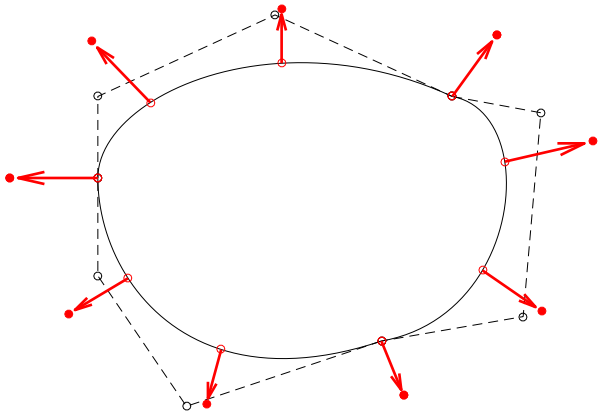


FIGURE 2.11 – Déformations des neuf points de la courbe.

représentée par une courbe de Bézier par morceaux construite avec trois patches cubiques (on est donc dans l'espace $\mathcal{B}^f_{3,3}$).

On a pris l'exemple où les patches sont cubiques car c'est ce qu'on utilise en pratique ; on travaille donc dans les espaces $\mathcal{B}^f_{N,3}$. Comme on l'a montré dans la section précédente, l'interpolation consiste en N problèmes d'interpolation plus petits et dans le cas de patches cubiques chaque problème est de taille 4×4

$$\begin{cases} B([\delta P_1, \delta P_2, \delta P_3, \delta P_4], 0) &= \delta M_1 \\ B([\delta P_1, \delta P_2, \delta P_3, \delta P_4], \frac{1}{3}) &= \delta M_2 \\ B([\delta P_1, \delta P_2, \delta P_3, \delta P_4], \frac{2}{3}) &= \delta M_3 \\ B([\delta P_1, \delta P_2, \delta P_3, \delta P_4], 1) &= \delta M_4 \end{cases}$$

La matrice de ce système est la matrice B_d que l'on a déjà introduite en prenant $d = 3$

$$B_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{8}{27} & \frac{4}{9} & \frac{2}{9} & \frac{1}{27} \\ \frac{1}{27} & \frac{2}{9} & \frac{4}{9} & \frac{8}{27} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Puisque chaque système fait intervenir cette même matrice B_3, en pratique on calcule son inverse $H_3 = B_3^{-1}$ dont l'expression est

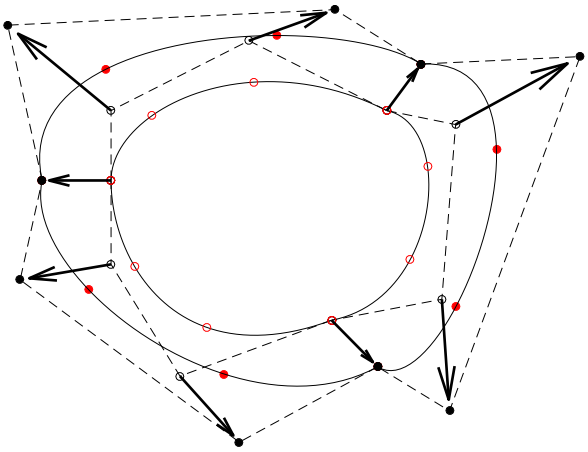


FIGURE 2.12 – Déformations des neuf points de contrôle après interpolation par $H_{3,3}$.

$$H_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{5}{6} & 3 & -\frac{3}{2} & \frac{1}{3} \\ \frac{1}{3} & -\frac{3}{2} & 3 & -\frac{5}{6} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

En prenant l'exemple de la courbe construite avec trois patches cubiques, l'interpolation est alors effectuée par les trois multiplications suivantes

$$\begin{bmatrix} \delta P_1 \\ \delta P_2 \\ \delta P_3 \\ \delta P_4 \end{bmatrix} = H_3 \begin{bmatrix} \delta M_1 \\ \delta M_2 \\ \delta M_3 \\ \delta M_4 \end{bmatrix}$$

$$\begin{bmatrix} \delta P_4 \\ \delta P_5 \\ \delta P_6 \\ \delta P_7 \end{bmatrix} = H_3 \begin{bmatrix} \delta M_4 \\ \delta M_5 \\ \delta M_6 \\ \delta M_7 \end{bmatrix}$$

$$\begin{bmatrix} \delta P_7 \\ \delta P_8 \\ \delta P_9 \\ \delta P_1 \end{bmatrix} = H_3 \begin{bmatrix} \delta M_7 \\ \delta M_8 \\ \delta M_9 \\ \delta M_1 \end{bmatrix}$$

Ceci conclue la présentation de la déformation effectuée en pratique. Cette méthode peut cependant présenter un inconvénient. A l'origine, la déformation $\delta\gamma$ de la courbe est une direction de descente pour la fonction objectif : elle assure qu'en déplaçant chaque point de la fontière on obtient une diminution du critère. Seulement notre méthode n'évalue cette déformation qu'en quelques points de la courbe. La déformation des

points de contrôle obtenue après interpolation n'est plus nécessairement une direction de descente. Une autre idée consiste alors à échantillonner la courbe en un plus grand nombre de points afin d'augmenter l'information donnée par $\delta\gamma$. Le nombre de points de contrôle, lui, reste cependant inchangé et on passe alors d'un problème d'interpolation à un problème d'approximation.

Concrètement on choisit une subdivision plus fine $\mathbf{t} = (t_1, ..., t_m)$ de l'intervalle $[0, 1]$ en m valeurs où $m > Nd$. En prenant Nd valeurs on a vu que cela revient à échantillonner chaque patch en $d + 1$ points, donc on peut par exemple prendre m de sorte à avoir deux fois plus de points par patch et ainsi être plus proche d'une direction de descente. Avec cette nouvelle approche, le calcul de la déformation des points de contrôle n'est plus un problème d'interpolation mais d'approximation puisqu'il y a plus d'équations que d'inconnues : chaque équation est de la forme

$$B(P, t_i) = M_i, \forall i \in \{1, ..., m\}.$$

Dans le cas de l'interpolation, on a montré qu'on peut réduire le problème en N problèmes identiques de tailles réduites. Dans ce cas-ci ce n'est plus possible mais on a toujours l'avantage que la matrice du système est constante pendant tout l'algorithme (puisque le degré et la subdivision sont constants). Cette étape de déformation peut donc être accélérée en précalculant sa factorisation SVD. La section suivante donne deux méthodes qui modifient le nombre de patches de la forme au cours de l'algorithme d'optimisation. En modifiant le nombre de patches, on modifie donc le nombre d'inconnues dans le système précédent et donc il faut recalculer la factorisation SVD de la nouvelle matrice. Cependant ces deux opérations qui augmentent ou diminuent le nombre de patches n'arrivent que rarement pendant une exécution de l'algorithme.

2.4.3 Contrôle de la forme par division et fusion de patches

Pour augmenter les chances de suivre une direction de descente, une méthode serait d'utiliser beaucoup de patches afin d'avoir suffisamment de degré de liberté. Cependant l'intérêt de la méthode est de conserver un nombre restreint de points de contrôle sinon cela revient à utiliser la méthode originale d'optimisation géométrique de formes (on ne veut pas utiliser autant de patches que d'arêtes décrivant la frontière de la forme dans un maillage). A l'initialisation de l'algorithme d'optimisation, on choisit donc un nombre raisonnable de points de contrôle pour représenter la forme initiale. Cependant au fur et à mésure de l'algorithme, la forme peut subir de grands changements ce qui a pour effet d'étirer les patches ou au contraire de les rétrécir. On propose alors deux méthodes pour modifier dynamiquement le nombre de patches au cours de l'algorithme qui permettent de diviser et fusionner des patches. Ces deux opérations sont initiées lorsque la taille d'un

patch est trop grand ou trop petit, mais on peut aussi faire appel à ces deux fonctions lorsque la courbure est trop forte ou trop faible. En effet, dans les zones de forte courbure il est plus intéressant d'avoir beaucoup de patches afin de ne pas passer à côté des détails géométriques et dans le cas des zones à courbure nulle seuls quelques patches suffisent. On nomme ces deux opérations la *division* et la *fusion* de patches. Comme leurs noms l'indiquent, la division partage un patch en deux et la fusion réunit deux patches en un. On pourrait naturellement généraliser ces deux fonctions à plus de deux patches (diviser un patch en n patches et fusionner n patches en un). On présente dans un premier temps la division et dans un second temps la fusion.

2.4.3.1 Division

L'opération de division d'une courbe de Bézier en deux autres possède, comme on le verra, l'avantage de ne pas modifier la courbe originale. Le premier patch résultant de la division décrit la première moitié de la courbe et le second l'autre moitié. On note $P_1, ..., P_d \in \mathbb{R}^2$ les points de contrôle de la courbe originale et on choisit la subdivision régulière de l'intervalle $[0, 1]$ en $2d + 1$ valeurs

$$t_i = \frac{i}{2d}, \qquad \forall i \in \{0, ..., 2d\}.$$

On échantillonne la courbe en ces $2d + 1$ valeurs et on note ces points $M_0, ..., M_{2d}$

$$M_i = B([P_0, ..., P_d], t_i), \qquad \forall i \in \{0, ..., 2d\}.$$

On utilise une seconde subdivsion régulière de l'intervalle $[0, 1]$ en $d + 1$ valeurs

$$s_i = \frac{i}{d}, \qquad \forall i \in \{0, ..., d\}.$$

La division du patch $P_0, ..., P_d$ consiste à déterminer les points de contrôle $Q_0, ..., Q_d$ et $R_0, ..., R_d$ vérifiant

$$\begin{aligned} B([Q_0, ..., Q_d], s_i) &= M_i, \qquad \forall i \in \{0, ..., d\} \\ B([R_0, ..., R_d], s_i) &= M_{i+d}, \qquad \forall i \in \{0, ..., d\}. \end{aligned}$$

Ce sont deux problèmes d'interpolation, que l'on a déjà rencontré dans la section

précédente. En utilisant les notations matricielles suivantes

$$\begin{array}{rclcrcl} Q & = & \begin{bmatrix} Q_0 \\ \vdots \\ Q_d \end{bmatrix} & & M^{(0)} & = & \begin{bmatrix} M_0 \\ \vdots \\ M_d \end{bmatrix} \\ R & = & \begin{bmatrix} R_0 \\ \vdots \\ R_d \end{bmatrix} & & M^{(1)} & = & \begin{bmatrix} M_d \\ \vdots \\ M_{2d} \end{bmatrix} \end{array}$$

on peut réécrire les équations précédentes en les deux systèmes suivants

$$\begin{aligned} B_d Q &= M^{(0)} \\ B_d R &= M^{(1)} \end{aligned}$$

où B_d est la matrice de la section 2.4.1, i.e. la matrice d'évaluation d'un patch de degré d pour la subdivision régulière $\mathbf{s} = (0, \frac{1}{d}, ..., 1)$. Puisque l'on a dit que l'on calculait une fois pour toute son inverse H_d, on résoud ces deux systèmes en utilisant cette dernière. Les deux patches solutions sont donc

$$\begin{aligned} Q &= H_d M^{(0)} \\ R &= H_d M^{(1)}. \end{aligned}$$

Avant de résumer l'opération de division par un algorithme, on introduit la matrice suivante

$$B^{div} = \begin{bmatrix} b_{0,d}(t_0) & \dots & b_{d,d}(t_0) \\ b_{0,d}(t_1) & \dots & b_{d,d}(t_1) \\ \dots & \dots & \dots \\ b_{0,d}(t_{2d}) & \dots & b_{d,d}(t_{2d}) \end{bmatrix}$$

utilisée dans l'algorithme pour évaluer la courbe en les $2d+1$ valeurs de la subdivision. En pratique le degré d est constant donc cette matrice aussi est calculée une fois pour toute et est, comme la matrice H_d, une variable globale dans l'algorithme 3.

En continuant toujours avec des exemples pour des courbes cubiques puisque c'est ce qu'on utilise en pratique, la Figure 2.13 illustre la division d'un patch. Comme le montre cette figure, lorsque l'on divise un patch en deux, les deux patches produisent la même courbe que le patch original. Le patch $Q_0, ..., Q_d$ trace la première moitié et le patch $R_0, ..., R_d$ la seconde. En effet, montrons cela pour le premier patch $Q_0, ..., Q_d$. La première moitié de courbe est un polynôme de degré d, de même que le patch $Q_0, ..., Q_d$.

Algorithm 3 Algorithme de division d'un patch

Input: $P = [P_0, ..., P_d]$
Output: $Q = [Q_0, ..., Q_d]$, $R = [R_0, ..., R_d]$
$M \leftarrow B^{div} P$
$M^{(0)} \leftarrow M[0..d]$
$M^{(1)} \leftarrow M[d..2d]$
$Q \leftarrow H_d M^{(0)}$
$R \leftarrow H_d M^{(1)}$

Or puisque ce dernier interpole $d+1$ points sur la première moitié de courbe, ces deux polynômes sont nécessairement égaux. De même pour le patch $R_0, ..., R_d$ et la seconde moitié.

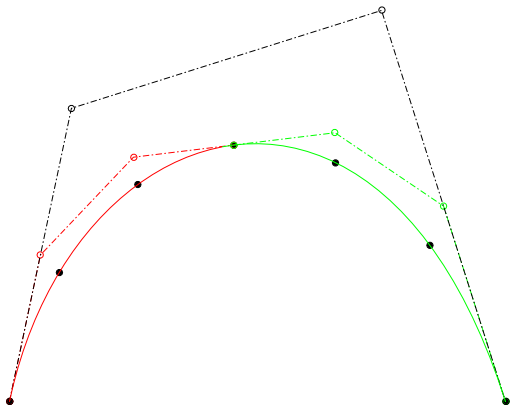


FIGURE 2.13 – Division d'un patch cubique.

2.4.3.2 Fusion

On présente maintenant l'opération inverse de la division qui est la fusion. La fusion agit sur deux patches consécutifs de degré d et en crée un seul de degré d également. Dans ce cas, la courbe décrite par ce nouveau patch est en général différente de celle décrite par les deux originaux. En utilisant les notations similaires, on note $Q_0, ..., Q_d$ et $R_0, ..., R_d$ les points de contrôle. Puisque ces patches sont consécutifs ils partagent le même point $Q_d = R_0$. L'opération de fusion de patches se résume, comme on le verra, à une opération d'interpolation comme pour la division d'un patch. Les points à interpoler sont définis par des valeurs particulières t_i et $u_j \in [0, 1]$, définies à partir de la subdivision régulière $s = (s_i)_{0 \leq i \leq d)}$ où $s_i = \frac{i}{d}$. On définit ces valeurs par

$$\begin{aligned} t_i &= 2s_i, \qquad \text{si } 0 \leq s_i \leq \frac{1}{2} \\ u_i &= 2s_i - 1, \qquad \text{si } \frac{1}{2} \leq s_i \leq 1. \end{aligned}$$

On échantillonne les deux patches en ces valeurs pour définir les points M_i à interpoler

$$M_i = \begin{cases} B([Q_0, ..., Q_d], t_i) & \forall i \in \{0, ..., \lfloor \frac{d}{2} \rfloor\} \\ B([R_0, ..., R_d], u_i) & \forall i \in \{\lfloor \frac{d}{2} \rfloor + 1, ..., d\}. \end{cases}$$

La fusion des deux patches consiste alors à trouver les points de contrôle $P_0, ..., P_d \in \mathbb{R}^2$ tels que

$$B([P_0, ..., P_d], s_i) = M_i, \qquad \forall i \in \{0, ..., d\}.$$

Une fois de plus, comme pour la division d'un patch, les points $P_0, ..., P_d$ solutions sont calculés grâce à la matrice d'interpolation H_d. En utilisant des notations évidentes, on a

$$P = H_d M.$$

Comme on l'a fait pour la division, on donne par l'algorithme 4 un résumé des étapes de la fusion. L'algorithme utilise les deux matrices suivantes pour calculer les points $M_0, ..., M_d$

$$B^{fus,t} = \begin{bmatrix} b_{0,d}(t_0) & \dots & b_{d,d}(t_0) \\ \dots & \dots & \dots \\ b_{0,d}(t_{\lfloor \frac{d}{2} \rfloor}) & \dots & b_{d,d}(t_{\lfloor \frac{d}{2} \rfloor}) \end{bmatrix}$$

$$B^{fus,u} = \begin{bmatrix} b_{0,d}(u_{\lfloor \frac{d}{2} \rfloor + 1}) & \dots & b_{d,d}(u_{\lfloor \frac{d}{2} \rfloor + 1}) \\ \dots & \dots & \dots \\ b_{0,d}(u_d) & \dots & b_{d,d}(u_d) \end{bmatrix}.$$

Comme c'était le cas avec la matrice B^{div} dans l'algorithme de division, ces deux matrices sont des variables globales dans l'algorithme 4.

Algorithm 4 Algorithme de fusion de deux patches

Input: $Q = [Q_0, ..., Q_d]$, $R = [R_0, ..., R_d]$
Output: $P = [P_0, ..., P_d]$
$M^{(0)} \leftarrow B^{div,t} Q$
$M^{(1)} \leftarrow B^{div,u} R$
$M \leftarrow [M^{(0)}; M^{(1)}]$
$P \leftarrow H_d M$

On propose également un exemple avec des patches cubiques $Q_0, ..., Q_3$ et $R_0, ..., R_3$, où $Q_3 = R_0$. La subdivision s est $s = (0, \frac{1}{3}, \frac{2}{3}, 1)$ et on pose

$$\begin{array}{lllllllll} t_0 & = & 2s_0 & = & 0, & t_1 & = & 2s_1 & = & \frac{2}{3} \\ u_2 & = & 2s_2 - 1 & = & \frac{1}{3}, & u_3 & = & 2s_3 - 1 & = & 1. \end{array}$$

Les quatres points à interpoler sont

$$\begin{aligned} M_0 &= B([Q_0, Q_1, Q_2, Q_3], 0), & M_1 &= B([Q_0, Q_1, Q_2, Q_3], \tfrac{1}{3}) \\ M_2 &= B([R_0, R_1, R_2, R_3], \tfrac{2}{3}), & M_3 &= B([R_0, R_1, R_2, R_3], 1). \end{aligned}$$

La Figure 2.14 illustre cette opération, où le patch Q_0, Q_1, Q_2, Q_3 est en rouge et R_0, R_1, R_2, R_3 en vert. On voit clairement que la nouvelle courbe est, *a priori*, très différente de l'originale. Cependant l'effet est accentué par l'échelle et en pratique, puisque une fusion est déclenché lorsque les patches Q_0, Q_1, Q_2, Q_3 et R_0, R_1, R_2, R_3 sont de petites tailles, la portion de courbe modifiée est relativement courte. Donc la différence entre la courbe avant la fusion et après est seulement légère.

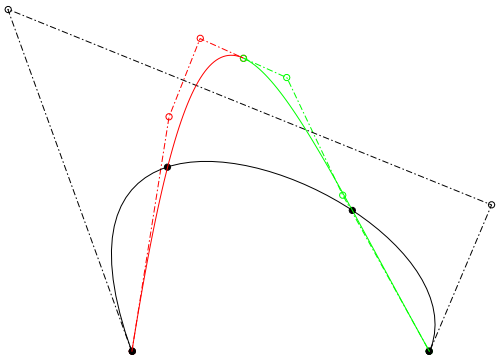


FIGURE 2.14 – Fusion de deux patches cubiques.

2.4.4 Changement topologique : algorithme flip

Bien que l'on utilise une méthode géométrique d'optimisation de formes en déplaçant les points de contrôle, on peut tirer parti de la représentation des formes par les courbes de Bézier afin de modifier leurs topologies. On propose une méthode capable de diviser une forme en deux ainsi que de réunir deux formes en une. Jusqu'à maintenant on a utilisé le terme de *forme* pour décrire un domaine de $\mathbb{R}^2$ dont la frontière est une courbe de Bézier par morceaux sans auto-intersection, ce qui sous-entend que la forme est connexe. Puisque l'on introduit ici des changements topologiques modifiant le nombre de composantes connexes, dorénavant lorsque l'on parle d'une forme, il faut l'imaginer avec plusieurs composantes connexes, où chacune possède une frontière appartenant à un espace $\mathcal{B}^f_{n,d}$. Encore une fois, en pratique $d = 3$ pour toutes les composantes mais le nombre de patches varie selon chaque composante. A chaque tour de l'algorithme d'optimisation, chaque composante est inspectée et un changement topologique est déclenché lorsque deux parties de courbes sont très proches l'une de l'autre. Si ces deux portions de frontière appartiennent à la même composante connexe, alors cette dernière sera divisée en deux. Par contre, si ces deux parties de courbes appartiennent à des composantes différentes

elles seront fusionner. On peut rapidement détecter les endroits où deux parties de frontière sont proches grâce aux courbes de Bézier. Les patches composant les frontières forment un partitionnement et l'on détecte rapidement si deux d'entre eux sont proches en cherchant les collisions entre les polygones de contrôle. S'il y a une collision, un algorithme, que l'on nomme *flip*, réorganise les points de contrôle de ces deux patches ce qui conduit à un changement topologique (une séparation de composante ou une réunion de deux composantes). Ce procédé de changement topologique s'articule en deux étapes qui s'exécutent à chaque tour de la boucle d'optimisation. La première étape est celle de la détection des patches trop proches ou plus précisément la détection de collisions entre deux polygones de contrôle. La seconde étape est l'algorithme de flip qui intervient s'il y a effectivement une collision entre deux polygones. Avant de présenter ces deux étapes en détail, on énonce deux hypothèses qui doivent être satisfaites à chaque tour de l'algorithme.

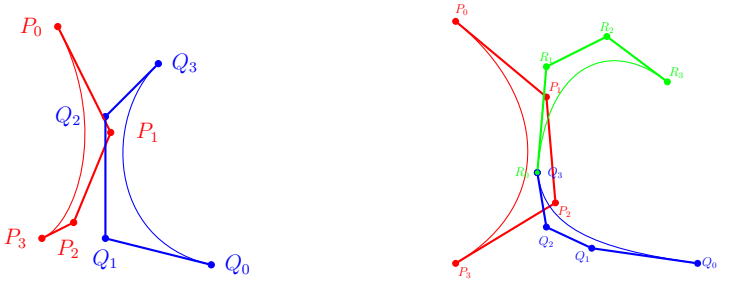
Afin d'être dans de bonnes conditions à la fois pour la détection de collisions et le flip, on requiert les deux hypothèses suivantes :

— la taille des polygones de contrôle (le diamètre de leur enveloppe convexe) reste dans un intervalle $[S_{\min}, S_{\max}]$,

— le pas de déformation α est très petit devant $S_{\min}$.

La première hypothèse permet à la fois de maintenir une stabilité numérique, en évitant de travailler avec des patches très grands et très petits en même temps, mais aussi d'éviter que plusieurs petits patches consécutifs aient leurs polygones de contrôle intersectant un même polygone de contrôle d'un grand patch. L'algorithme de flip est en effet conçu pour gérer deux cas : deux polygones de contrôles s'intersectent ou un polygone intersecte deux polygones consécutifs (voir Figure 2.15). En respectant cette première condition sur la taille des patches, on suppose que seuls ces deux cas se produisent en pratique. Même si théoriquement il peut se produire des collisions plus complexes, dans la grande majorité des cas ce sera l'une des deux situations considérées. Notez qu'il est facile de respecter cette première hypothèse en utilisant les fonctions de division et fusion de patches permettant de contrôler la taille des patches.

La seconde hypothèse concerne le pas de déformation $\alpha \in \mathbb{R}$ utilisé dans l'algorithme d'optimisation. A chaque tour, les points de contrôle $\mathbb{P}$ sont modifiés dans une direction de descente d par $\mathbb{P} \leftarrow \mathbb{P} + \alpha d$. En choisissant un pas trop grand, on risque de déformer top violemment la frontière jusqu'à créer des auto-intersections. Si par contre le pas est petit, on sera capable de prévenir ces intersections et d'agir avant qu'elles ne se produisent en scindant ou réunissant les composantes connexes. Cette seconde condition est illustrée par la Figure 2.16. Maintenant que l'on a présenté ces deux hypothèses, on peut passer à



(a) Cas avec deux patches.

(b) Cas avec deux patches.

FIGURE 2.15 – Les deux situations gérées par le flip.

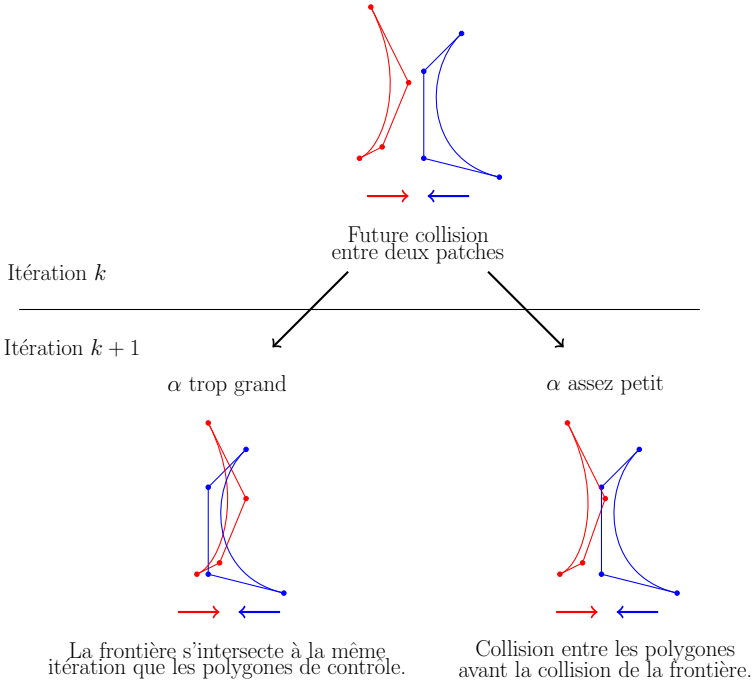


FIGURE 2.16 – Le pas de déformation α est choisi petit devant $S_{\min}$.

l'étape de détection des collisions entre polygones de contrôle.

Vérifier la collision entre deux polygones de contrôle est une tâche qui peut être globalement coûteuse s'il on calcule directement les intersections entre les segments des polygones. En effet, il faut se rappeler que la détection a lieu à chaque tour de l'agorithme et que le nombre de patches peut être élevé puisque la forme est constituée de plusieurs composantes connexes. La détection de collisions est donc une opération fréquemment appelée, c'est pourquoi on utilise pour un filtre géométrique afin d'accéler cette étape. La détection de collisions entre polygones de contrôle commence par calculer des approximations grossières des polygones sous forme de rectangles. Ces rectangles sont appelés AABB (pour Axis-Aligned Bounding Boxes) et sont les plus petits rectangles englobant les polygones dont les côtés sont parallèles aux axes d'un repère global (voir

Figure 2.17). Une condition nécessaire pour que deux polygones s'intersectent est que leurs AABB respectives s'intersectent. Donc s'il n'y a pas de collision entre les deux AABB, on est certain que les polygones ne s'intersectent pas. Si par contre les AABB s'intersectent, alors il faut vérifier en détail les polygones. Chaque patch étant cubique, on considère les trois segments $[P_0P_1]$, $[P_1P_2]$ et $[P_2P_3]$ et on vérifie les collisions possibles entre les neufs paires de segments [50, p. 28-30]. A la fin de cette étape, chaque couple de polygones de contrôle présentant une collision est donné en entrée à la fonction flip, que l'on présente maintenant.

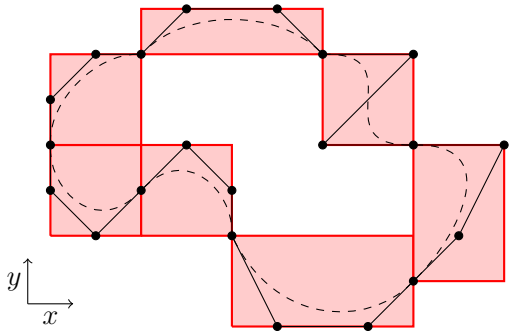


FIGURE 2.17 – AABB de plusieurs polygones de contrôles.

Comme on l'a précédemment mentionné, seuls deux cas de collisions entre polygones de contrôle sont considérés (voir Figure 2.15). L'algorithme de flip décrit ici est une procédure simple traitant ces deux cas et conduisant à une séparation d'une composante connexe ou à la réunion de deux composantes. On commence par décrire l'algorithme pour le cas d'une collision entre deux polygones de contrôle et ensuite le second cas métant en jeu trois polygones.

2.4.4.1 Premier cas : collision entre deux polygones de contrôle

On note $\mathbf{P} = \{P_0, P_1, P_2, P_3\}$ et $\mathbf{Q} = \{Q_0, Q_1, Q_2, Q_3\}$ les points des deux polygones de contrôle qui s'intersectent. L'algorithme de flip construit les deux nouveaux polygones suivants

$$\mathbf{P}' = \left\{ P_0, P_0 + \frac{1}{3}\overrightarrow{P_0Q_3}, P_0 + \frac{2}{3}\overrightarrow{P_0Q_3}, Q_3 \right\} \qquad \text{et} \qquad \mathbf{Q}' = \left\{ Q_0, Q_0 + \frac{1}{3}\overrightarrow{Q_0P_3}, Q_0 + \frac{2}{3}\overrightarrow{Q_0P_3}, P_3 \right\}.$$

La Figure 2.18 montre l'action du flip dans le cas d'une division d'une composante et la Figure 2.19 dans le cas d'une réunion de deux composantes.

Après avoir créé ces deux nouveaux polygones, le flip exécute une opération de réorganisation des patches, différente selon que les deux polygones originaux appartenaient à la même composante connexe ou à deux composantes différentes. En pratique, il y a un tableau de points de contrôle pour chaque composante et pour chaque composante tous

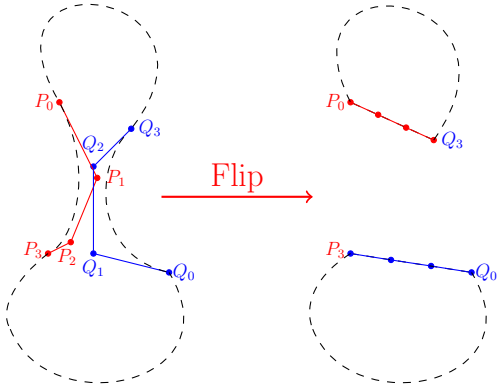


FIGURE 2.18 – Flip dans le cas de deux polygones - Division d'une composante.

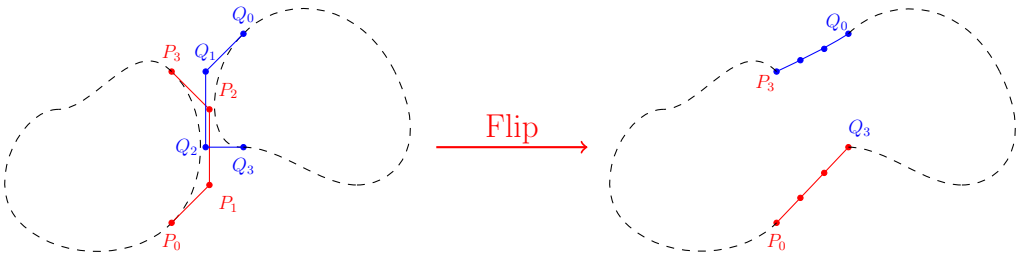


FIGURE 2.19 – Flip dans le cas de deux polygones - Réunion de deux composantes.

ses patchs ont un même numéro afin savoir à quelle composante ils appartiennent. Si le flip sépare une composante connexe dont le numéro est k en deux autres, le tableau numéro k de points de contrôle sera lui aussi scindé en deux. Une première partie du tableau conservera son numéro k et contiendra le nouveau patch $\mathbf{P}'$ et une seconde partie du tableau original formera un nouveau tableau numéroté $M+1$ contenant le patch $\mathbf{Q}'$ (où M correspond au nombre de composantes connexes de la forme avant le flip). Si par contre le flip réunit deux composantes connexes, alors une opération inverse est nécessaire, fusionnant les deux tableaux des points de contrôle.

2.4.4.2 Second cas : collision entre trois polygones de contrôle

L'action du flip dans ce cas est très similaire au cas précédent. On note $\mathbf{P} = \{P_0, P_1, P_2, P_3\}$ le polygone de contrôle intersectant les deux polygones $\mathbf{Q} = \{Q_0, Q_1, Q_2, Q_3\}$ et $\mathbf{R} = \{R_0, R_1, R_2, R_3\}$. L'algorithme de flip produit les deux polygones suivants

$$\mathbf{P}' = \left\{ P_0, P_0 + \frac{1}{3}\overrightarrow{P_0R_3}, P_0 + \frac{2}{3}\overrightarrow{P_0R_3}, R_3 \right\} \qquad \text{et} \qquad \mathbf{Q}' = \left\{ Q_0, Q_0 + \frac{1}{3}\overrightarrow{Q_0P_3}, Q_0 + \frac{2}{3}\overrightarrow{Q_0P_3}, P_3 \right\}.$$

La Figure 2.20 illustre le cas de la division et la Figure 2.21 le cas de la réunion.

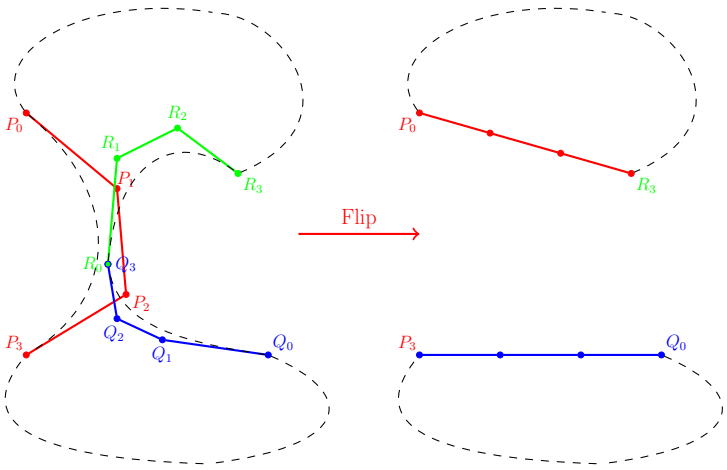


FIGURE 2.20 – Flip dans le cas de trois polygones - Division d'une composante.

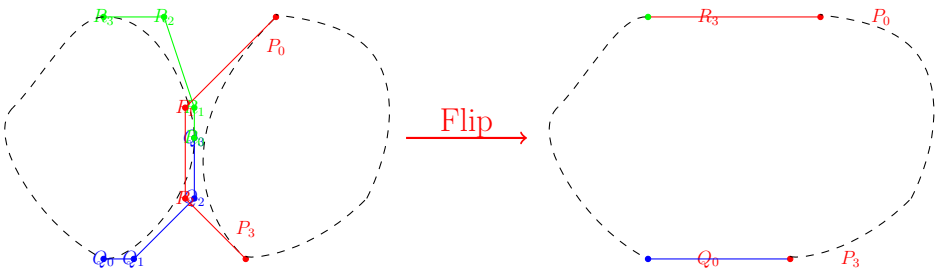


FIGURE 2.21 – Flip dans le cas de trois polygones - Fusion de deux composantes.

Chapitre 3 :

Applications

Dans ce chapitre on présente trois problèmes sur lesquels on a utilisé la représentation par courbes de Bézier par morceaux et le flip pour modifier la topologie des formes. La première application provient du domaine de l'électromagnétisme puisqu'il s'agit d'un problème de réalisation de filtre micro-ondes. Classiquement, on concevait ces composants électriques en changeant les matériaux constitutifs et les dimensions des structures pour obtenir un filtre vérifiant des spécifications préétablies. L'optimisation géométrique par les courbes de Bézier permet une plus grande souplesse et donc *a priori* d'obtenir des configurations plus performantes que celles obtenues par l'optimisation paramétrique. Le deuxième problème est celui de la détection d'objets immergés à la surface d'un fluide. La formulation en terme de problème d'optimisation de formes a été considérée à maintes reprises et la littérature propose plusieurs méthodes d'optimisation comme les lignes de niveau, le gradient topologique ou l'otpimisation géométrique classique. Ce fut l'occasion pour nous de tester la méthode de flip pour détecter deux obstacles partant d'un forme initiale connexe. On propose une troisième application à des problèmes de trajectoires optimales où les courbes de Bézier ne servent plus à représenter des formes par une courbe fermée, mais des trajectoires ou chemins donc des courbes ouvertes. Les résultats donnés proviennent de premiers exemples simples et l'on propose de généraliser la méthode à des problèmes de contrôle optimal et à la génération de chemins pour méthodes par homotopie.

On commence par présenter le problème de conception de filtres micro-ondes puis l'on présente le problème de la détection d'inclusions et l'on termine avec les problèmes de trajectoires optimales.

3.1 Application à la conception d'un filtre micro-ondes

Une première application de notre méthode d'optimisation de formes par contours libres concerne le domaine de l'électromagnétisme et plus particulièrement la conception de filtres micro-ondes passe-bande. Les filtres sont des éléments clés dans les systèmes d'émission-réception sans fils (stations de base, satellites, mobiles ...) car ils permettent de sélectionner le signal utile en rejetant les signaux interférents dans les bandes de fréquences voisines. Leur conception doit satisfaire des spécifications électriques, d'encombrements et de coût qui dépendent de l'application (pour une cible grand public le coût doit être faible). Les technologies ultra-performantes sont généralement encombrantes et entrainent un coût élevé. A l'inverse, on peut utiliser des technologies beaucoup plus intégrées et à coût réduit mais cela conduit généralement à une dégradation des performances (forte dissipation du signal, faible tenue en niveau de puissance ...).

L'objectif en conception/optimisation est donc de dimensionner le filtre pour répondre

aux spécifications électriques et les optimiser : maximiser la transmission dans la bande passante, ce qui revient à minimiser la réflexion et minimiser les pertes et atténuer la transmission hors bande. Pour optimiser, classiquement, on travaille sur les matériaux et sur les dimensions des structures, ce qui correspond à une optimisation paramétrique. Notre but est d'utiliser une méthode d'optimisation géométrique par l'intermédiaire des courbes de Bézier afin d'augmenter le degré de liberté et d'élargir l'espace des solutions atteignables et donc potentiellement d'obtenir des solutions plus performantes.

Ce travail est une collaboration entre l'équipe DMI de mathématiques et l'équipe MINACOM de physique du laboratoire XLIM. Plusieurs travaux de recherche ont été dirigés par l'équipe MINACOM [45, 43] où trois méthodes d'optimisation ont été testées, à savoir une première méthode d'optimisation par gradient topologique, une deuxième utilisant les lignes de niveaux et une méthode évolutionnaire par un algorithme génétique. Pour les deux premières, la direction de déformation est calculée par le logiciel EMXD, développé par Michel Aubourg, et qui est un gradient topologique. Le but de notre travail est de tester notre méthode de contours libres en utilisant ce gradient topologique pour tenter d'obtenir de meilleurs solutions.

Dans un premier temps on expose brièvement le contexte en présentant le fonctionnement d'un filtre et les spécifications que l'on cherche à respecter, ensuite nous posons le problème d'optimisation de formes puis nous présentons en détail la méthode de déformation des formes à partir du gradient renvoyé par le logiciel EMXD.

3.1.1 Fonctionnement d'un filtre passe-bande

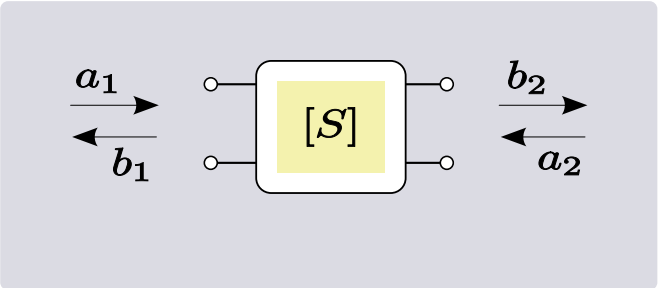


FIGURE 3.1 – Représentation schématique d'un filtre à deux ports.

Un filtre passe-bande est un composant électrique réalisant une fonction de filtrage des signaux entrants suivant leurs fréquences. On se place ici dans le cas d'un filtre à deux ports (voir Figure 3.1) où le port 1 correspond à l'entrée du filtre et le port 2 la sortie. On définit la bande passante du filtre par l'intervalle de fréquences $[\nu_0, \nu_1]$ correspondant aux signaux qui doivent être transmis. Les signaux dont la fréquence se situe en dehors de la bande passante sont filtrés en atténuant leurs amplitudes. Le comportement d'un

filtre est mesuré par ses paramètres-S [34]. En notant a_i le signal ou onde incidente au port i et b_i l'onde réfléchie au port i, on a les deux relations suivantes :

$$b_1 = s_{11}a_1 + s_{12}a_2$$
$$b_2 = s_{21}a_1 + s_{22}a_2$$

où $s_{11}, s_{12}, s_{21}, s_{22}$ sont les quatres paramètres-S du filtre. Chaque paramètre s_{ij} mesure le rapport entre l'onde réfléchie b_i et l'onde incidente a_j. Le paramètre s_{21} correspond à la fonction de transfert du filtre car il relie l'onde sortante b_2 et l'onde entrante a_1. Une courbe de gain, définie pour chaque paramètre-S, permet de visualiser le comportement du filtre suivant les fréquences des signaux entrants. Dans le cas d'un filtre passe-bande, on cherche une allure spécifique du paramètre s_{21}. La Figure [Y idéale] montre la courbe de gain du paramètre s_{21} pour un filtre idéal : le transfert des singaux est maximal dans la bande passante et minimal en dehors. Le problème de conception d'un filtre passe-bande consiste à chercher une allure de paramètre s_{21} approchant le gabarit idéal de la Figure [Y idéale]. Ainsi on utilise un critère du type moindes carrés de la forme

$$\mathcal{J} = \int_{\nu_{min}}^{\nu_{max}} \mid s_{21}(\nu) - s_{21}^0(\nu) \mid^2 d\nu$$

où $[\nu_{min}, \nu_{max}]$ est un intervalle de fréquences contenant la bande passante et $s_{21}^0(\nu)$ est un paramètre-S, dit de référence, à atteindre.

3.1.2 Position du problème

Le problème de conception de filtre est un problème inverse : on désire une sortie spécifique suivant les fréquences des ondes entrantes et la question est de trouver un filtre qui respecte cette spécification. Le type de filtre que l'on cherche est un guide d'onderectangulaire ayant deux accès correspondant aux deux ports du filtre. La partie basse est un subtrat diélectrique au dessus duquel se trouve deux parties métalliques permettant d'alimenter le guide au niveau de ses accès. La surface du substrat constitue le domaine du problème d'optimisation où l'on applique un dépôt de métal. C'est ce dernier qui correspond à la forme Ω determinant le comportement du filtre (voir Figure 3.2).

Pour définir la fonction objectif du type moindres carrés, on échantillonne l'intervalle de fréquence $[\nu_{min}, \nu_{max}]$ en N points $\nu_1, ..., \nu_N$. L'expression du critère est la suivante

$$\mathcal{J}(\Omega) = \sum_{i=1}^{N} \mid s_{21}(\Omega, \nu_i) - s_{21}^0(\nu_i) \mid^2 .$$

Le paramètre $s_{21}(\Omega, .)$ correspond au paramètre-S du filtre pour la forme courante Ω.

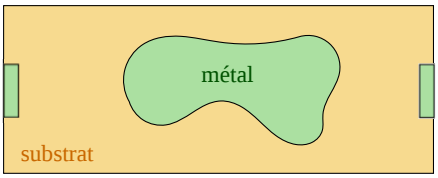


FIGURE 3.2 – Vue de dessus du filtre.

En utilisant le logiciel EMXD calculant le critère $\mathcal{J}(\Omega)$ et son gradient topologique g et en paramétrant la frontière $\partial\Omega$ par courbes de Bézier par morceaux, on a implémenté un algorithme d'optimisation de formes visant à obtenir une allure de paramètre s_{21} proche de celle du paramètre de référence s_{21}^0.

3.1.3 Présentation du code d'optimisation

Le code d'optimisation réalisé en Matlab fonctionne avec EMXD qui calcule la valeur de la fonction objectif $\mathcal{J}(\Omega)$ et le gradient topologique g. Ce logiciel discrétise le domaine en un maillage régulier rectangulaire. On note ici M le nombre de mailles dans le domaine. A chaque maille est associée un réel compris entre 0 et 1 donnant la proportion de métal dans la maille. Un valeur nulle correspond à une maille sans métal donc en dehors de la forme Ω. Dans le cas contraire, c'est-à-dire une valeur égale à 1, cela signifie que la maille est à l'intérieur de la forme. Les mailles traversées par la frontière $\partial\Omega$ ont des valeurs intérmédiaires. Une forme est donc décrite par un tableau de taille M et celui-ci consitue l'entrée d'EMXD. Ce dernier résoud les équations de Maxwell à l'intérieur du guide d'onde 3D pour obtenir le champ électromagnétique et calcule ensuite le paramètre s_{21} et le critère. Il calcule aussi le gradient topologique de la fonction objectif et le renvoie sous forme de tableau de réels de longueur M, où chaque valeur g_i a la signification suivante :

— si $g_i \geq 0$, alors ajouter du métal dans la maille i fera augmenter le critère et retirer du métal le fera diminuer,

— si $g_i \leq 0$, alors c'est l'inverse du cas précédent.

Pour résumer les entrées/sorties d'EMXD, on a un tableau ou vecteur de $[0,1]^M$ en entrée décrivant la forme courante Ω sur le maillage et la valeur du critère $\mathcal{J}$ avec le gradient topologique g comme vecteur de $\mathbb{R}^M$ en sortie.

Puisque notre méthode représente la frontière $\partial\Omega$ par une courbe de Bézier par morceaux, si l'on veut utiliser EMXD on doit passer de la représentation par une courbe à la représentation par un vecteur de $[0,1]^M$. Une courbe de Bézier par morceaux est stockée simplement par un tableau de points de contrôle que l'on note ici simplement P.

En notant $T \in [0,1]^M$ le vecteur donné en entrée à EMXD, une première tâche est de construire une fonction transformant le tableau P de points de contrôle en le tableau T de proportions de métal

$$T \longleftarrow BezierToProportions(P).$$

Ensuite une seconde étape s'occupe de la déformation par déplacement de la frontière en calculant de nouvelles coordonnées pour les points de contrôle P. La direction de déformation est calculée à partir de la donnée du gradient topologique g retourné par EMXD. On a donc la fonction suivante

$$P \longleftarrow MoveShape(P, g)$$

qui met à jour les points P.

Entre chaque appel de ces deux fonctions, EMXD est lancé et ces trois composantes constituent un tour de l'algorithme d'optimisation, résumé par l'algorithme suivant

Algorithme

Pour i allant de 1 à n :

1. $T \longleftarrow BezierToProportions(P)$
2. $\mathcal{J}, g \longleftarrow EMXD(T)$
3. $P \longleftarrow MoveShape(P, g)$

Dans la suite, on détaille les deux fonctions permettant le passage entre la représentation par courbes de Bézier et le tableau de proportions et la partie déformation à partir des valeurs du gradient topologique.

3.1.3.1 Fonction *BezierToProportions*

On rappelle que le maillage utilisé par EMXD est un maillage régulier rectangulaire. Pour calculer la proportion de métal dans chaque maille, on calcule un rapport d'aires. Si une maille est traversée par la frontière de la forme, une partie de la maille est à l'intérieur de la forme et la partie complémentaire est à l'extérieur. Dans ce cas la proportion de métal est calculée comme le rapport entre l'aire à l'intérieur de la forme et l'aire total de la maille, donnant ainsi une valeur appartenant à $[0,1]$. Partant alors de la donnée des points de contrôle de la courbe, il faut dans un premier temps calculer les intersections de la courbe avec le maillage. Ceci nous permet aussi d'identifer les mailles qui contiennent la frontière de la forme dont nous avons besoin pour la partie déformation. Pour chacune des mailles traversées par la frontière, la courbe est approchée par un segment joignant les deux points d'intersection ce qui permet de calculer le rapport d'aires mentionné

précédemment. Les autres mailles totalement à l'extérieur ou totalement à l'intérieur sont ensuite déterminées et on leur attribue les valeurs zéro ou un. Ceci conclut la première étape permettant de passer du tableau de points de contrôle au tableau de proportions.

3.1.3.2 Fonction *MoveShape*

Cette second fonction interprète les valeurs du gradient topologique g renvoyé par EMXD pour calculer une déformation de la frontière, c'est-à-dire un déplacement des points de contrôle. La méthode qu'on utilise consiste à se limiter aux mailles à la frontière : pour chacune de ces mailles, la valeur du gradient topologique indique s'il faut augmenter ou diminuer la quantité de métal pour diminuer le critère. Le but est donc de déplacer les points de contrôle de sorte à respecter au mieux le sens des variations pour ces mailles. Pour calculer les nouveaux points de contrôle, on construit un système linéaire avec autant d'équations qu'il y a de mailles à la frontière. Afin de décrire la signification de chaque équation, on introduit les notations suivantes. On considère une maille, numérotée i, à la frontière et l'on suppose qu'elle est traversée une seule fois par la courbe en deux points d'intersection M_1 et M_2. Cette hypothèse est réaliste car les mailles sont petites par rapport aux patches. Ces deux points étant sur la courbe, il existe deux valeurs $t_1, t_2 \in [0,1]$ telles que

$$M_1 = B(P, t_1) \text{ et } M_2 = B(P, t_2).$$

On note $t^* = \frac{t_1+t_2}{2}$ et $\vec{n}$ le vecteur normal extérieur au point $B(P,t^*)$. En notant g_i la valeur du gradient topologique pour la maille et $P^{'}$ les points de contrôle, inconnues du système, on construit l'équation suivante

$$B(P^{'}, t^*) = B(P, t^*) - \alpha g_i \vec{n}$$

avec $\alpha > 0$, par exemple 10^{-3}.

Cette équation signifie que l'on souhaite déplacer la courbe de sorte à ce qu'elle passe par le point $B(P,t^*) - \alpha g_i \vec{n}$. Ainsi si $g_i > 0$, l'aire métallisée dans la maille va diminuer et si $g_i < 0$ cette aire va augmenter. Le nombre de mailles étant supérieur à celui du nombre de points de contrôle, on résoud le système au sens des moindres carrés.

3.1.4 Résultats

Nous avons testé l'algorithme précédent sur un exemple où le paramètre de référence s_{21}^0 à atteindre a été calculé pour une forme particulière. Le but était de commencer l'algorithme avec une forme initiale légèrement modifiée et de voir si l'on retrouvait la forme utilisée pour calculer le paramètre s_{21}^0.

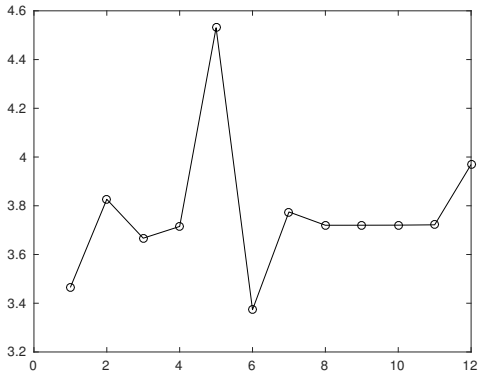


FIGURE 3.3 – Evolution du critère.

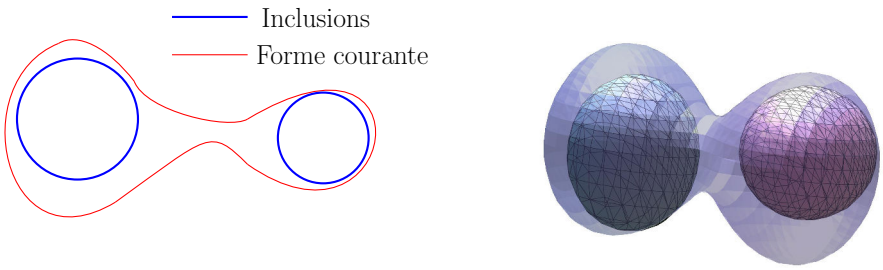
Le domaine d'optimisation consiste en un rectangle discrétisé en 3200 mailles carrées (80 mailles par 40 mailles). Le paramètre s_{21}^0 de référence a été calculé pour un dépôt métallique carré de 10 mailles par 10 centré dans le domaine. La forme initiale est une perturbation de ce carré : c'est un rectangle de taille 12 par 8 centré dans le domaine. L'évolution de la fonction objectif est donné en Figure 3.3. Le critère a plutôt tendance à augmenter alors qu'il devrait diminuer. En effet, la forme initiale étant légérement différente du carré de côté 10, la méthode de descente suivant le gradient devrait converger vers le carré. Le problème vient du gradient topologique calculé par EMXD qui n'est pas adapté pour notre méthode géométrique. Un gradient géométrique est mieux adapté car il donne une variation en chaque point de la frontière ce qui est idéal pour notre méthode puisqu'on aurait qu'à interpoler la déformation de la frontière. La déduction d'une variation géométrique à partir du gradient topologique échoue : les nouveaux points de contrôle, calculés en résolvant le système linéaire, définissent une forme qui ne respecte pas suffisamment les variations de métal données par le gradient topologique.

Un travail est en cours avec la société *CST - Computer Simulation Technology* afin d'obtenir la dérivée des paramètres-S en fonction d'un déplacement de la frontière. Ainsi on pourrait ensuite en déduire, par composition, le gradient de forme du problème et mettre en place un algorithme d'optimisation géométrique en utilisant les courbes de Bézier ainsi que le flip.

3.2 Application à la détection d'inclusions

Comme seconde application, on s'est intéressé au problème de la détection d'inclusions en deux dimensions. Cette section reprend les résultats publiés dans notre article *Flip procedure in geometric approximation of multiple-component shapes - Application to multiple-inclusion detection* du journal *SMAI - Journal of computational mathematics*. Ce problème consiste à localiser des obstacles présents dans un domaine Ω et à déterminer leur forme à partir de mesures prises à la frontière $\partial\Omega$. Il y a donc un aspect topologique

car il faut trouver le nombre d'obstacles et un aspect géométrique puisqu'on souhaite connaître leurs formes. Comme pour la conception de filtre, il s'agît d'un problème inverse que l'on peut résoudre numériquement grâce à l'optimisation de formes, où la forme optimale consiste en ces différentes inclusions présentes dans le domaine. Ce problème a permi de tester non seulement la paramétrisation par courbes de Bézier dans un problème d'optimisation de formes mais aussi l'algorithme de flip permettant de modifier la topologie des formes tout en utilisant une méthode géométrique de déformation. Dans le cas où il y a plusieurs obstacles et partant d'une forme initiale connexe, celle-ci va progressivement se déformer jusqu'à englober les inclusions comme l'illustre la Figure 3.4 tout en conservant sa topologie initiale. Cette figure montre que dans une telle situation l'algorithme de flip serait déclenché et diviserait la forme courante en deux composantes connexes.



(a) Cas en dimension deux (b) Cas en dimension trois

FIGURE 3.4 – Une forme connexe détectant deux obstacles.

Plusieurs approches utilisant l'optimisation de formes existent pour résoudre le problème de la détection d'inclusions. On classe ces méthodes dans deux grandes catégories qui sont les méthodes topologiques utilisant le gradient topologique et les méthodes géométriques avec le gradient de forme. L'approche par le gradient topologique a été proposée par Schumacher [58] et Sokolowski *et al.* [65]. On mentionne également les travaux de [31] concernant des généralités sur l'expansion asymptotique topologique dans le cas de l'élasticité. On fait remarquer que cette méthode repose sur l'expansion asymptotique et donc est adaptée pour le cas d'inclusions de petites tailles. De plus si l'optimisation topologique permet de déterminer le nombre d'obstacles, elle n'est cependant pas bien adaptée lorsqu'il s'agît de donner une bonne approximation de leur géométrie (voir par exemple [17] ainsi que les références qui s'y trouvent). Concernant l'approche géométrique, on distingue deux méthodes, qui ont en commun l'utilisation du gradient géométrique donnant la direction de déplacement, mais qui se différencient l'une de l'autre par la façon dont elles représentent et déforment la frontière de la forme. La première est celle utilisant la méthode des lignes de niveaux (voir par exemple l'article de revue [16] de Burger *et al.* et les références qui y figurent ou [61]). La méthode des lignes de niveaux permet de changer la topologie de la forme donc elle n'a pas besoin de

connaître le nombre d'inclusions à l'avance mais elle offre une représentation implicite des obstacles. La seconde est basée sur le déplacement de frontière (au sens de l'optimisation géométrique classique en déplaçant les sommets du maillage) qui est utilisé notamment dans les travaux de Afraites *et al.* [1]. On rappelle que cette méthode géométrique par déformation du maillage n'autorise pas de changement topologique et donc le nombre d'obstacles doit nécessairement être connu à l'avance. D'autres travaux plus récents combinent les différentes approches mentionnées précédemment. Par exemple dans les travaux de Allaire *et al.* [3, 4] et Burger *et al.* [15] (voir aussi la thèse [54, Section 5]) les auteurs utilisent à la fois le gradient topologique et la méthode des lignes de niveaux, les travaux de Pantz *et al.* [53] utilisant la variation de frontière, gradient topologique et la méthode d'homogénéisation, et les travaux de Caubet *et al.* [18] et Christiansen *et al.* [19] couplant l'approche topologique avec la variation de frontière.

Notre méthode de paramétrisation par les courbes de Bézier et l'algorithme de flip peut être vue comme une alternative aux méthodes combinant les lignes de niveaux et le gradient topologique. Cette nouvelle méthode semble bien adaptée pour étudier le problème de détection d'inclusion, en particulier lorsque le nombre d'obstacles est inconnu. L'avantage offert par les courbes de Bézier réside dans la facilité d'implantation de la représentation de la frontière (à savoir un tableau de points de contrôle) et dans le fait de posséder une représentation explicite des formes. Dans la suite de cette section, on présente et pose dans un premier temps le problème de la détection d'inclusions et dans un second temps on donne des exemples de simulations numériques.

3.2.1 Position du problème

Le problème consiste à trouver numériquement un ou plusieurs obstacles notés ω_{ex} présent à l'intérieur d'un domaine borné $\Omega \in \mathbb{R}^2$ à partir de mesures prises à sa frontière $\partial\Omega$. Afin de résoudre numériquement ce problème inverse, on utilise l'optimisation de formes en minimisant une fonction objectif du type moindres carrés. La méthode utilise le gradient de forme provenant de l'optimisation géométrique présentée dans le premier chapitre.

On note L^p, $\mathrm{W}^{m,p}$ et H^s les espaces de Lebesgue et Sobolev usuels. On utilise la notation en gras pour désigner les espaces et fonctions vectorielles comme $\mathbf{W}^{m,p}$. On pose Ω ouvert non-vide borné de $\mathbb{R}^2$ dont la frontière est $C^{2,1}$ et $g \in \mathrm{H}^{5/2}(\partial\Omega)$ telle que $g \neq 0$. On note $\mathbf{n}$ le vecteur normal extérieur unitaire à $\partial\Omega$, et pour une fonction u suffisamment régulière, on note $\partial_{\mathbf{n}} u$ la dérivée normale de u. On fixe $d_0 > 0$ (petit) et on note $\mathcal{O}_{d_0}$ l'ensemble des ouverts ω strictement contenu dans Ω dont la frontière est $C^{2,1}$ tels que la distance $\mathrm{d}(x, \partial\Omega)$ de x au compact $\partial\Omega$ est strictement supérieure à d_0 pour tout $x \in \omega$ et tels que $\Omega \backslash \overline{\omega}$ est connexe. Enfin on note Ω_{d_0} ouvert à frontière C^∞ vérifiant

$$\{x \in \Omega\,;\ \mathrm{d}(x, \partial\Omega) > d_0/2\} \subset \Omega_{d_0} \subset \{x \in \Omega\,;\ \mathrm{d}(x, \partial\Omega) > d_0/3\}\,.$$

On considère le problème inverse suivant. Supposons qu'un obstacle $\omega_{\mathrm{ex}} \in \mathcal{O}_{d_0}$ est situé à l'intérieur de Ω. On considère l'équation de Laplace suivante dans $\Omega\backslash\overline{\omega_{\mathrm{ex}}}$ avec condition aux limites de Dirichlet homogène sur $\partial\omega_{\mathrm{ex}}$ et condition aux limites de Dirichlet non-homogène sur $\partial\Omega$. On note $u_{\mathrm{ex}} \in \mathrm{H}^1(\Omega\backslash\overline{\omega_{\mathrm{ex}}})$ l'unique solution du problème

$$\left\{\begin{array}{rcll} -\Delta u_{\mathrm{ex}} & = & 0 & \text{in } \Omega\backslash\overline{\omega_{\mathrm{ex}}}, \\ u_{\mathrm{ex}} & = & g & \text{on } \partial\Omega, \\ u_{\mathrm{ex}} & = & 0 & \text{on } \partial\omega_{\mathrm{ex}}. \end{array}\right. \tag{3.1}$$

Puisque $g \in \mathrm{H}^{5/2}(\partial\Omega)$ et $\Omega\backslash\overline{\omega_{\mathrm{ex}}}$ possède une frontière $C^{2,1}$, u_{ex} appartient à $\mathrm{H}^3(\Omega\backslash\overline{\omega_{\mathrm{ex}}})$. Le but est de retrouver numériquement la forme ω_{ex} à partir de mesures sur la frontière $\partial\Omega$. On travaille dans le cas où la mesure $f_b := \partial_{\mathbf{n}} u_{\mathrm{ex}} \in \mathrm{H}^{3/2}(\partial\Omega)$ sur $\partial\Omega$ est exacte. Ainsi, étant donné une paire de Cauchy non-nulle $(g, f_b) \in \mathrm{H}^{5/2}(\partial\Omega) \times \mathrm{H}^{3/2}(\partial\Omega)$, on cherche à résoudre le problème inverse suivant

trouver $\omega \in \mathcal{O}_{d_0}$ *et* $u \in \mathrm{H}^1(\Omega\backslash\overline{\omega}) \cap C^0(\Omega\backslash\overline{\omega})$ *satisfaisant le système*

$$\left\{\begin{array}{rcll} -\Delta u & = & 0 & \text{in } \Omega\backslash\overline{\omega}, \\ u & = & g & \text{on } \partial\Omega, \\ \partial_{\mathbf{n}} u & = & f_b & \text{on } \partial\Omega, \\ u & = & 0 & \text{on } \partial\omega. \end{array}\right. \tag{3.2}$$

L'existence d'une solution est triviale puisque on suppose la mesure f_b exacte. Par le théorème de Holmgren (voir, par exemple, [41]), on obtient un résultat d'identifiabilité pour ce problème inverse qui certifie l'unicité de la solution. Cette question fondamentale de l'unicité d'une solution d'un problème surdimensionné (3.2) a été vivement étudié (voir, par exemple, [13, Theorem 1.1], [21, Theorem 5.1] or also [22, Proposition 4.4 p. 87]). On rappelle ce résultat d'identifiabilité.

Théorème 1 *Le domaine* ω *et la fonction* u *satisfaisant* (3.2) *sont uniquement déterminés par la donnée de Cauchy* $(g, f_b) \neq (0, 0)$.

Remarque 1 *Les résultats présentés peuvent être adaptés dans le cas où la grandeur* f_b *n'est mesurée que sur un sous-ensemble non-vide* O *de* $\partial\Omega$ *(voir, par exemple, [17]).*

Afin de résoudre le problème inverse (3.2) on considère le problème d'optimisation de forme

$$\omega^* \in \operatorname*{argmin}_{\omega \in \mathcal{O}_{d_0}} \mathcal{J}(\omega), \tag{3.3}$$

où $\mathcal{J}$ est la fonction objectif de type moindres carrés définie par

$$\mathcal{J}(\omega) := \int_{\partial\Omega} |\partial_{\mathbf{n}} u_\omega - f_b|^2 \,,$$

avec $u_\omega \in \mathrm{H}^3(\Omega\backslash\overline{\omega})$ étant l'unique solution du problème

$$\left\{ \begin{array}{rcll} -\Delta u_\omega &=& 0 & \text{in } \Omega\backslash\overline{\omega}, \\ u_\omega &=& g & \text{on } \partial\Omega, \\ u_\omega &=& 0 & \text{on } \partial\omega. \end{array} \right. \tag{3.4}$$

Le résultat d'identifiabilité assure en effet que $\mathcal{J}(\omega) = 0$ si et seulement si $\omega = \omega_{\mathrm{ex}}$. On passe maintenant au calcul du gradient du critère $\mathcal{J}$ afin d'utiliser une méthode numérique de résolution de type descente par gradient.

Afin de définir la dérivée de forme, on utilise la méthode d'Hadamard. On introduit d'abord l'espace des déformations admissibles

$$\boldsymbol{U} := \{\boldsymbol{V} \in \mathbf{W}^{3,\infty};\ \mathrm{Supp}\ \boldsymbol{V} \subset \overline{\Omega_{d_0}}\}. \tag{3.5}$$

Le gradient de forme de $\mathcal{J}$ est défini par

$$\mathrm{D}\mathcal{J}(\omega)\cdot\boldsymbol{V} := \lim_{t\to 0} \frac{\mathcal{J}\big((\mathbf{I}+t\boldsymbol{V})(\omega)\big) - \mathcal{J}(\omega)}{t},$$

pour tout $\omega \in \mathcal{O}_{d_0}$ et tout $\boldsymbol{V} \in \boldsymbol{U}$.

Proposition 1 *La fonctionnelle de type moindres carrés $\mathcal{J}$ est différentiable en $\omega \in \mathcal{O}_{d_0}$ dans la direction $\boldsymbol{V} \in \boldsymbol{U}$ et*

$$\mathrm{D}\mathcal{J}(\omega)\cdot\boldsymbol{V} = -\int_{\partial\omega} \partial_{\mathbf{n}} u_\omega\, \partial_{\mathbf{n}} w_\omega\, (\boldsymbol{V}\cdot\mathbf{n})\,, \tag{3.6}$$

où $w_\omega \in \mathrm{H}^1(\Omega\backslash\overline{\omega})$ est l'unique solution du problème adjoint suivant

$$\left\{ \begin{array}{rcll} -\Delta w_\omega &=& 0 & \text{in } \Omega\backslash\overline{\omega}, \\ w_\omega &=& 2\,(\partial_{\mathbf{n}} u_\omega - f_b) & \text{on } \partial\Omega, \\ w_\omega &=& 0 & \text{on } \partial\omega. \end{array} \right. \tag{3.7}$$

Grâce à l'expression explicite du gradient de forme donné ci-dessus, on est maintenant en mesure d'effectuer des simulations numériques basées sur la méthode classique de descente suivant le gradient et l'on inclut aussi l'algorithme de flip présenté au chapitre 2 afin de détecter plusieurs obstacles.

3.2.2 Simulations numériques

Afin de passer aux simulations numériques, rappelons que l'on peut rencontrer plusieurs difficultés pour résoudre numériquement le problème (3.3), comme expliqué dans [1, Théorème 1] (voir aussi [10, Proposition 2.4]). En effet, la sensibilité du gradient n'est pas uniforme par rapport aux directions de déformations. Cependant on utilise une méthode de paramétrisation, par les courbes de Bézier par morceaux, qui est une méthode de régularisation (comme celle utilisée dans [1] avec des séries de Fourier tronquées) ce qui permet de compenser le caractère mal posé du problème inverse et de le résoudre numériquement.

Notons que l'on utilise des courbes de Bézier par morceaux qui ne satisfont pas la condition de régularité $\mathcal{C}^{2,1}$. Cette hypothèse de régularité est suffisante pour prouver les résultats théoriques précédents. Dans cette section dédiée aux simulations numériques, on fait le choix d'oublier ce problème de régularité puisque l'on observe de relativement bonnes reconstructions des obstacles.

Les simulations numériques présentées ici ne traitent que le cas de la dimension deux et ont été réalisées grâce au logiciel de résolution d'équations différentielles par la méthode des éléments finis *FreeFem++* [37]. La frontière extérieure $\partial\Omega$ est le cercle de rayon 10 centré à l'origine et l'on pose comme condition de Dirichlet $g = 100$. Afin d'avoir une mesure f_b, on fixe une forme ω_{ex}, puis l'on résoud le problème (3.1) par la méthode des éléments finis (avec des éléments P2) et l'on extrait la mesure f_b en calculant $\partial_{\mathbf{n}} u_{\text{ex}}$ sur $\partial\Omega$.

Ensuite on utilise une discrétisation en éléments P1 pour résoudre les problèmes (3.4) et (3.7) avec 50 points de discrétisation pour à la fois la frontière extérieure et chaque patch cubique de Bézier décrivant la forme ω. Afin de résoudre numériquement le problème d'optimisation (3.3), on utilise l'algorithme suivant de descente selon le gradient où l'on ajoute la procédure de flip à l'étape (2).

Algorithme $\mathcal{A}$

1. Fixer $k = 0$, choisir une forme initiale ω_0, un nombre d'itérations maximum $M \in \mathbb{N}^*$ et $\lambda \geq 1$ un coefficient de tolérance pour la procédure de flip (voir l'étape (2(b)ii), typiquement λ proche de 1).
2. Parcourir la paramétrisation $\partial\omega_k$ et chercher des intersections entre polygones de contrôle :
   (a) s'il n'y a aucune intersection, aller à l'étape (3) ;
   (b) sinon :

i. appliquer la procédure de flip pour obtenir une forme à plusieurs composantes $\omega_k^1 \cup \omega_k^2$;

ii. calculer $J(\omega_k^1 \cup \omega_k^2)$ et $J(\omega_k)$, et poser $J(\omega_k) = +\infty$ si ω_k possède une auto-intersection :

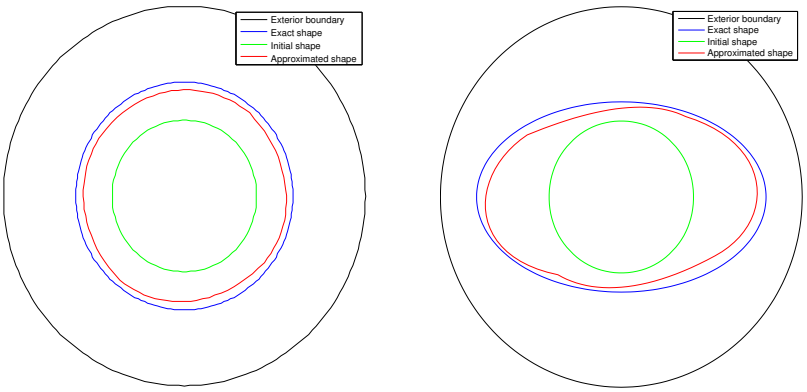
A. si $J(\omega_k^1 \cup \omega_k^2) < \lambda J(\omega_k)$, alors $\omega_k \leftarrow \omega_k^1 \cup \omega_k^2$;

B. sinon aller à l'étape (3).

3. Résoudre les problèmes (3.4) et (3.7) avec $\omega = \omega_k$.
4. Calculer le gradient de forme $\mathrm{D}J(\omega_k)$ avec la formule (3.6).
5. Déplacer les points de contrôle de la forme, c'est-à-dire, remplacer $\omega_{k+1} \leftarrow \omega_k - \alpha_k \mathrm{D}J(\omega_k)$, où α_k est un petit coefficient de déplacement.
6. $k \leftarrow k+1$ et retourner à l'étape (2) si $k < M$.

3.2.2.1 Premières simulations : détection de formes lisses et convexes

On teste d'abord l'algorithme précédent sur le problème de détection d'objets lisses et convexes. Plus précisément, on commence par la détection du cercle centre à l'origine et de rayon 6 et l'ellipse $\{(8\cos\theta,\ 5\sin\theta), \theta \in [0, 2\pi]\}$ en utilisant quatres patches de Bézier cubiques. Les résultats de ces simulations numériques sont donnés par la Figure 3.5.



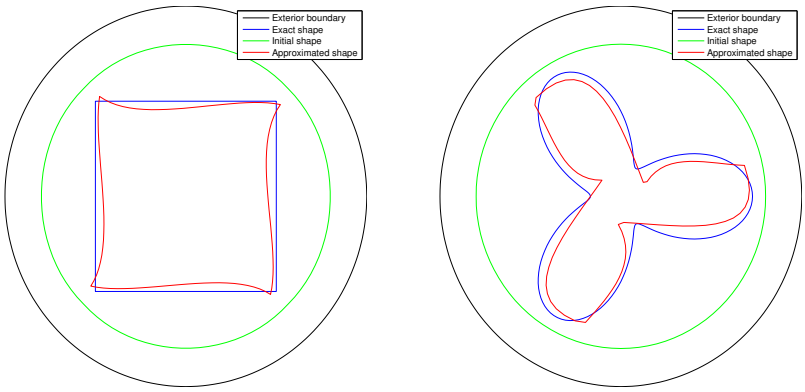
(a) Détection d'un cercle (b) Détection d'une ellipse

FIGURE 3.5 – Détection d'obstacles lisses et convexes.

3.2.2.2 Détection de formes non-lisses et non-convexes

On utilise maintenant l'algorithme d'optimisation sur le problème de détection de formes non-lisses et non-convexes (voir Figure 3.6). On considère un obstacle carré de côté de longueur 10 centré à l'origine et on utilise toujours quatre patches cubiques. Comme

on peut le voir sur la Figure 3.6(a), chaque patch de Bézier détecte un côté du carré. La Figure 3.7 montre la décroissance de la fonction objectif au cours de la simulation. Dans une deuxième simulation, on souhaite détecter la forme non-convexe paramétrée par $\{(2.8(1.6+\cos(3\theta))\cos(\theta), 2.8(1.6+\cos(3\theta))\cos(\theta)), \theta \in [0, 2\pi]\}$ et l'on augmente le nombre de patches à six.



(a) Détection d'un carré

(b) Détection d'une forme non-convexe.

FIGURE 3.6 – Détection d'un obstacle non lisse et d'un obstacle non-convexe.

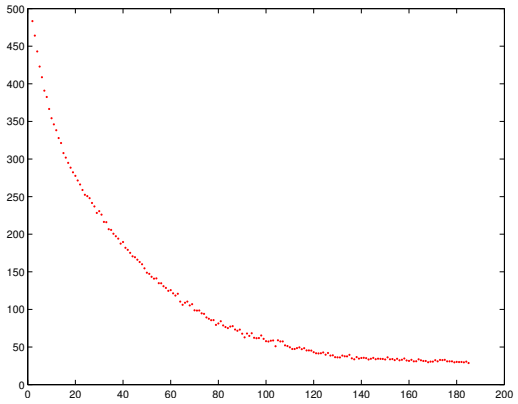
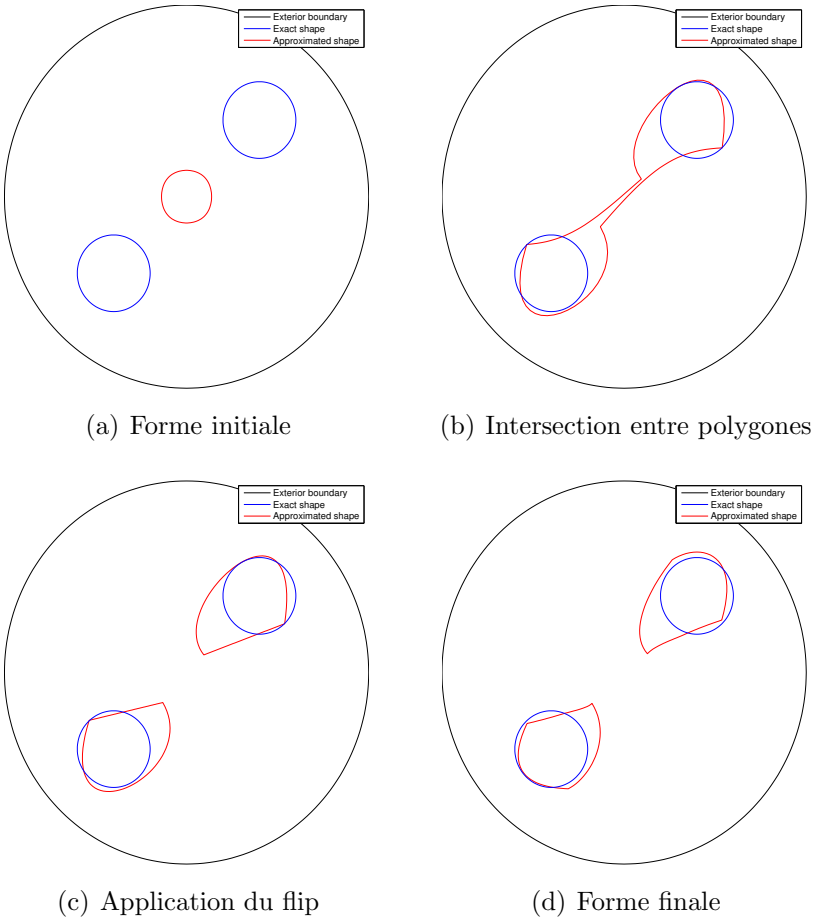


FIGURE 3.7 – Evolution de la fonction objectif pour la détection du carré.

3.2.2.3 Détection de deux obstacles partant d'une forme à une seule composante connexe

Ici on utilise la procédure de flip afin de détecter deux obstacles alors que la forme initiale utilisée dans l'algorithme n'est composée que d'une seule composante connexe. Les deux obstacles sont deux cercles de rayon 2 centrés aux points $(-4,-4)$ et $(4,4)$. On présente quatres étapes dans l'algorithme avec la Figure 3.8. La forme initiale consiste en une seule composante utilisant quatre patches de Bézier et est située au centre du domaine

(Figure 3.8(a)). La forme s'élargit et entoure les deux cercles jusqu'à ce que deux polygones de contrôle s'intersectent l'un l'autre (Figure 3.8(b)). Le flip est ensuite appliqué, divisant la forme en deux composantes connexes (Figure 3.8(c)). En fin d'algorithme, on obtient une approximation de la forme des deux obstacles (Figure 3.8(d)).



(a) Forme initiale (b) Intersection entre polygones

(c) Application du flip (d) Forme finale

FIGURE 3.8 – Détection de deux obstacles partant d'une forme initiale connexe.

La Figure 3.9 montre l'évolution du critère pendant la simulation. On remarque un changement de comportement après l'itération 133 qui correspond au moment du flip. Plus précisément, l'algorithme trouve un minimum local à l'itération 13, qui est un minimum local à une composante. Après quelques oscillations autour de ce minimum local, le flip intervient et la fonction décroit puis se stabilise autour d'un minimum local à deux composantes connexes.

3.2.2.4 Valeur de la fonction objectif après un flip

Dans l'Algorithme $\mathcal{A}$, l'étape (2(b)ii) assure que, lorsque un flip est exécuté, la fonction objectif n'augmente pas de manière significative. Si $J(\omega_k^1 \cup \omega_k^2) \geq \lambda J(\omega_k)$ (par exemple $\lambda = 1.1$), alors on considère que la division en deux composantes n'est pas un bon

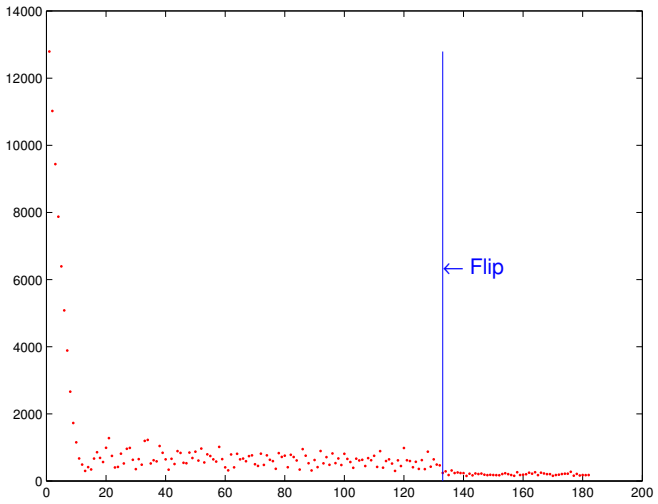
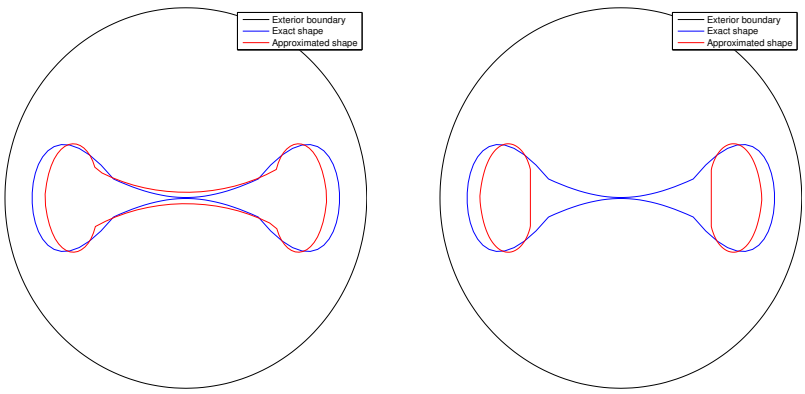


FIGURE 3.9 – Evolution de la fonction objectif pour la détection de deux obstacles.

choix et l'on annule le flip. Cette situation se produit par exemple lorsque l'obstacle à détecter possède une composante où deux parties de sa frontière sont très proches l'une de l'autre. Dans un tel cas, deux polygones de contrôle vont probablement s'intersecter et un flip sera lancé, alors qu'il s'agit d'un domaine connexe. On donne un exemple avec la Figure 3.10. L'obstacle est une unique composante connexe avec une partie très fine et la forme courante ω_k de l'algorithme possède deux polygones de contrôle qui s'intersectent l'un l'autre. Le critère a pour valeur $J(\omega_k) = 3211$ avant l'appel de la fonction flip et augmente jusqu'à $J(\omega_k^1 \cup \omega_k^2) = 3579$ après le flip. Puisque le rapport est supérieur à λ, le flip est annulé et l'algorithme continue avec une forme courante à une seule composante.



(a) Avant le flip, $J(\omega_k) = 3211$ (b) Après le flip, $J(\omega_k^1 \cup \omega_k^2) = 3579$

FIGURE 3.10 – Le critère augmente de façon significative lorsqu'un flip n'est pas nécessaire.

3.2.2.5 Détection d'un obstacle partant d'une forme à deux composantes

Comme on l'a vu au chapitre précédent, le flip peut aussi être utilisé pour réunir deux composantes en une seule (voir Figure 3.11). On présente ici un exemple où l'on cherche à détecter un obstacle formé d'une seule composante, paramétrée par $\{(4\cos\theta, 6 + 2.5\sin\theta), \theta \in [0, 2\pi]\}$, en partant d'une forme initiale à deux composantes. On présente quatres états de l'algorithme en Figure 3.12. A la fin, l'algorithme donne une approximation de l'obstacle par une seule forme connexe.

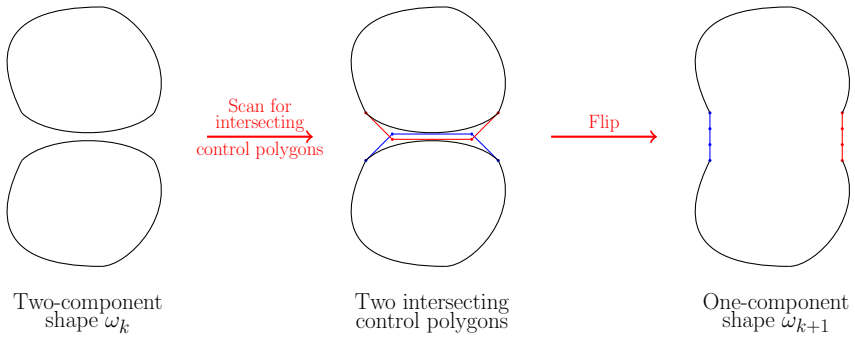
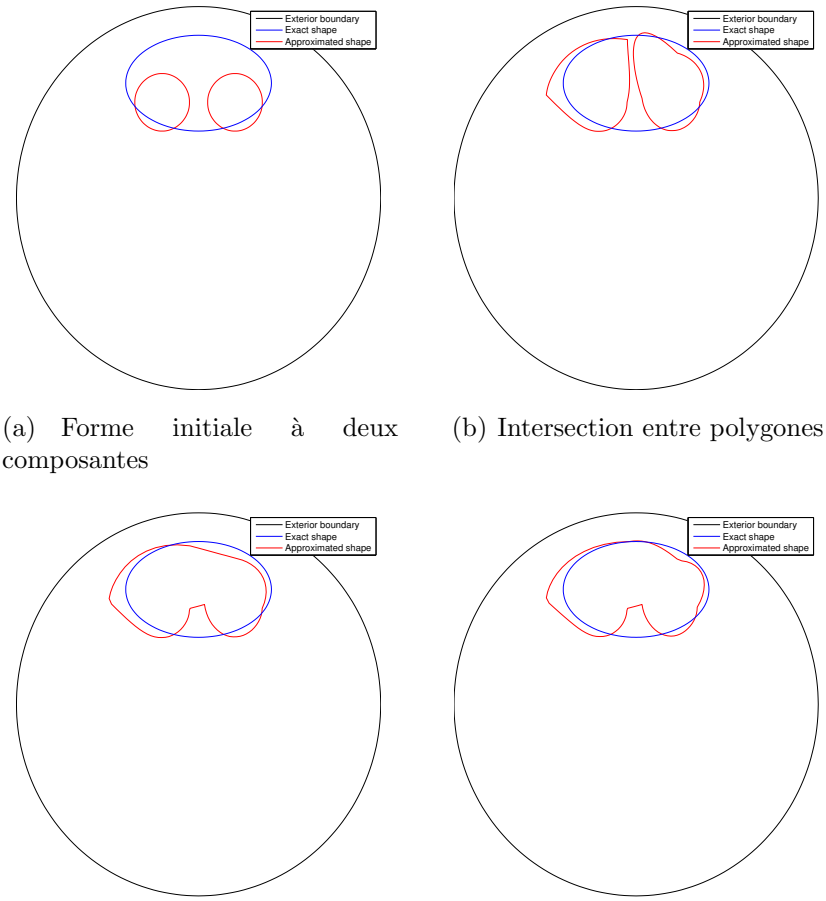


FIGURE 3.11 – Le flip peut aussi fusionner deux formes connexes en une.

Dans cette section concernant la détection d'objets on a pu appliquer notre méthode de représentation des formes par contours libres ainsi que l'algorithme de flip permettant de modifier la topologie pendant l'optimisation. Le choix d'une paramétrisation polynomiale par morceaux que sont les courbes de Bézier permet de détecter des obstacles non-lisses et non-convexes et la procédure de flip permet de trouver deux obstacles. On s'est restreint à un problème en deux dimensions mais la généralisation en dimension trois serait intéressante bien que plus délicate à mettre en place. On pourrait aussi essayer d'augmenter le nombre d'obstacles au lieu de se limiter à deux, mais les résultats d'un exemple à trois objets ne sont pour l'instant pas concluant. De plus l'implémentation décrite ici n'a qu'un but expérimental et une version plus complète et optimisée est envisageable.

3.3 Application aux problèmes de trajectoires optimales

Dans cette troisième section, on présente deux problèmes de trajectoires optimales dans lesquels on a utilisé les courbes de Bézier par morceaux pour approcher les solutions. Les travaux de cette section ont fait l'objet d'une communication lors des *Journées annuelles du GdR MOA 2014*. Dans les deux applications précédentes, on a utilisé les courbes de Bézier fermées pour représenter une forme par sa frontière. Dans le cas des trajectoires



(a) Forme initiale à deux composantes

(b) Intersection entre polygones

(c) Application du flip

(d) Forme finale

FIGURE 3.12 – Détection d'un obstacle partant d'une forme initiale à deux composantes.

optimales, on formule le problème en un problème d'optimisation (de formes) où les formes sont des trajectoires donc des courbes ouvertes. Etant donnés deux points A, B d'un espace vectoriel, un problème de trajectoire optimal consiste à relier A et B par un chemin minimisant une fonction coût. Dans le chapitre précédent, on parlait d'espace des formes pour un problème d'optimisation de formes, ici on utilise le terme d'espace des chemins. On considère l'espace des chemins noté $\mathcal{C}^{A\to B}$ reliant deux points A et B d'un espace vectoriel E

$$\mathcal{C}^{A\to B} = \{\ m\ :\ [0,1]\ \to\ E \text{ immersion injective propre vérifiant}$$
$$m(0) = A \text{ et } m(1) = B\}.$$

Les reparamétrisation sont les bijections $r\ :\ [0,1]\ \to\ [0,1]$ telles que $r^{'}(t) > 0$ pour tout $t \in [0,1]$ et l'on note $Diff([0,1])^{+}$ l'ensemble des reparamétrisations. De la même façon, on utilise l'espace des courbes de Bézier suivant

$$\mathcal{B}_{N,d}^{A\to B} = \{\gamma \in \mathcal{B}_{N,d} \text{ telle que } \gamma(0) = A \text{ et } \gamma(1) = B\}$$

pour approcher les éléments de la variété $\mathcal{C}^{A\to B}\Big/Diff([0,1])^+$.

On s'est intéressé à deux problèmes de trajectoires optimales. Dans le premier problème, on se donne un champ de vecteurs v et deux points A et B de $\mathbb{R}^2$ et l'on souhaite utiliser au mieux le champ v pour se déplacer de A à B. Le second problème est la génération de chemins pour des méthodes par homotopie pour la résolution de systèmes (calcul de racines, calcul de valeurs propres, valeurs singulières ...). Les méthodes par homotopie utilise l'homotopie linéaire, c'est-à-dire qu'elles considèrent le segment reliant un problème facile et le problème que l'on cherche à résoudre. Sur ce segment il risque d'y avoir un problème mal conditionné et l'idée est alors d'utiliser un chemin au lieu du segment de sorte à rester éloigner des problèmes mal conditionnés. On présente maintenant les premiers résultats obtenus dans chacun de ces deux problèmes.

3.3.1 Calcul de chemins dans l'espace soumis à un champ de vecteur

On se place ici dans le plan $E = \mathbb{R}^2$. On considère un champ de vecteur v de $\mathbb{R}^2$ dans $\mathbb{R}^2$ et deux points $A, B \in \mathbb{R}^2$. On souhaite relier A et B par un chemin tel que l'énergie $\mathcal{J}$ à fournir est minimale. En d'autres termes, on veut tirer avantage du champ v afin de fournir le moins d'effort possible. Dans le cas où A et B appartiennent à la même orbite, l'énergie à fournir est nulle et la trajectoire optimale est la portion d'orbite joignant A à B. Dans le cas contraire, si A et B ne sont pas sur la même orbite, il faut ajouter une force allant contre le champ pour atteindre B et donc dépenser de l'énergie. On a choisi la fonction énergie suivante

$$\begin{array}{rccl} \mathcal{J} & : & \mathcal{C}^{A\to B} & \to & \mathbb{R} \\ & & m & \mapsto & \displaystyle\int_0^1 \left\| m'(t) \times v(m(t)) \right\|_2^2 . \end{array}$$

En cherchant à minimiser $\mathcal{J}(m)$, on souhaite trouver une trajectoire $m(t)$ dont le vecteur vitesse $m'(t)$ est le plus possible colinéaire à $v(m(t))$.

3.3.1.1 Résultats des tests avec AMPL

Pour approcher les trajectoires par des courbes de Bézier par morceaux, on a réalisé des tests avec AMPL et le solveur SPDOPT. Les variables sont donc les coordonnées des points de contrôle des courbes de Bézier $\gamma \in \mathcal{B}_{N,d}^{A\to B}$. On a remplacé le critère $\mathcal{J}(\gamma)$ par la somme discrète suivante

$$\sum_{i=0}^{n} \left\| \gamma'(\frac{i}{n}) \times v(\gamma(\frac{i}{n})) \right\|_2^2 .$$

Champ	Expression
v_1	$v_1(x,y) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, if $-1 \leq x \leq 1$
v_2	$v_2(x,y) = \frac{1}{1+x^2} \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, if $-1 \leq x \leq 1$
v_3	$v_3(x,y) = \lvert x \rvert \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, if $-1 \leq x \leq 1$

FIGURE 3.13 – Les trois champs de vecteurs utilisés dans les tests.

On a ajouté les deux contraintes $\gamma(0) = A$ et $\gamma(1) = B$ qui s'expriment simplement en terme des points de contrôle par $P_{0,1} = A$ et $P_{N,d} = B$. On a aussi effectué des tests en forçant une régularité $\mathcal{C}^1$ aux points de raccordement des patches. Le gradient du critère et le gradient des contraintes est calculé par différentiation automatique par AMPL. On a choisi trois champs de vecteurs, listés dans le tableau en Figure 3.13, et illustrés par les Figures 3.14, 3.15 et 3.16. Les trois champs sont nuls en dehors de $[-1,1] \times \mathbb{R}$ et l'on a interdit les points de contrôle de sortir de cet ensemble en ajoutant des contraintes sur leurs abscisses.

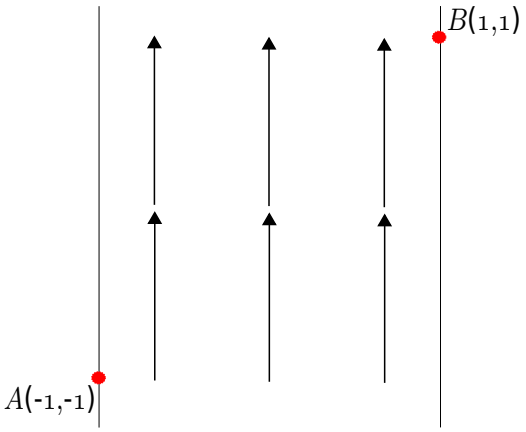


FIGURE 3.14 – Champ de vecteur v_1.

Chaque courbe de Bézier par morceaux est constituée de trois patches cubiques. Les résultats obtenus sont regroupés dans les trois Figures 3.17, 3.18 et 3.19. La courbe initiale est notée γ_{init} et est générée de façon aléatoire. La courbe γ_{sol} est la courbe vers laquelle l'algorithme a convergé. A chaque fois on a effectué des tests avec la contrainte de régularité $\mathcal{C}^1$ et d'autres avec la simple contrainte de continuité $\mathcal{C}^0$. On donne un exemple de trajectoires obtenues avec trois figures dans le cas du champ de vecteur v_3. La Figure 3.20 montre une courbe initiale et les Figures 3.21 et 3.22 donnent les trajectoires obtenues.

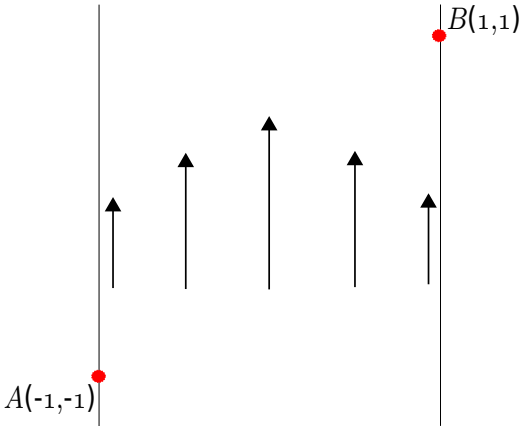


FIGURE 3.15 – Champ de vecteur v_2.

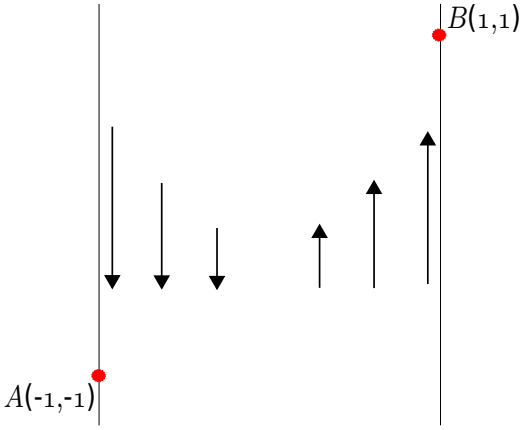


FIGURE 3.16 – Champ de vecteur v_3.

Test	$\mathcal{J}(\gamma_{init})$	$\mathcal{J}(\gamma_{sol})$ continuité $\mathcal{C}^0$	$\|\|\nabla L(\gamma_{sol})\|\|$ continuité $\mathcal{C}^0$	$\mathcal{J}(\gamma_{sol})$ continuité $\mathcal{C}^1$	$\|\|\nabla L(\gamma_{sol})\|\|$ continuité $\mathcal{C}^1$
1	93.8891	57.2268	5.425e-13	62.0520	9.946e-11
2	104.8780	57.2268	1.158e-14	62.0536	3.209e-05
3	102.2690	57.2268	4.902e-15	62.0520	1.3e-10
4	98.5732	57.2268	7.287e-14	62.0520	1.643e-10
5	100.4460	57.2268	1.093e-12	62.0525	4.276e-06
6	101.2860	57.2268	1.008e-15	62.0637	0.000132

FIGURE 3.17 – Résultats - Champ de vecteur v_1.

3.3.2 Génération de chemins des méthodes par homotopie

Le deuxième exemple de problème de trajectoires optimales est celui de la génération de chemins des méthodes par homotopie pour la résolution des systèmes algébriques [63, 12] et les références qui s'y trouvent, le calcul de valeurs propres [20]. Si l'ensemble des

Test	$\mathcal{J}(\gamma_{init})$	$\mathcal{J}(\gamma_{sol})$ continuité $\mathcal{C}^0$	$\|\|\nabla L(\gamma_{sol})\|\|$ continuité $\mathcal{C}^0$	$\mathcal{J}(\gamma_{sol})$ continuité $\mathcal{C}^1$	$\|\|\nabla L(\gamma_{sol})\|\|$ continuité $\mathcal{C}^1$
1	91.3621	64.8161	1.401e-12	68.1324	1.849e-10
2	99.5451	64.8161	1.58e-10	68.2118	0.0002592
3	95.8244	64.8161	4.423e-11	68.1324	2.436e-10
4	94.7440	64.8161	2.819e-11	68.1324	1.561e-10
5	95.2103	64.8161	3.07e-11	68.1324	5.789e-11
6	95.4909	64.8161	9.949e-11	68.1324	7.841e-11

FIGURE 3.18 – Résultats - Champ de vecteur v_2.

Test	$\mathcal{J}(\gamma_{init})$	$\mathcal{J}(\gamma_{sol})$ continuité $\mathcal{C}^0$	$\|\|\nabla L(\gamma_{sol})\|\|$ continuité $\mathcal{C}^0$	$\mathcal{J}(\gamma_{sol})$ continuité $\mathcal{C}^1$	$\|\|\nabla L(\gamma_{sol})\|\|$ continuité $\mathcal{C}^1$
1	84.6001	64.0525	1.101e-13	64.5407	1.683e-10
2	86.4755	64.0525	1.826e-11	64.5407	1.19e-10
3	85.1234	64.0525	2.927e-10	64.5407	5.668e-10
4	85.6600	64.0525	1.624e-10	64.5407	1.462e-10
5	84.5831	64.0525	2.303e-10	64.5407	7.468e-11
6	84.0690	64.0525	4.81e-10	64.5426	6.956e-05

FIGURE 3.19 – Résultats - Champ de vecteur v_3.

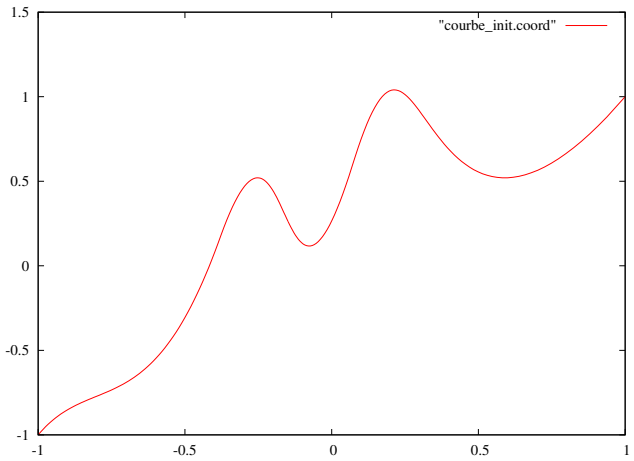


FIGURE 3.20 – Courbe initiale.

systèmes possède une structure d'espace vectoriel, supposons que l'on veuille résoudre un système B, on commence à partir d'un système A que l'on sait résoudre et on le déforme continuement jusqu'au problème B. Une solution est de considérer le segment $[A, B]$ et de suivre les solutions le long du segment : c'est l'homotopie linéaire. Cependant on risque de rencontrer un problème plus mal conditionné que le système B auquel on s'intéresse. L'idée est de ne pas se restreindre au segment mais à des chemins, de sorte à éviter le

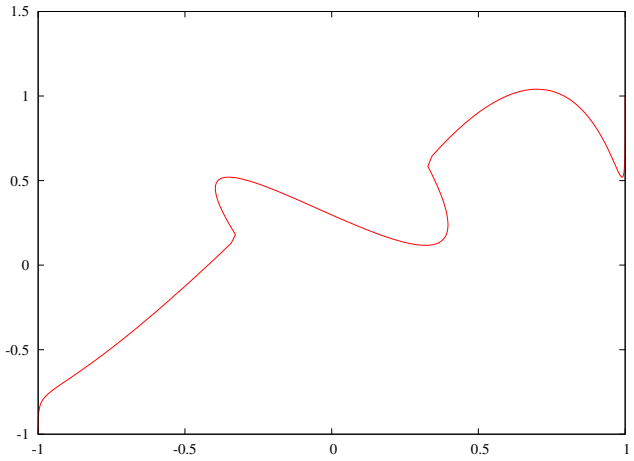


FIGURE 3.21 – Courbe obtenue dans le cas de la contrainte $\mathcal{C}^0$.

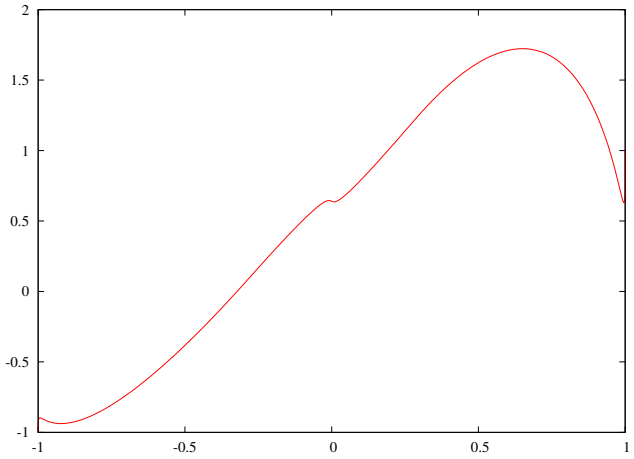


FIGURE 3.22 – Courbe obtenue dans le cas de la contrainte $\mathcal{C}^1$.

plus possible les systèmes mal conditionnés.

3.3.2.1 Position du problème

Ici E est l'ensemble des problèmes avec une structure d'espace vectoriel. On note $\Sigma \subset E$ l'ensemble des problèmes mal conditionnés et l'on note $\mu(v) = dist(v, \Sigma)$ pour tout $v \in E$ la distance d'un élément v à l'ensemble Σ. Etant donnés deux problèmes $A, B \in E$, où A est un problème que l'on sait résoudre et B est un problème que l'on ne sait pas résoudre, on cherche une courbe $m^* : [0,1] \to E$ telle que $m^*(0) = A$, $m^*(1) = B$ et m^* est le minimum de $\mathcal{J}(m) = \displaystyle\int_0^1 \mu(m(t))dt$ ou $\mathcal{J}(m) = \max_{0 \leq t \leq 1} \mu(m(t))$.

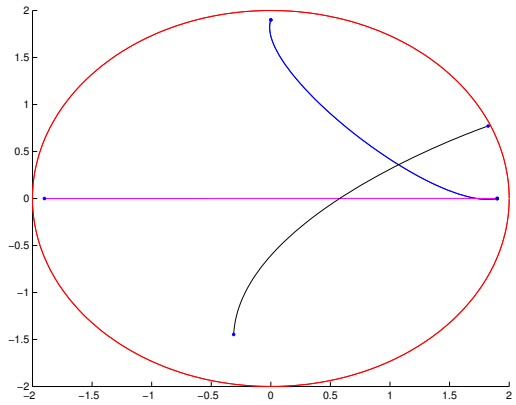


FIGURE 3.23 – Cas où Σ est un cercle.

3.3.2.2 Exemple dans $\mathbb{R}^2$

On a effectué des premiers tests dans $E = \mathbb{R}^2$ sur un exemple où Σ est un lieu géométrique. On cherche à relier deux points A et B du plan par le plus court chemin tout en restant le plus loin possible de Σ. Encore une fois, on utilise les courbes de Bézier pour approcher les trajectoires m et l'on a utilisé la fonction *fmincon* de Matlab pour résoudre le problème

$$\min_{P_0,...,P_d \in \mathbb{R}^2} \int_0^1 d_2(B[P_0, ..., P_d], t), \Sigma) dt$$

en laissant la fonction calculer le gradient. La distance d_2 est la distance euclidienne.

On présente les résultats obtenus avec les trois Figures 3.23, 3.24 et 3.25. Le lieu géométrique Σ apparait en rouge dans les deux figures. La Figure 3.23 montre trois trajectoires obtenues pour trois couples de points A, B dans le cas où Σ est un cercle. On voit que les courbes tendent vers les droites de la géométrie hyperbolique du disque de Poincaré. La Figure 3.24 montre deux trajectoires où Σ est la réunion de deux segments. La Figure 3.25 donne une trajectoire passant entre deux points à éviter. Dans chaque cas, on n'a utilisé qu'un seul patch cubique.

Les résultats présentés dans cette partie sur les trajectoires optimales proviennent de nos premiers travaux. Le but était de tester que l'utilisation des courbes de Bézier pour approcher les trajectoires donne des résultats attendus sur des problèmes concrets et simples. Ces résultats expérimentaux sont la première étape d'un travail plus théorique actuellement mené par Hoang Van Duc à XLIM. Le premier problème peut se généraliser aux problèmes de contrôle optimal où l'on approcherait aussi le contrôle par une courbe de Bézier par morceaux. Le second problème est comme on l'a évoqué voué à la généralisation de l'homotopie linéaire avec une amélioration au niveau numérique puisque les chemins sont calculés de sorte à être éloignés des problèmes mal conditionnés.

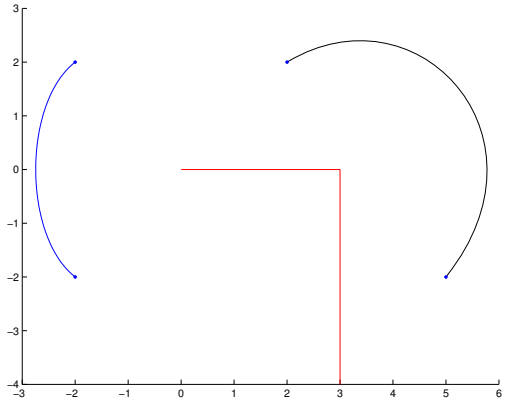


FIGURE 3.24 – Cas où Σ est la réunion de deux segments.

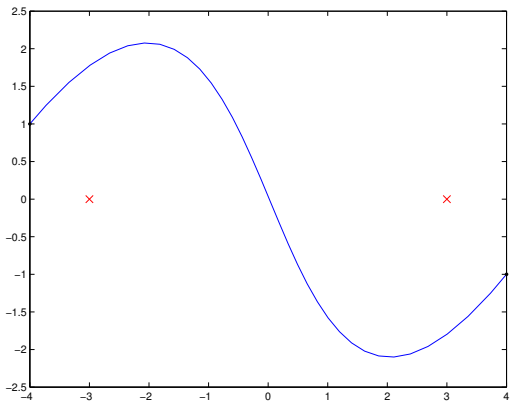


FIGURE 3.25 – Cas où Σ est la réunion de deux points.

Bibliographie

[1] L. Afraites, M. Dambrine, K. Eppler, and D. Kateb. Detecting perfectly insulated obstacles by shape optimization techniques of order two. *Discrete Contin. Dyn. Syst. Ser. B*, 8(2) :389–416 (electronic), 2007.

[2] G. Allaire. *Shape Optimization by the Homogenization Method*. Applied Mathematical Sciences. Springer New York, 2012.

[3] G. Allaire, C. Dapogny, and P. Frey. Shape optimization with a level set based mesh evolution method. *Comput. Methods Appl. Mech. Engrg.*, 282 :22–53, 2014.

[4] G. Allaire, F. de Gournay, F. Jouve, and A.-M. Toader. Structural optimization using topological and shape sensitivity via a level set method. *Control Cybernet.*, 34(1) :59–80, 2005.

[5] G. Allaire, F. Jouve, and A.-M. Toader. Structural optimization using sensitivity analysis and a level-set method. *Journal of computational physics*, 194(1) :363–393, 2004.

[6] G. Allaire, F. Jouve, and A.-M. Toader. Structural optimization using sensitivity analysis and a level-set method. *J. Comput. Phys.*, 194(1) :363–393, Feb. 2004.

[7] S. Amstutz. The topological asymptotic for the navier-stokes equations. *ESAIM : Control, Optimisation and Calculus of Variations*, 11(3) :401–425, 3 2010.

[8] S. Amstutz, I. Horchani, and M. Masmoudi. Crack detection by the topological gradient method. *Control and Cybernetics*, 34(1) :81–101, 2005.

[9] S. Amstutz, T. Takahashi, and B. Vexler. Topological sensitivity analysis for time-dependent problems. *ESAIM : Control, Optimisation and Calculus of Variations*, 14(3) :427–455, 11 2007.

[10] M. Badra, F. Caubet, and M. Dambrine. Detecting an obstacle immersed in a fluid by shape optimization methods. *Math. Models Methods Appl. Sci.*, 21(10) :2069–2101, 2011.

[11] M. Bauer, M. Bruveris, and P. W. Michor. Overview of the geometries of shape spaces and diffeomorphism groups. *J. Math. Imaging Vis.*, 50(1-2) :60–97, Sept. 2014.

[12] C. Beltrán and L. M. Pardo. Fast linear homotopy to find approximate zeros of polynomial systems. *Foundations of Computational Mathematics*, 11(1) :95–129, 2011.

[13] L. Bourgeois and J. Dardé. A quasi-reversibility approach to solve the inverse obstacle problem. *Inverse Probl. Imaging*, 4(3) :351–377, 2010.

[14] M. Burger, B. Hackl, and W. Ring. Incorporating topological derivatives into level set methods. *J. Comput. Phys.*, 194(1) :344–362, Feb. 2004.

[15] M. Burger, B. Hackl, and W. Ring. Incorporating topological derivatives into level set methods. *J. Comput. Phys.*, 194(1) :344–362, 2004.

[16] M. Burger and S. J. Osher. A survey on level set methods for inverse problems and optimal design. *European J. Appl. Math.*, 16(2) :263–301, 2005.

[17] F. Caubet. Instability of an inverse problem for the stationary Navier-Stokes equations. *SIAM J. Control Optim.*, 51(4) :2949–2975, 2013.

[18] F. Caubet, C. Conca, and M. Godoy. On the detection of several obstacles in 2d stokes flow : topological sensitivity and combination with shape derivatives. *Inverse Probl. Imaging*, 10(2) :327–367, 2016.

[19] A. N. Christiansen, M. Nobel-Jørgensen, N. Aage, O. Sigmund, and J. A. Bærentzen. Topology optimization using an explicit interface representation. *Structural and Multidisciplinary Optimization*, 49(3) :387–399, 2014.

[20] M. T. Chu. A simple application of the homotopy method to symmetric eigenvalue problems. *Linear Algebra and its Applications*, 59 :85 – 90, 1984.

[21] D. Colton and R. Kress. *Inverse acoustic and electromagnetic scattering theory*, volume 93 of *Applied Mathematical Sciences*. Springer-Verlag, Berlin, second edition, 1998.

[22] J. Dardé. *Quasi-reversibility and level set methods applied to elliptic inverse problems*. Theses, Université Paris-Diderot - Paris VII, Dec. 2010.

[23] M. J. de Ruiter and F. v. Keulen. Computational techniques for materials, composites and composite structures. chapter Topology of Optimization : Approaching the Material Distribution Problem Using a Topological Function Description, pages 111–119. Civil-Comp press, Edinburgh, UK, UK, 2000.

[24] M. de Schoenauer and G. Allaire. *Conception optimale de structures*. Mathématiques et Applications. Springer Berlin Heidelberg, 2006.

[25] T. A. El-mihoub, A. A. Hopgood, L. Nolle, and A. Battersby. Hybrid genetic algorithms : A review.

[26] H. A. Eschenauer, V. V. Kobelev, and A. Schumacher. Bubble method for topology and shape optimization of structures. *Structural optimization*, 8(1) :42–51, 1994.

[27] G. Farin. *Curves and Surfaces for CAGD : A Practical Guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2002.

[28] D. B. Fogel. *Evolutionary Computation : Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, USA, 1995.

[29] L. Fogel, A. Owens, and M. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons, 1966.

[30] S. Garreau, P. Guillaume, and M. Masmoudi. The topological asymptotic for pde systems : The elasticity case. *SIAM Journal on Control and Optimization*, 39(6) :1756–1778, 2001.

[31] S. Garreau, P. Guillaume, and M. Masmoudi. The topological asymptotic for PDE systems : the elasticity case. *SIAM J. Control Optim.*, 39(6) :1756–1778 (electronic), 2001.

[32] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[33] P. Guillaume and K. S. Idris. The topological asymptotic expansion for the dirichlet problem. *SIAM J. Control Optim.*, 41(4) :1042–1072, Apr. 2002.

[34] K. Gupta, R. Garg, and R. Chadha. *Computer-aided Design of Microwave Circuits*. Artech, 1981.

[35] S.-H. Ha and S. Cho. Level set based topological shape optimization of geometrically nonlinear structures using unstructured mesh. *Comput. Struct.*, 86(13-14) :1447–1455, July 2008.

[36] J. Hadamard. *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées*. Mémoires présentés par divers savants à l'Académie des sciences de l'Institut de France : Éxtrait. Imprimerie nationale, 1908.

[37] F. Hecht. Finite Element Library FREEFEM++. http ://www.freefem.org/ff++/.

[38] J. H. Holland. *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[39] Y.-L. Hsu. A review of structural shape optimization. *Comput. Ind.*, 25(1) :3–13, nov 1994.

[40] D. Hutton. *Fundamentals of Finite Element Analysis*. Engineering Series. McGraw-Hill, 2003.

[41] V. Isakov. *Inverse problems for partial differential equations*, volume 127. Springer Science & Business Media, 2006.

[42] J. M. Johnson. *Genetic Algorithms in Engineering Electromagnetics*. PhD thesis, 1997.

[43] H. Khalil. *Développement des techniques d'optimisation de forme pour la conception de composants hyperfréquences*. PhD thesis, Faculté des Sciences et Techniques de Limoges, 2009.

[44] S. Larnier, J. Fehrenbach, and M. Masmoudi. The topological gradient method : From optimal design to image processing. *Milan Journal of Mathematics*, 80(2) :411–441, 2012.

[45] N. Mahdi. *Développement d'une bibliothèque de techniques d'optimisation de formes pour la conception assistée par ordinateur de composants et de circuits hyperfréquences*. PhD thesis, Faculté des Sciences et Techniques de Limoges, 2012.

[46] A. Marco and J.-J. Martinez. A fast and accurate algorithm for solving bernstein–vandermonde linear systems. *Linear Algebra and its Applications*, 422(2) :616 – 628, 2007.

[47] M. Masmoudi, J. Pommier, and B. Samet. The topological asymptotic expansion for the maxwell equations and some applications. *Inverse Problems*, 21(2), 2005.

[48] F. Murat and J. Simon. *Quelques résultats sur le contrôle par un domaine géométrique*. VI Laboratoire d'Analyse Numérique, 1974.

[49] F. Murat and J. Simon. *Sur le contrôle par un domaine géométrique*. VI Laboratoire d'Analyse Numérique, 1976.

[50] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, New York, NY, USA, 2nd edition, 1998.

[51] S. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed : Algorithms based on Hamilton-Jacobi formulations. *JOURNAL OF COMPUTATIONAL PHYSICS*, 79(1) :12–49, 1988.

[52] S. J. Osher and F. Santosa. Level set methods for optimization problems involving geometry and constraints i. frequencies of a two-density inhomogeneous drum. *J. Comput. Phys.*, 171(1) :272–288, July 2001.

[53] O. Pantz and K. Trabelsi. Simultaneous shape, topology, and homogenized properties optimization. *Struct. Multidiscip. Optim.*, 34(4) :361–365, 2007.

[54] P.-O. Persson. *Mesh Generation for Implicit Geometries*. PhD thesis, Cambridge, MA, USA, 2005. AAI0807802.

[55] J. Pommier and B. Samet. The topological asymptotic for the helmholtz equation with dirichlet condition on the boundary of an arbitrarily shaped hole. *SIAM Journal on Control and Optimization*, 43(3) :899–921, 2004.

[56] I. Rechenberg. *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Problemata, 15. Frommann-Holzboog, 1973.

[57] A. Schumacher. *Topologieoptimierung von Bauteilstrukturen unter Verwendung von Lochpositionierungskriterien*. PhD thesis, Uni Siegen, 1995.

[58] A. Schumacher. Topologieoptimisierung von Bauteilstrukturen unter Verwendung von Lopchpositionierungkrieterien. *Thesis*, 1995. Universität-Gesamthochschule-Siegen.

[59] H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA, 1981.

[60] J. Sethian. *Level Set Methods and Fast Marching Methods : Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*.

Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 1999.

[61] J. Sethian. *Level set methods and fast marching methods. Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science.* Cambridge : Cambridge University Press, 1999.

[62] J. Sethian and A. Wiegmann. Structural boundary design via level set and immersed interface methods. *J. Comput. Phys.*, 163(2) :489–528, Sept. 2000.

[63] M. Shub. Complexity of bezout's theorem vi : Geodesics in the condition (number) metric. *Foundations of Computational Mathematics*, 9(2) :171–178, 2009.

[64] J. Sokolowski and A. Zochowski. On the topological derivative in shape optimization. *SIAM J. Control Optim.*, 37(4) :1251–1272, Apr. 1999.

[65] J. Sokołowski and A. Żochowski. On the topological derivative in shape optimization. *SIAM J. Control Optim.*, 37(4) :1251–1272 (electronic), 1999.

[66] P. Wei, M. Y. Wang, and X. Xing. A study on x-fem in continuum structural optimization using a level set model. *Comput. Aided Des.*, 42(8) :708–719, Aug. 2010.

[67] L. Younes. Shapes and diffeomorphisms, 2010.

Conclusion générale

Dans cette thèse nous nous sommes intéressés à une nouvelle méthode de déformation et de représentation des formes, applicable à un algorithme d'optimisation de formes. Notre méthode est de nature géométrique car elle utilise le principe de variation de la frontière. Elle possède également un aspect topologique car l'on propose un moyen de séparer une forme en deux ou de réunir deux formes en une.

Afin de tester notre technique de déformation, nous nous sommes penchés sur plusieurs problèmes d'optimisation de formes particuliers. Un premier problème dans le domaine de l'électromagnétisme sur la conception d'un filtre micro-ondes passe-bande, un deuxième problème sur la détection d'objets immergés dans un fluide et un troisième problème concernant les trajectoires optimales.

Pour le premier problème d'électromagnétisme, nous avons travaillé avec l'équipe MINACOM du laboratoire XLIM. En utilisant leur logiciel EMXD effectuant la résolution du problème direct et le calcul du gradient topologique, nous avons greffé notre code de déformation pour construire un algorithme de type de descente suivant le gradient. Nous avons testé cet algorithme sur un exemple où le but était de retrouver une allure de paramètre-S correspondant à un dépôt de métal centré dans le domaine d'optimisation. La nature géométrique de notre méthode semble incompatible avec l'information topologique du gradient calculé par EMXD. Nous avons alors entrepris de calculer un gradient de forme qui est plus adapté à notre méthode de déformation. Un travail est actuellement en cours avec la société CST Microwave pour obtenir un gradient géométrique et développer notre technique de déformation.

Nous nous sommes ensuite intéressés au problème de détection d'objets immergés dans un fluide. Grâce au choix de paramétrisation par les courbes de Bézier par morceaux, nous avons pu tester la détection d'objets lisses, non-lisses et non-convexes. Nous avons également pu détecter deux obstacles en partant d'une forme initiale connexe grâce à la technique du flip. Il serait intéressant de développer notre méthode pour la reconstruction d'objets en trois dimensions et de chercher à détecter plus de deux objets, toujours en partant d'une forme initale connexe.

Une troisième application a été celle des problèmes de trajectoires otimales. Notre méthode basée sur les courbes de Bézier n'est en effet pas restreinte à des problèmes d'optimisation de formes classiques mais peut s'étendre à la recherche de chemins, les formes étant des courbes de Bézier par morceaux ouvertes. Nous avons choisi deux problèmes de trajectoires optimales, un premier lié à la recherche de chemin minimisant une dépense d'énergie et un second lié à la recherche de chemin dans des espaces de problèmes. Ces premiers travaux ont été réalisés sur exemples simples. Nous souhaitons développer nos recherches à des problèmes de contrôle optimal et à la génération de chemins pour des méthodes de résolution de systèmes par homotopie.