Modèle géométrique déformable pour la simulation et l'optimisation automatique de forme

Elisa Berrini

École doctorale de Sciences Fondamentales et Appliquées
INRIA Sophia Antipolis, équipe AROMATH

Thèse de doctorat

Présentée en vue de l'obtention du
grade de docteur en Mathématiques
de
l'UNIVERSITE COTE D'AZUR

par
Elisa Berrini

Modèle géométrique déformable pour la simulation et l'optimisation automatique de forme

Dirigée par Bernard Mourrain

Soutenue le 7 Juin 2017
Devant le jury composé de :

M. Desideri Jean-Antoine	DR	Inria Sophia Antipolis	Président
M. Kaklis Panagiotis	PR	NTUA and University of Strathclyde	Rapporteur
M. Léon Jean-Claude	PR	ENSE3-Grenoble-INP	Rapporteur
M. Mourrain Bernard	DR	Inria Sophia Antipolis	Directeur de thèse
M. Roux Yann	PhD	MyCFD	Examinateur
M. Visonneau Michel	DR	CNRS, École Centrale de Nantes	Examinateur

Remerciements

Je tiens à remercier mes encadrants de thèse pour le support et le suivi de mon travail tout au long de ces années. Je remercie Bernard Mourrain, mon directeur de thèse, pour m'avoir donné la possibilité de faire cette thèse, de son soutien, sa patience et ses encouragements. Je remercie Yann Roux, directeur de MyCFD, pour m'avoir encouragé à commencer une thèse, et pour m'avoir fait comprendre l'importance de la recherche y compris dans le milieu industriel.

Je tiens à remercier Panagiotis Kaklis et Jean-Claude Léon d'avoir accepté d'être les rapporteurs de cette thèse, Michel Visonneau et Jean-Antoine Desideri d'avoir accepté de participer au jury.

Cette thèse n'aurait jamais pu se faire sans l'aide de toute l'équipe de K-Epsilon. Guillaume, merci pour le temps que tu as passé à m'aider et ton support dans les moments difficiles de la thèse. Merci à David, Corentin, Catherine, Mathieu, Mike, Delphine pour la cohésion d'équipe forte et enrichissante tant au niveau humain que technique. Merci aussi à tous les stagiaires de K-Epsilon, qui participent toujours à l'ambiance de l'entreprise. Je remercie Matthieu pour nos travaux sur les foils, et Emmanuel pour le support que tu m'as apporté à la fin de la rédaction.

L'équipe AROMATH (et GALAAD) n'est pas en reste pour l'aide et le support qu'ils m'ont donné durant la thèse. Je remercie les permanents de l'équipe, Laurent, Evelyne, André, pour avoir suivi mes travaux du début jusqu'à la fin. Je remercie Sophie, grâce à qui tout devient facile et simple. Évidement, merci à toute l'équipe présente et passée Mathieu, Anais, Fatmanur, Alessandro, Ahmed, Alvaro, Jouhayna. Je vous remercie pour les bons moments passés ensemble.

Je remercie également Régis Duvigneau, qui a suivi l'avancement du projet et a toujours apporté des conseils pertinents et avisés.

Finalement, je ne remercierai jamais assez ma famille et mes amis pour leur patience et leur support tout au long de ces années. Merci à ma mère, qui a lu chacune de ces pages avec sa propre interprétation. Merci à ma sœur et à mon père qui m'ont apporté un support sans faille depuis le début. David, merci de m'avoir soutenue et supportée sans relâche.

Résumé

Le contrôle précis des modèles géométriques joue un rôle important dans de nombreux domaines comme la Conception Assistée par Ordinateur et la simulation numérique. Pour l'optimisation de forme en mécanique des fluides, le choix des paramètres de contrôle et la technique de déformation de forme est critique.

Dans cette thèse, nous proposons un modeleur paramétrique avec une nouvelle méthode de déformation d'objets, ayant pour objectif d'être intégré dans une boucle d'optimisation automatique de forme avec un solveur CFD (simulation en mécanique des fluide, ou Computational Fluid Dynamics en anglais). Notre méthodologie est basée sur une double paramétrisation des objets : géométrique et architecturale. L'approche géométrique consiste à décrire les formes par un squelette. Ce squelette est composé d'une famille de courbes B-Splines, appelées courbes génératrice et courbes de section. Le squelette est paramétré avec une approche architecturale. Des paramètres de design pertinents sont choisis sur l'objet étudié. Ainsi, au lieu d'utiliser les points de contrôle de la représentation classique par courbes B-Splines, la géométrie est contrôlée par ces paramètres architecturaux. Cela permet de réduire considérablement le nombre de degrés de liberté utilisés dans le problème d'optimisation de forme, et permet de maintenir une description haut niveau des objets. Notre technique intègre un contrôle de forme et un contrôle de régularité, permettant d'assurer la génération de nouvelles formes valides et réalistes. Les déformations de la géométrie sont réalisées en posant un problème inverse : déterminer une géométrie correspondant à un jeu de paramètres cibles. En pratique, pour résoudre ce problème nous résolvons plusieurs systèmes de minimisation avec pour inconnues les coordonnées des points de contrôle des courbes du squelette. Enfin, une technique de reconstruction de surface est proposée, permettant d'évaluer les performances d'une forme avec un solveur CFD basé sur un maillage volumique.

Nous illustrons le modeleur paramétrique développé sur trois cas : un profil d'aile d'avion, un foil AC45 d'un voilier de course et un bulbe de chalutier de pêche. Pour chaque cas, nous obtenons un ensemble de géométries déformées dont nous évaluons les caractéristiques hydrodynamiques avec un solveur numérique, différent pour chacun des trois cas. Les performances de chaque forme sont analysées. Les cas du profil d'aile d'avion et de l'AC45 ont été entièrement automatisés, montrant des applications fonctionnelles d'une boucle d'optimisation automatique de forme.

Mots clés

Modeleur paramétrique, modèle géométrique, optimisation automatique de forme, CAO, solveur numériques, CFD, architecture navale

Geometric modelling and deformation for automatic shape optimisation

Abstract

The precise control of geometric models plays an important role in many domains such as Computer Aided Geometric Design and numerical simulation. For shape optimisation in Computational Fluid Dynamics (CFD), the choice of control parameters and the way to deform a shape are critical.

In this thesis, we propose a new approach to shape deformation for parametric modellers with the purpose of being integrated into an automatic shape optimisation loop with a CFD solver. Our methodology is based on a twofold parameterisation : geometrical and architectural. The geometrical approach consist of a skeleton-based representation of object. The skeleton is made of a family of B-Spline curves, called generating curve and section curves. The skeleton is parametrised with an architectural approach : meaningful design parameters are chosen on the studied object. Thus, instead of using the control points of a classical B-spline representation, we control the geometry in terms of architectural parameters. This reduce the number of degrees of freedom and maintain a high level description of shapes. We ensure to generate valid shapes with a strong shape consistency control based on architectural considerations. Deformations of the geometry are performed by solving optimisation problems on the skeleton. Finally, a surface reconstruction method is proposed to evaluate the shape's performances with CFD solvers.

We illustrate the parametric modeller capabilities on three problems : the wind section of an plane (airfoil), the foil of an AC45 racing sail boat and the bulbous bow of a fishing trawler. For each case, we obtained a set of shape deformations and then we evaluated and analysed the performances of the shapes with different CFD solvers. The airfoil and the AC45 cases of study were fully automated, showing functional automatic shape optimisation loops.

Keywords

Parametric modeller, geometrical model, automatic shape optimization, CAD, numerical solvers, CFD, naval architecture

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Introduction générale

L'architecture navale est l'art de concevoir des structures pour la navigation en mer ou en fleuve. Jusqu'à l'apparition de la conception assistée par ordinateur (CAO) dans les années 1970, les architectes réalisaient manuellement toutes les étapes de la *spirale de conception* d'un bateau, illustrée synthétiquement dans la figure 1.1. L'intérêt de cette spirale est de vérifier pour chaque modification apportée au dessin que les autres aspects du cahier des charges sont respectés. Pour chaque étape du design (conceptuel, préliminaire et final), plusieurs tours de boucles sont effectués pour converger vers un modèle respectant toutes les contraintes.

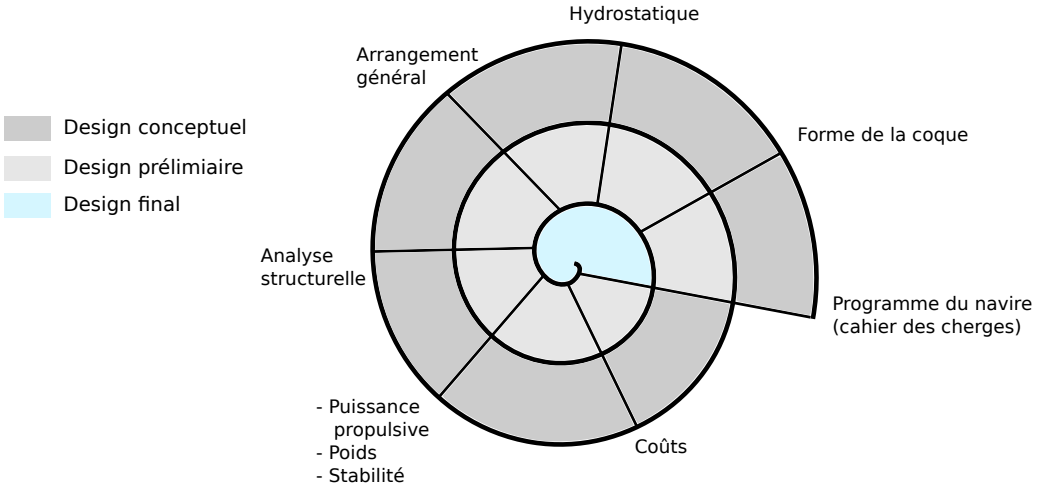


FIGURE 1.1 – Spirale de conception architecturale

Pour apporter des modifications pertinentes au design de la coque, les architectes se sont longtemps basés sur leur expérience et leur intuition, sans recourir à un cadre scientifique précis.

Les essais en bassins de carène ont été pour la première fois utilisés par William Froude dans les années 1860 [Froude and Froude, 1888, Michell, 1898]. Grâce à la possibilité d'effectuer des essais sur modèles réduits, les architectes ont intégré une composante expérimentale dans leurs techniques d'optimisation de design. Cependant, la mise en place d'essais en bassin est très coûteuse et ne peut pas être utilisée systématiquement.

Avec l'arrivée de la CAO puis des codes de calculs numériques dans les années 1970/1980, l'utilisation d'outils numériques s'est peu à peu démocratisée [Dawson, 1977, van Oossanen,

1985]. Des logiciels de prédiction de la vitesse des voiliers (VPP, *velocity prediction program*) ont été développés, en utilisant des méthodes de calculs d'écoulements fluide de plus en plus précis et performants. A ce stade, les architectes utilisent ces outils numériques pour réaliser des modifications de design manuellement. Ils effectuent quelques boucles de la spirale de conception avec les nouveaux designs et concluent sur une forme optimale par rapport aux contraintes de leur projet (par exemple, la réduction de la consommation par rapport à une forme de navire existant).

Depuis le début des années 2000, les recherches s'orientent vers l'automatisation du processus d'optimisation de la forme de la coque. Le but est de concevoir un outil capable de réaliser de façon autonome une optimisation automatique de forme, en tenant compte de toutes les autres contraintes du cahier des charges.

L'optimisation automatique de forme en général est un domaine d'étude en croissance, avec des applications dans des secteurs industriels variés tels que l'aéronautique, l'automobile, l'industrie navale, l'offshore, les énergies renouvelables ou encore le BTP. La recherche de l'amélioration des performances est motivée principalement par la réduction de la consommation d'énergie. Le coût élevé et la raréfaction des énergies fossiles poussent les constructeurs à développer des concepts à la fois performants et économes.

Pour réaliser une optimisation automatique de forme, il est usuel de construire une boucle automatisée. Différents éléments ont besoin d'être développés et liés pour concevoir cette boucle. Il s'agit de réunir un modeleur paramétrique, un outil d'analyse de performances et un algorithme d'optimisation adaptés au problème considéré.

Le travail de cette thèse se focalise sur la conception d'un modeleur paramétrique.

Un modeleur paramétrique est un outil capable de générer des variations de forme d'un objet à partir d'un ensemble de paramètres. Dans cette thèse nous proposons une méthode permettant de paramétriser et déformer un objet existant, c'est-à-dire de produire une série de formes filles à partir d'une forme mère initiale. Cette approche s'oppose à la création de formes libres, sans modèle initial. Ici le but est d'améliorer une géométrie existante, en proposant des variations dans un espace autour de la forme initiale.

En termes de temps passé dans la spirale de conception d'un bateau, le design conceptuel prend environ 5% du temps total passé sur le projet. Le design préliminaire prend 10% et le design final demande 85%. Le modeleur paramétrique que nous avons développé a pour but d'être utilisé à la dernière étape, pour finaliser le design. C'est là où le gain de temps potentiel est le plus important.

Le modeleur que nous proposons est générique et adapté à l'utilisation dans une boucle d'optimisation automatique.

La généricité est induite par la méthode de description et de paramétrisation des formes mise en place. Cela permet d'utiliser le modeleur sur des applications très variées. La paramétrisation proposée est double :

- **géométrique**, en se basant sur une représentation des objets par un squelette,
- **architecturale**, en utilisant un ensemble de paramètres métier choisis pour leur importance dans le design ou pour l'influence qu'ils ont sur les performances de l'objet.

Pour s'adapter à une boucle d'optimisation, le modeleur se base sur un petit nombre de paramètres de variation de forme, et donc un petit nombre de degrés de liberté pour l'algorithme d'optimisation, tout en assurant un contrôle total de la forme. Ces paramètres sont ceux issus de la paramétrisation architecturale de l'objet.

Cette thèse CIFRE est le résultat d'un travail collaboratif entre l'Inria et l'entreprise MyCFD, société spécialisée dans l'automatisation des calculs hydrodynamiques et aérodynamiques. Le domaine d'activité de MyCFD est principalement naval, et l'entreprise souhaite développer un service d'optimisation automatique de forme pour compléter ses offres commerciales d'automatisation de simulations. Comme il sera expliqué au chapitre 2, l'absence de logiciels ou de méthodes systématiques disponibles pour réaliser cette tache justifie les développements réalisés dans cette thèse.

Étant donné le domaine d'expertise naval de l'entreprise, les applications que nous proposons dans cette thèse porteront sur les coques de bateau et leurs appendices (foil, bulbe, safran, mât, etc.). Un effort important a été fournis sur la généricité du modeleur paramétrique. Grâce à cet généricité, le modeleur pourra être étendu à d'autres applications.

Dans la boucle d'optimisation automatique de forme adaptée aux applications proposées, l'analyse de performance est fournie par une simulation numérique résolvant les équations de la mécanique des fluides, *Computational Fluid Dynamics (CFD)* en anglais. Ce type de solveurs nécessite l'intégration d'une étape de maillage pour discrétiser la géométrie avant de l'analyser. L'algorithme d'optimisation choisi dépend du type de problème traité, mono ou multi-objectifs, et du coût de calcul du solveur numérique. Nous détaillons les solveurs numériques et les algorithmes d'optimisation utilisés dans les chapitres 7 et 8.

La figure 1.2 schématise la boucle d'optimisation automatique.

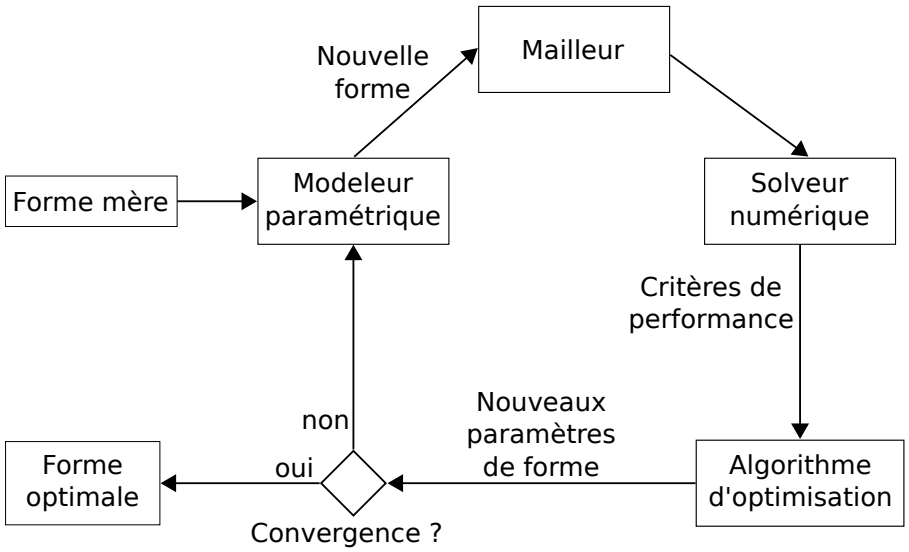


FIGURE 1.2 – Boucle d'optimisation automatique de forme

Notre modeleur nécessite que la forme mère, géométrie d'entrée, soit décrite par des courbes et surfaces **NURBS** (B-Splines rationnelles non uniformes, *Non-Uniform Rational Basis Splines* en anglais) ou **B-Splines**. La plupart des logiciels de CAO utilisent ce format, ce qui le rend très repandu parmi les communautés d'architectes et designers industriels. En utilisant le format

NURBS comme entrée de notre modeleur, nous pouvons être indépendant du logiciel de CAO utilisé pour dessiner la forme.

Les NURBS sont un format de description de courbes et surfaces paramétriques. Les B-Splines sont polynomiales, et les NURBS sont en réalité une extension des B-Spline à la représentation de courbes et surfaces rationnelles. Les "coefficients" des courbes et surfaces B-Splines (ou NURBS) sont appelés *points de contrôle* et ont la particularité d'avoir une signification géométrique. En effet, les points de contrôle permettent de modifier intuitivement l'allure des courbes et des surfaces.

De nombreux algorithmes et extensions ont été développés pour ce type de courbes et surfaces, amenant à une utilisation intensive de ces descriptions dans les logiciels de CAO.

De part ces constatations, nous allons baser notre méthode de paramétrisation et de déformation sur ce type de représentations.

1.2 Description de la méthode

Les performances d'un objet soumis à un écoulement fluide seront calculées avec précision grâce à la CFD. L'impact de modifications de la forme de l'objet sur les caractéristiques hydrodynamiques ou aérodynamiques peut être capturé par la simulation et analysé. Pour exploiter au mieux ces capacités d'analyse, il est important d'avoir un contrôle précis et efficace de la géométrie des objets étudiés. Des stratégies d'optimisation peuvent être mises en place pour déterminer une forme optimale comme illustré par la figure 1.2.

Les solveurs numériques et les algorithmes d'optimisation ont été largement étudiés et des solutions adaptées à l'automatisation d'un grand nombre de calculs ont été développées. Cependant, peu de travaux ont été menés sur les modeleurs paramétriques. Leur rôle est crucial pour l'exploration de l'espace des formes possibles dans la recherche d'un optimum. De plus, le modeleur paramétrique est l'intermédiaire entre la CAO et la CFD. La problématique de la compatibilité entre ces deux domaines est importante et communément identifiée [Aftosmis et al., 1999, Samareh, 2005]. Pour être utilisé avec un algorithme d'optimisation, le modeleur doit être en mesure de modifier une géométrie en utilisant peu de paramètres (moins d'une vingtaine). Il doit fournir un contrôle précis de la forme, tout en permettant de générer des formes admissibles variées.

L'approche que nous proposons dans cette thèse permet de générer des formes valides d'un point de vue géométrique et architectural, grâce à une méthode de contrôle de la cohérence globale de la forme basée sur la double paramétrisation d'un objet.

Géométriquement, nous représentons l'objet par un ***squelette***, composé d'une courbe génératrice et de courbes de section.

Le contrôle de la géométrie est réalisé en inversant une application, dite ***fonction observer***, qui associe un ensemble de paramètres métier à la géométrie. Cette application inverse est obtenue en résolvant un système de minimisation composé d'équations découplées pour la génératrice et les sections.

Les paramètres métier pilotent donc directement les déformations de l'objet.

Le système de minimisation proposé intègre un contrôle de forme et un terme de régularité, permettant d'assurer la génération de nouvelles formes valides et réalistes.

Finalement, une méthode de reconstruction de surfaces est proposée, permettant d'obtenir une représentation de l'objet déformée en 3D sous forme de patchs de surface B-Splines. Ce type de représentation est utilisé par certains logiciels de simulation numérique.

Notre outil propose une méthodologie pour lier la représentation géométrique d'un objet, ses déformations et la simulation numérique.

Travailler avec des paramètres architecturaux plutôt qu'avec les points de contrôle des NURBS de la géométrie se justifie pour plusieurs raisons :

- Le nombre de degrés de liberté vu par l'optimisation automatique de forme est considérablement réduit.
- Ces paramètres ont une signification physique, permettant de rendre l'analyse des résultats de l'optimisation plus explicite. De plus, fixer les limites de variation des paramètres est beaucoup plus facile.
- Les paramètres de forme sont indépendants les uns des autres. Les points de contrôle sont aussi indépendants les uns des autres, mais leur effet sur la forme ne l'est pas. Pour modifier un paramètre de forme architectural, il est nécessaire de modifier une famille de points de contrôle. L'absence de corrélation entre les variables permet à l'algorithme d'optimisation de converger plus rapidement vers une solution stable.
- Enfin, le caractère intuitif de ces paramètres pour les architectes est essentiel car ils seront les utilisateurs finaux du modeleur.

Durant la thèse, un logiciel a été développé permettant de mettre en pratique la méthodologie proposée. Ce logiciel est principalement codé en MATLAB. Une attention particulière a été portée à la généricité du code afin de permettre une transition vers le C++, dont l'exécution est rapide. Une architecture semblable à un arbre d'héritage a été implémentée, permettant de classer les différentes applications sous de grandes familles d'objets plus générales. Certaines fonctionnalités ont d'ailleurs déjà été implémentées en C++.

Le code est entièrement automatisé et disponible sous forme d'exécutable multi-plateforme. Les temps d'exécution actuels sont déjà courts et permettent une utilisation industrielle.

1.3 Contributions

Les contributions majeures de cette thèse peuvent être résumées comme suit :

- La construction d'un modèle géométrique générique, permettant de décrire les formes de façon précise et efficace.
- La capacité de déformer des géométries à partir d'un petit nombre de paramètres de contrôle pertinents.
- La capacité de conserver une cohérence géométrique et architecturale des formes, de telle façon que toute forme générée soit valide, réalisable et utilisable dans un logiciel de simulation.
- L'adéquation de ce modèle aux solveurs numériques et aux algorithmes d'optimisation.
- L'automatisation de toutes les tâches composant le modeleur paramétrique développé et son intégration dans une boucle d'optimisation automatique de forme, avec des temps d'exécution adaptés pour l'industrialisation de l'outil.

- La réalisation d'expérimentations qui montrent la validité, la faisabilité et l'intérêt de l'approche développée. Les cas tests traités sont des cas réalistes, ils sont issus de projets industriels.

Le travail de cette thèse a donné lieu à la publication de quatre articles :

- **COMPIT'16** : 15th International Conference on Computer Applications and Information Technology in the Maritime Industries.
  Présentation et publication dans les *procceeding* de la conférence du papier intitulé : *Parametric shape modeler for hulls and appendages*, E. Berrini, B. Mourrain, Y. Roux, G. Fontaine, E. Jean.

  Thème du papier : présentation des résultats de l'optimisation de forme d'un bulbe de chalutier.

- **MARINE 2017** : VII International Conference on Computational Methods in Marine Engineering.
  Présentation et publication dans les *procceeding* de la conférence du papier intitulé : *Geometric model for automated multi-objective optimization of foils*, E. Berrini, B. Mourrain, R. Duvigneau, M. Sacher, Y. Roux.

  Thème du papier : présentation des résultats de l'optimisation de forme automatique et multi-objectif d'un foil de voilier de course.

- **Journal of Ship Research** : journal édité par Society of Naval Architects and Marine Engineers.
  Soumission d'une publication intitulée : *Geometric modelling and deformation for shape optimization of ship hulls and appendages*, E. Berrini, B. Mourrain, Y. Roux, M. Durand, G. Fontaine. (accepté le 09/04/2017, publication dans le volume 61 numéro 2).

  Thème du papier : présentation détaillée du modeleur paramétrique, et présentation des deux cas d'expérimentation (foil de voilier de course et bulbe de chalutier).

- **INNOV'SAIL 2017** : International Conference on Innovation in High Performance Sailing Yachts, possibilité de publication dans une édition spéciale du journal **Ocean Enginnering**, édité par Elsevier.
  Présentation et publication dans les *procceeding* de la conférence du papier intitulé : *Flexible hydrofoil optimization for the 35th America's Cup with constrained EGO method*, M. Sacher, M. Durand, E. Berrini, F. Hauville, R. Duvigneau, O. Le Maître, J. A. Astolfi.

  Thème du papier : optimisation en interaction fluide-structure du foil utilisé par les voiliers de la Coupe de l'America.

1.4 Plan de la thèse

La thèse est organisée autour des chapitres suivants :

- Le **chapitre 2** présente l'état de l'art, et les méthodes existantes pour réaliser des déformations.
- Le **chapitre 3** introduit les notions de bases sur les courbes et surfaces B-Splines, utilisées tout au long de la thèse. Il présente aussi l'implémentation que nous avons réalisée des techniques connues d'approximation de courbes et surfaces.
- Le **chapitre 4** présente la première partie du travail réalisé sur le modeleur : la paramétrisation de formes. Nous détaillons la construction du squelette et des fonctions *observer*, qui permettent d'associer un ensemble de paramètres métier à une géométrie.
- Le **chapitre 5** détaille le fonctionnement opérationnel du processus de déformation que nous avons mis en place.
- Le **chapitre 6** présente plusieurs techniques de reconstructions de surfaces à partir du squelette, pour obtenir une représentation de l'objet déformé compatible avec certains solveurs CFD.
- Le **chapitre 7** donne une vision globale de la boucle d'optimisation automatique de forme. Nous présentons les solveurs numériques utilisés et introduisons des notions sur les algorithmes d'optimisation mono et multi-objectifs.
- Enfin, le **chapitre 8** est consacré aux applications réalisées avec le modeleur : l'optimisation automatique de forme d'un profil d'aile d'avion et d'un foil, et la réalisation d'un plan d'expérience pour la variation de forme d'un bulbe.

Dans la suite, tous les temps de calculs sont donnés pour un HP Probook-450 de quatre cœurs et un Intel® Core™ i7-4702MQ CPU 2.20GHZ, RAM 8.00 GB.

Chapitre 2

État de l'art

Les méthodes numériques de déformation de formes pour les applications navales sont relativement récentes. Lackenby [Lackenby, 1950] est le premier à proposer une méthode de variation de forme numérique pour les coques de bateau. Les procédés de déformation automatique et d'optimisation de forme apparaissent bien plus tard. Des techniques développées dans d'autres domaines d'applications, notamment l'animation 3D, ont d'abord été utilisées, puis des logiciels spécialisés ont été développés.

Dans ce chapitre, nous exposons les différentes méthodes de déformation et optimisation de forme appliquées aux coques de bateau et appendices.

Les logiciels de CAO utilisés par les architectes sont basés sur la description de courbes et surfaces avec des NURBS ou des B-Splines.

Les NURBS sont un type de description de courbes et surfaces paramétriques [Piegl and Tiller, 1997]. Elles sont caractérisées par l'utilisation de points de contrôles et de vecteurs de nœuds, détaillés dans le chapitre 3. Les NURBS sont une généralisation des B-Splines, qui elles-mêmes généralisent les courbes et surfaces de Bézier.

Pierre Bézier introduit la notion de points de contrôles pour permettre une manipulation facile et intuitive des courbes [Piegl and Tiller, 1997]. Une courbe de Bézier est une combinaison linéaire de fonctions de bases polynomiales, dont les coefficents sont appelés points de contrôle. Avec cette définition, les points de contrôle deviennent des "poignées de contrôle" de la courbe, que l'on peut modifier intuitivement. Pour une courbe de Bézier, l'influence des point de contrôle est globale, c'est-à-dire que la modification d'un des points modifie l'allure de toute la courbe.

Les courbes B-Splines ont été introduites pour pallier à certaines limitations des Bézier. Tout d'abord, pour modéliser des formes complexes ou pour assurer la continuité entre plusieurs courbes, la seule solution est d'augmenter le degré des courbes de Bézier. Les B-Splines permettent de conserver un degré faible, tout en facilitant la conception d'objet complexe. Grâce à l'introduction des vecteurs de nœuds, l'influence des points de contrôle d'une B-Spline est local. Les vecteurs de nœuds définissent les fonctions de base des B-Splines et en pratique il définit la zone d'influence de chaque point de contrôle sur la courbe. Nous donnons les définitions mathématiques de ces courbes dans le chapitre 3.

Enfin, les NURBS ont été introduites pour généraliser les B-Spline, qui sont des courbes polynomiales, et permettre de représenter des courbes rationnelles.

D'autre part, les logiciels de simulation numérique sont basés sur une description des objets avec des maillages, surfaciques ou volumiques.

Nous détaillons principalement les méthodes de déformation adaptées pour ces deux types de descriptions.

Nous divisons les méthodes décrites en deux catégories :

- les méthodes *"géométriques"*, c'est-à-dire les techniques génériques qui n'ont pas été développées spécialement pour les navires et nécessitent donc une expertise en CAO,
- les méthodes *"métier"*, c'est-à-dire les techniques ou logiciels développés en vue d'une application ciblée aux coques et à leurs appendices et pour laquelle l'expertise seule de l'architecte dans son domaine est suffisante.

2.1 Méthodes géométriques

Morphing

Le ***morphing*** consiste à interpoler une série de formes intermédiaires à partir de deux formes extrémales. Le *morphing* a d'abord été appliqué à des maillages [Kaul and Rossignac, 1991, Kent et al., 1992, Zockler et al., 2000, Alexa, 2002], puis des extensions aux courbes et surfaces continues ont été développées [Johan et al., 2000, Ju and Goldman, 2003], comme illustré dans la figure 2.1

La méthode la plus simple de transformation se base sur une interpolation linéaire entre les points de la forme initiale et ceux de la forme finale. D'autres méthodes, adaptées spécialement pour les courbes et surfaces NURBS permettent d'utiliser les caractéristiques de ces modèles, par exemple les poids dans le travail de [Ju and Goldman, 2003].

Des applications du morphing au domaine naval sont présentées dans [Tahara et al., 2008, Peri et al., 2009, Baiwei et al., 2011, Kang and Lee, 2012, Hock et al., 2016].

Le domaine d'exploration de formes du *morphing* est borné par les deux formes extrémales. Cette méthode permet d'obtenir rapidement des formes si l'architecte a une idée précise des limites de variations. Malgré tout, l'espace des formes atteignables est restreint.

FIGURE 2.1 – Illustration d'un processus de morphing appliqué sur une surface B-Spline, issue de l'article [Ju and Goldman, 2003]

Free Form Deformation

La méthode de ***Free Form Deformation*** ou ***FFD*** consiste à déformer un objet en modifiant une cage de contrôle qui l'enveloppe [Sederberg and Parry, 1986]. La cage de contrôle

est définie comme polyèdre englobant la forme initiale. Ensuite, une correspondance entre les points du polyèdre et les points de l'objet est établie. Plusieurs techniques de paramétrisation existent pour définir cette correspondance [Hormann, 2000, Desbrun et al., 2002, Ju et al., 2005]. Lorsque la cage de contrôle est modifiée, les nouvelles coordonnées des points de l'objet à l'intérieur sont calculées et l'objet se déforme. La figure 2.2 illustre une déformation par FFD.

La *FFD* peut être appliquée à des maillages ou a des surfaces continues. Des applications de la *FFD* au domaine naval sont présentées dans [Duvigneau and Visonneau, 2003, Mason and Thomas, 2007, Peri et al., 2009, Geremia et al., 2011, Peri and Diez, 2013, Rozza et al., 2013, Brizzolara et al., 2015, Ang et al., 2015].

La *FFD* est une méthode relativement simple à mettre en place, et efficace pour obtenir des formes variées, d'où son utilisation intensive. La *FFD* peut être très efficace avec un petit nombre de degrés de liberté contrôlant l'objet. Cependant pour réaliser des déformations locales ou plus précises, la seule solution est d'augmenter le nombre de points de contrôle en raffinant les zones d'intérêt et donc d'augmenter le nombre de degrés de liberté.

Les paramètres de variation de forme de la *FFD* sont les coordonnées des points de contrôle de la cage. Ces coordonnées n'ont pas de sens physiques ou métier par rapport à l'objet déformé. Cela limite leur pertinence d'un point de vue utilisateur. Pour une optimisation de forme il est plus difficile de déterminer le domaine de variation des paramètres. Il est aussi plus difficile d'analyser la corrélation et la sensibilité des paramètres et de la solution.

La *FFD* peut générer des formes irréalistes, et une optimisation de forme basée sur cette technique peut aboutir sur une géométrie irréalisable. Finalement, si les déformations sont appliquées directement au maillage, volumique ou surfacique, de l'objet utilisé pour la simulation, alors la *FFD* est limitée par la conservation de la qualité de ce maillage.

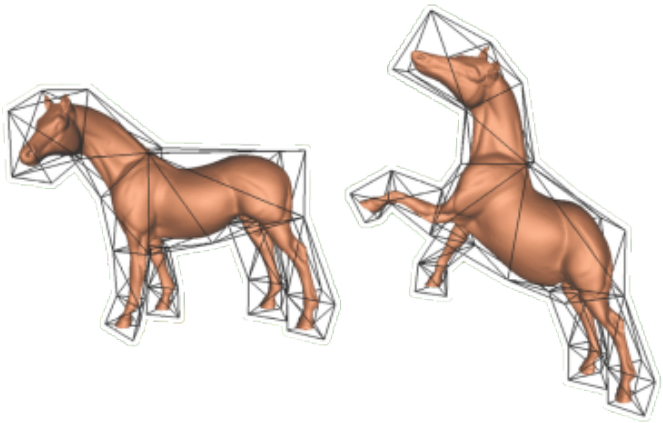


FIGURE 2.2 – Illustration d'une FFD, issue de l'article [Ju et al., 2005]

Laplacian Surface Editing

L'***édition par Laplacien*** consiste à définir les points d'une surface de façon relative, plutôt que d'utiliser la représentation absolue usuelle des coordonnées de l'espace en 3D. Les coordonnées Laplaciennes proposées par [Alexa, 2003] décrivent les coordonnées d'un point d'une surface comme la distance de ce point au barycentre de ses voisins. Les points de la surface peuvent être les sommets d'un maillage ou les points de contrôle d'une surface NURBS.

L'édition par Laplacien est une extension de la méthode de *morphing* ou *Free Form Deformation*, basée sur l'utilisation des coordonnées Laplaciennes. L'édition par Laplacien est une technique dédiée aux déformations de maillages. L'utilisation de ces techniques permet de mieux préserver la forme de l'objet car les détails de la géométrie sont conservés lors des déformations [Alexa, 2003, Sorkine et al., 2004, Zhou et al., 2007]. La figure 2.3 illustre une application de la déformation par FFD réalisée avec des coordonnées Laplaciennes.

Tout comme le *morphing* ou *Free Form Deformation*, l'édition par Laplacien n'est pas spécialement adaptée pour les applications métier ou avec un lien avec la simulation numérique et les limitations présentées sont identiques.
L'édition par Laplacien n'a pas été utilisée pour des applications navales.

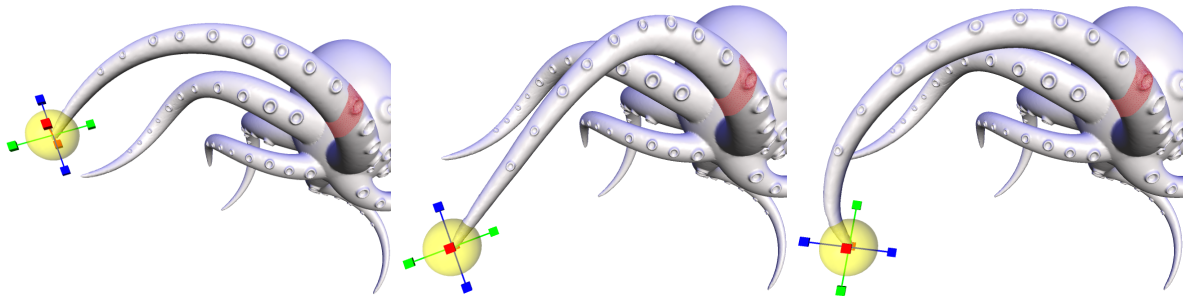


FIGURE 2.3 – Illustration d'une FFD locale basée sur l'utilisation des coordonnées Laplaciennes, issue de l'article [Sorkine et al., 2004]. Les déformations se propagent jusqu'à la zone en rouge puis s'arrêtent.

Axe médian

En animation 3D, le concept de squelette sert à contrôler des objets. Généralement, le squelette est défini comme un ***axe médian*** de l'objet, par exemple au sens de Voronoï pour les surfaces discrétisées comme décrit par [Damon, 2005]. C'est un axe central de l'objet, qui décrit sa forme générale, comme illustré par la figure 2.4.

L'axe médian est déformé, par exemple en utilisant une méthode de *Free Form Deformation*, et l'objet est déformé pour correspondre à ce nouvel axe [Capell et al., 2002, Yoshizawa et al., 2007].

Le concept d'axe médian est intéressant pour contrôler efficacement des objets complexes, avec des embranchements (bras d'un personnage animé par exemple) et des protubérances (les doigts de sa main). Mais pour des formes telles que les coques de bateau cette représentation est trop simpliste car le squelette ne décrit qu'une direction de l'objet et ne permet pas de réaliser des déformations dans les autres.

Par exemple, il est possible d'augmenter la longueur d'un navire avec un axe central, mais changer sa largeur nécessite d'agir sur d'autres paramètres. Les éléments de l'axe médian sont à une certaine distance des bords de la coque. Cette distance peut être modifiée pour changer la largeur.

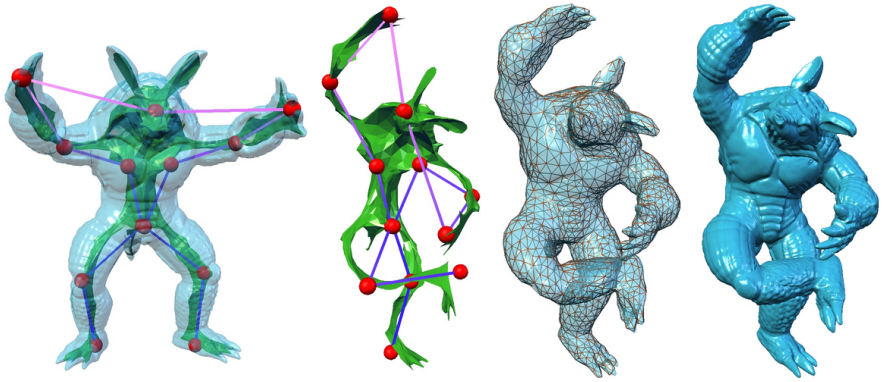


FIGURE 2.4 – Illustration de la création d'un axe médian avec des poignées de contrôle et de sa déformation, issue de l'article [Yoshizawa et al., 2007].

Surfaces de subdivision

Une surface de subdivision est obtenue comme la limite de raffinement d'un maillage initial de contrôle [Peters and Reif, 2008]. Ce maillage initial, ou maillage de contrôle, est subdivisé un certain nombre de fois suivant un algorithme donné et converge vers la surface finale que l'on souhaite obtenir. L'algorithme de subdivision le plus connu est *Catmull-Clark* [Catmull and Clark, 1978].

Ce type de surface est de plus en plus utilisé dans l'animation 3D [DeRose et al., 1998, Warren and Weimer, 2001]. Les déformations directes se font en modifiant directement les points du maillage de contrôle, comme illustré dans la figure 2.5.

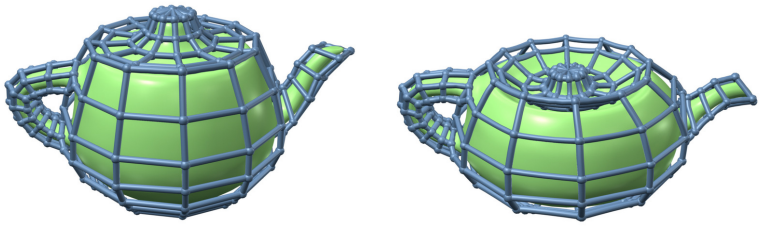


FIGURE 2.5 – Illustration de la déformation d'une surface de subdivision en utilisant son maillage de contrôle, issue de l'article [Zhou et al., 2007]

Des techniques spécifiques ont aussi été développées pour déformer ce type de surfaces, nous en décrivons deux ci-dessous.

[Zhou et al., 2007] propose une technique de déformation basée sur la manipulation directe de points sur la surface de subdivision. Le maillage de contrôle correspondant à la nouvelle surface est calculé à partir des déplacements de la surface finale. Les déformations sont gérées avec des coordonnées Laplaciennes (voir *édition par Laplacien*).

[Lee et al., 2000] introduit une nouvelle notion pour les surfaces de subdivision : les surfaces de subdivision déplacées ou *Displaced Subdivision Surfaces*. Cette technique permet de définir les détails d'une surface par un champ de variations distribué le long d'une représentation lissée de la surface. Les détails des surfaces deviennent donc très facilement maniables : il suffit de modifier la distribution de variations. La surface de base, représentée sous forme lisse, n'a pas besoin d'être modifiée.

Les surfaces de subdivisions ne sont pas spécialement adaptée pour les applications architecturales ou ayant un lien avec la simulation numérique car elles présentent des limitations similaires à celles décrites précédemment. Tout comme les points de contrôle des NURBS, les points du maillage de contrôle n'ont pas de signification métier.

Dans cette thèse, nous abordons les problématiques de reconstruction des morceaux de surfaces paramétriques (voir chapitre 6). L'utilisation des surfaces de subdivision a amené au développement d'algorithmes de "conversion" avec des surfaces NURBS. Le principe de "conversion" consiste en réalité à approcher une surface d'un certain type par une surface de l'autre catégorie.

[Loop and Schaefer, 2008] propose une méthode d'approximation de surfaces de subdivision par des surfaces NURBS en imposant la continuité entre les différents morceaux. Cependant, seule la qualité visuelle de la surface finale est considérée et l'ensemble des surfaces sont G^0 (C^0).

Pour nos applications, il est nécessaire d'obtenir des ensembles de surfaces au moins G^1, car les résultats de la simulation numérique sont impactés par la régularité des surfaces.

De plus, le manque de régularité des courbures des surfaces de subdivision autour de points du maillage de contrôle ayant une valence différente de 4 peut entraîner des difficultés dans l'approximation des modèles géométriques par des surfaces de subdivision.

Conclusion

Ces méthodes de déformation issues de l'animation 3D tiennent uniquement compte de la description géométrique de l'objet, et il est difficile d'inclure des contraintes architecturales pertinentes pour une optimisation de forme (longueurs, angles, rayons de courbures, aires, etc.) comme paramètres de contrôle. Nous constatons que seules les méthodes de *morphing* et *Free Form Deformation* ont été transférées pour des applications navales.

Pour envisager une application de l'une des méthodes présentées, une attention particulière doit être portée au maintien de la continuité entre surfaces adjacentes si le modèle comporte plusieurs morceaux de surfaces paramétriques. De même, si des déformations locales sont envisagées, une jonction régulière doit être préservée entre les parties mobiles et les parties fixes. Enfin, le lissage de la forme doit être conservé. En architecture navale, la courbure de la surface d'une coque doit être régulière, mais il n'existe pas de formulation universelle mathématique pour traduire ce critère [Gravesen and Ungstrup, 2002]. Généralement, le lissage est défini comme la minimisation des variations de la courbure [Eck and Hadenfeld, 1995, Zhang et al., 2001, Gravesen and Ungstrup, 2002, Ceruti et al., 2014].

2.2 Méthodes orientées métier

Physically driven optimisation

Pour des applications avec une interaction directe avec les caractéristiques physiques de l'objet, des méthodes de ***déformations guidées par la physique*** (*physically driven deformations* en anglais) existent [Gibson and Mirtich, 1997, Nealen et al., 2006, Gonzalez-Hidalgo et al., 2007, Schafer et al., 2014]. Ces méthodes sont liées à la fois à la description géométrique de l'objet et à l'application physique réalisée. Nous détaillons ci-dessous plusieurs exemples de

l'utilisation de propriétés ou d'équations de mécanique dans le processus de déformation d'un objet.

[Tang et al., 2014] propose un procédé de déformation de maillages polyédraux. Un système linéaire est construit en regroupant les contraintes physiques et géométriques de la forme finale, avec comme inconnues les coordonnées des points du maillage. Cette formulation est rapide et robuste, mais permet d'utiliser uniquement des contraintes linéaires ou quadratiques. La technique décrite dans [Tang et al., 2014] permet d'assurer un équilibre statique des forces linéaires.

Cette technique s'oppose à la construction de systèmes de minimisation, qui permettent d'inclure des contraintes non-linéaires par exemple.

[Deng et al., 2015] propose une application similaire à [Tang et al., 2014] sur les maillages polyédraux en utilisant un système de minimisation pour inclure les contraintes physiques dans la déformation du maillage.

[Skouras et al., 2012] utilise les propriétés mécaniques des matériaux composants les objets à déformer, représentés sous forme de maillages. Un système de minimisation est résolu pour réaliser la déformation : il s'agit de déterminer la position des points du maillage pour équilibrer le bilan des forces appliquées sur l'objet. Les caractéristiques physiques du matériel composant l'objet sont prises en compte lors de l'application de forces, permettant de modéliser les comportements non linéaires.

[Allaire et al., 2002] propose une méthode de déformation topologique. Les applications de cette méthode portent sur des problèmes structurels, en 2D ou 3D [De Gournay, 2005]. Les formes sont représentées par des "lignes de niveau" associées à des fonctions appelées *level-set*, sur un maillage fixe donné. L'objet est déformé suivant la direction du gradient calculé sur le domaine de calcul et sur ses bords. Ces déplacements sont modélisés par une équation de transport. Une formulation du gradient topologique est proposée par [Eschenauer et al., 1994]. L'évolution des valeurs de ce gradient lors du transport du domaine permet de créer de nouveaux "trous" ou d'en faire disparaître, et donc de modifier la topologie de l'objet.

Les travaux [Aguilar, 1996] et [Guido, 1997] portent sur l'optimisation de forme liée à la résolution des équations de Navier-Stokes. Une énergie, fonction de la pression exercée, est calculée le long des surfaces où l'écoulement est modélisé. La forme est modifiée en fonction de la répartition de l'énergie : les zones présentant une valeur élevée de l'énergie doivent être réduites.

Les *déformations guidées par la physique* sont des méthodes efficaces, mais extrêmement dépendantes du solveur utilisé et du critère d'optimisation choisi.

Logiciels de CAO paramétriques généralistes

Des outils de ***CAO paramétriques*** dédiés à des applications industrielles comme *Catia*^{TM} ou *Grasshopper* pour Rhinoceros 3D intègrent des fonctionnalités de design paramétrique, permettant à l'utilisateur de paramétriser le modèle dès le début de la conception. Quand ces paramètres de conception sont modifiés, les éléments qui en dépendent sont modifiés. Grâce au plan de construction liant hiérarchiquement les éléments de l'objet, tous les éléments dépendant indirectement du paramètre sont aussi modifiés et propagés dans tout le modèle.

[Kostas et al., 2015] montre une application à l'optimisation automatique de forme d'une

coque de bateau entièrement paramétrée dans CATIA, [Guha and Falzarano, 2015] montre l'utilisation de *Grasshopper* pour le même type d'application.

Ces logiciels permettent de modifier des formes rapidement et facilement, mais tous les paramètres de design introduits lors de la conception ne sont pas conservés par les formats d'échanges standard de géométrie NURBS (IGES ou STEP) [Mun et al., 2003]. C'est une limitation pour l'intégration de tels modèles dans une boucle d'optimisation automatique de forme. De plus, le modèle paramétrique peut ne pas être directement compatible avec les solveurs numériques.

[Guha and Falzarano, 2015] utilisent un solveur basé sur une discrétisation en panneaux, et ont développé un mailleur adapté à la représentation des coques du modèle *Grasshopper*. [Kostas et al., 2015] utilise un solveur basé sur *l'analyse isogéométrique* pour réaliser les calculs. L'analyse isogéométrique est une méthode de calcul numérique basée sur l'utilisation des B-Splines dans la méthode des éléments finis. Le modèle initial est dessiné spécialement pour être utilisé par le solveur isogéométrique, c'est-à-dire qu'il est simplifié et nettoyé le plus possible. Dans cette thèse, nous n'utilisons pas de solveur isogéométrique.

Logiciels de CAO spécialisés

Les méthodes décrites précédemment sont relativement génériques, c'est-à-dire qu'elles peuvent s'appliquer à des domaines variés (aéronautique, automobile, naval, architecture, etc.).

Nous présentons ci-dessous trois méthodes conçues pour des applications spécifiques, deux pour les coques de bateau (Bataos et CAESES) et une pour la conception de profils portants (PARSEC), forme très utilisée dans les appendices de bateau (safrans, foils, hélices, mâts, etc.).

PARSEC [Sobieczky, 1999] est un outil permettant de générer des profils à partir d'un ensemble de 11 paramètres de forme pré-déterminés. Les courbes d'intrados et d'extrados des profils sont définies comme des polynômes, dont les coefficients sont solutions d'un système linéaire construit à partir des paramètres de forme. La forme est donc entièrement déterminée par ces paramètres. Des applications d'optimisation automatique de forme avec PARSEC ont été réalisée par exemple dans [Jeong et al., 2005] et [Mukesh et al., 2014]. PARSEC est appliqué uniquement aux profils. Les profils type NACA sont naturellement décrits par des polynômes, d'où l'extension de cette formulation pour des profils génériques réalisée par PARSEC. Identifier les coefficients d'un ou de polynômes décrivant des formes plus complexes, comme une coque de bateau, est beaucoup plus difficile et c'est d'ailleurs la raison pour laquelle les NURBS sont utilisées.

BATAOS [Jacquin et al., 2004, Huetz and Guillerm, 2014] est un logiciel développé spécialement pour déformer les coques de bateau. Il se base sur une bibliothèque de fonctions de déformation. Les points de contrôle des courbes de clés du navire, décrites sous une forme de Bézier, sont additionnés ou multipliés par ces fonctions de déformation pour obtenir une nouvelle forme. Les courbes clé sont les courbes de sections, la ligne de quille, les contours, etc. identifiées par l'utilisateur. Ensuite, une surface est reconstruite sur les courbes déformées et le maillage structuré est déformé pour correspondre à la nouvelle géométrie. *BATAOS* est capable de gérer des déformations locales des objets en définissant une "zone tampon" entre les parties fixes et les parties déformées. Cette zone interpole les déformations jusqu'à la valeur nulle (c'est-à-dire addition avec 0 ou multiplication par 1), pour réaliser une transition régulière

entre les différentes parties du modèle.

La figure 2.6 illustre l'utilisation de Bataos pour modifier la forme d'un bulbe. Le bulbe est découpé suivant le plan de symétrie en Y et deux courbes de Bézier sont utilisées pour représenter les contours du bulbe. Dans la première image (à gauche), les points de contrôle de la courbe subissent une translation en Z et dans la deuxième image (à droite), ils subissent une translation en X.

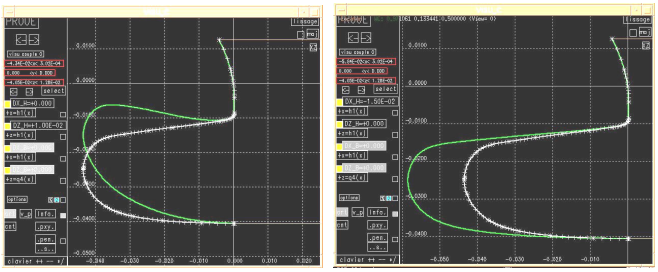


FIGURE 2.6 – Illustration de la déformation d'un bulbe avec Bataos, issue de l'article [Jacquin et al., 2004]

La méthodologie de *BATAOS* est efficace et relativement simple. La limitation principale est la bibliothèque de fonctions de déformation pré-déterminées. Pour chaque application, l'utilisateur va devoir créer de nouvelles fonctions pour s'adapter à son cas. De plus, la description par courbes de Bézier peut amener à définir beaucoup de points de contrôle, impactant la qualité du lissage de la forme finale. En effet, avec de nombreux de points de contrôle, les faibles oscillations dans le réseau de points influencent fortement la régularité de la courbe.

Le logiciel ***CAESES*** de Friendship Systems™ présente deux grandes fonctionnalités, qui définissent deux modes d'utilisation.

Le premier, "Modèle entièrement paramétrique", consiste à utiliser CAESES comme un logiciel de CAO paramétrique (cf. description ci-dessus). La particularité de *CAESES* est l'application dédiée aux coques, c'est-à-dire qu'il est possible de créer des courbes de sections paramétriques, comme dans un plan de forme d'architecture navale, puis de construire des surfaces qui passent par ces sections. Les mesures utilisées dans la conception de navire peuvent être calculées sur les géométries créées : la courbe d'aire, les données hydrostatiques, etc. Les variations de formes sont répercutées sur la valeur de ces mesures.

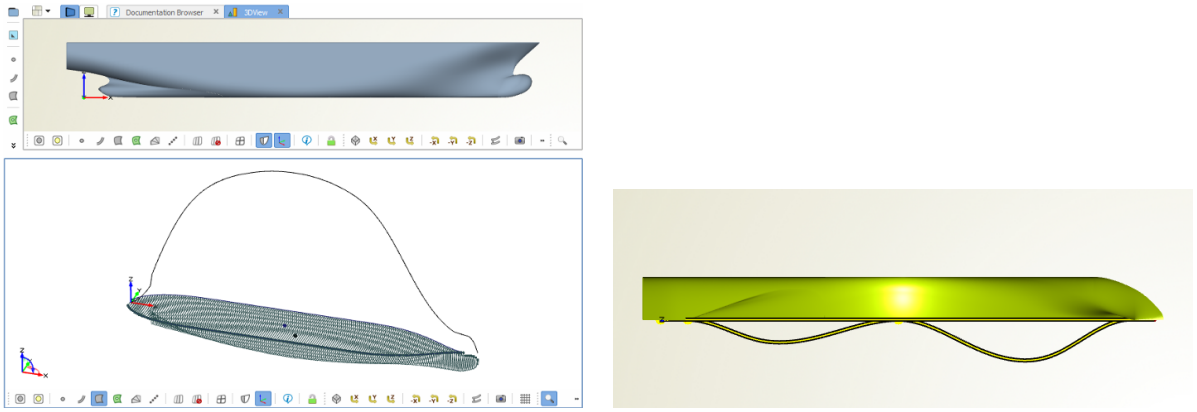
Le deuxième, "Modèle partiellement paramétrique", permet d'importer une géométrie en format standard (IGES, STEP) et de paramétriser *a posteriori* le modèle. L'utilisateur a la possibilité de créer des courbes ou des surfaces de modification de deux types.

Les éléments les plus simples sont les *delta shift transformations* : les courbes et surfaces *delta shift* sont définies avec des B-Splines, dont l'utilisateur modifie la position des points de contrôle en effectuant des translations dans une direction donnée. Les points de contrôle des courbes ou des surfaces composant le bateau suivent ces déformations, amenant à la modification de la forme de la coque. Ce type de modification s'apparente à la FFD, mais en utilisant la définition par courbes de sections disponible dans *CAESES*, les déformations obtenues sont plus réalistes.

La transformation par *Generalized Lackenby Shift* est plus complexe. Elle permet de modifier la courbe d'aire du bateau, et d'obtenir la déformation de la coque correspondante. Des courbes de *delta shift* sont créées, et les positions de leurs points de contrôle sont optimisées pour obtenir

la variation de forme souhaitée par l'utilisateur. Cette transformation est réalisée en résolvant un système de minimisation, où la position des points de contrôle du *delta shift* sont les paramètres et la différence entre la courbe d'aire cible et la courbe d'aire de la coque modifiée est la fonction objectif.

La figure 2.7(a) illustre la courbe d'aire calculée sur un coque de bateau, et la figure 2.7(b) montre la courbe de *delta shift* obtenue pour une déformation donnée de la courbe d'aire.



(a) Courbe d'aire d'un coque de bateau

(b) Courbe *delta shift* correspondant à une déformation de la courbe d'aire

FIGURE 2.7 – Illustration d'une transformation par *Generalized Lackenby Shift* dans *CAESES*, disponible sur le site web de Friendship Systems™ (`https://www.caeses.com/blog/2016/ship-hull-optimization-generalized-lackenby/`)

Des exemples d'utilisation de *CAESES* pour l'optimisation de bateaux sont décrits dans [Maisonneuve et al., 2003, Couser et al., 2011, Papanikolaou et al., 2011, MacPherson et al., 2016].

Conclusion

Les deux catégories de méthodes existantes présentent chacune des avantages.

Les méthodes géométriques sont généralistes et nécessitent peu d'interactions avec l'utilisateur pour être mises en place. En revanche, la qualité et la vraisemblance des formes générées ne sont pas assurées.

Les méthodes orientées métier demandent plus d'investissement de la part de l'utilisateur, et sont spécifiques à chaque application étudiée. En revanche, les formes générées sont systématiquement valides.

Notre but est de proposer un modeleur paramétrique tirant avantage des deux types de méthode, et ainsi être en mesure de déformer des objets avec peu de contraintes pour l'utilisateur tout en conservant leur validité géométrique et architecturale. Le modeleur doit être compatible avec :

- les logiciels de simulation numérique, d'où l'importance de la validité géométrique,
- un algorithme d'optimisation, d'où l'importance de la validité architecturale.

Chapitre 3

Approximation par courbes et surfaces B-Splines

3.1 Notions sur les courbes et surfaces B-Splines

Nous rappelons ici les définitions classiques des courbes et surfaces B-Splines. Pour plus de détails sur les propriétés des B-Splines et de leurs fonctions de base, il est recommandé de se référer à [Piegl and Tiller, 1997].

Définition 3.1.1. ***Fonctions de bases B-Spline***
Considérons une suite de $m+1$ nœuds t_i dans tels que $t_0 \leq t_1 \leq ... \leq t_m$. S'il y a r nœuds t_i égaux à τ, alors on dit que τ est un nœud de multiplicité r.
Les fonctions de base B-Spline de degré p, notées $B_{i,p}(t)$, sont définies de façon récursive par :

$$B_{i,0}(t) = \begin{cases} 1 & si\ t_i \leq t < t_{i+1} \\ 0 & sinon \end{cases}$$

$$B_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} B_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} B_{i+1,p-1}(t)$$

pour $i = 0, ..., m - k - 1$.

Les fonctions de bases B-Spline constituent une base de l'espace des fonctions polynomiales par morceau de degré inférieur ou égal à p sur la subdivision $[t_i, t_{i+1}[$.

Définition 3.1.2. ***Courbe B-Spline***
Soit $d+1$ points de l'espace $c_0, c_1, ..., c_d$. Soit un vecteur de $m+1$ nœuds t_i dans $[0,1]$ tels que $t_0 \leq t_1 \leq ... \leq t_m$. Alors la courbe B-Spline de degré p tel que $m = n + p + 1$ est définie par :

$$\sigma(t) = \sum_{i=0}^{d} c_i B_{i,p}(t),\ t \in [0,1]$$

Il existe beaucoup d'algorithmes numériques qui permettent de manipuler les B-Splines en modifiant leurs points de contrôle. Par exemple l'élévation de degré, l'insertion de nœuds, l'insertion de point de contrôle, la dérivée, l'évaluation, etc. Ces algorithmes sont stables et efficaces, ils sont décrit dans [Piegl and Tiller, 1997].

La figure 3.1 montre une courbe B-Spline de degré $p = 2$ avec 7 points de contrôle et le vecteur de nœuds associé $[0, 0, 0, 0.25, 0.5, 0.75, 0.75, 1, 1, 1]$. La multiplicité $p + 1$ des nœuds aux extrémités permet à la courbe de passer exactement par le premier et le dernier point de contrôle. Les éléments du vecteur de nœud n'étant pas équidistants, la courbe est dite non-uniforme. La figure 3.2 montre les fonctions de base associées au vecteur de nœuds.

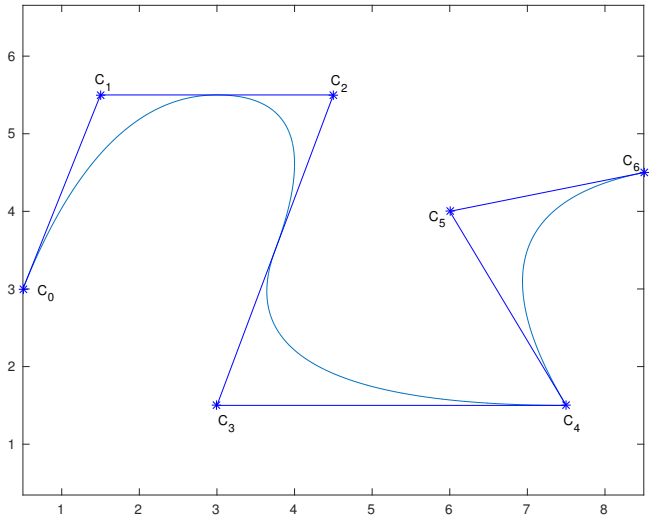


FIGURE 3.1 – Courbe B-Spline non-uniforme de degré 2 avec 7 points de contrôle

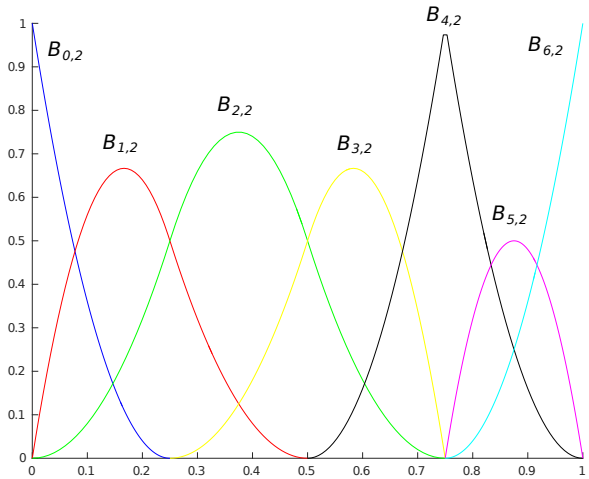


FIGURE 3.2 – Fonction de base pour le vecteur de nœuds $0, 0, 0, 0.25, 0.5, 0.75, 0.75, 1, 1, 1$

Définition 3.1.3. *Considérons un réseau de points de contrôle $c_{i,j}$ et deux vecteurs de nœuds $u_0, ..., u_{m_1}$ et $v_0, ..., v_{m_2}$. Alors une surface B-Spline de degré (p, l) est définie comme :*

$$S(u, v) = \sum_{i=0}^{d_1} \sum_{j=0}^{d_2} c_{i,j}\, B_{i,p}(u)\, B_{j,l}(v), \;\; (u, v) \in [0, 1] \times [0, 1]$$

où les $B_{i,p}(u)$, $B_{j,l}(v)$ sont les fonctions de base B-Spline définies respectivement sur les vecteurs de nœuds u_i et v_i

La figure 3.3 représente le réseau de points de contrôle d'une surface et la surface B-Spline correspondante.

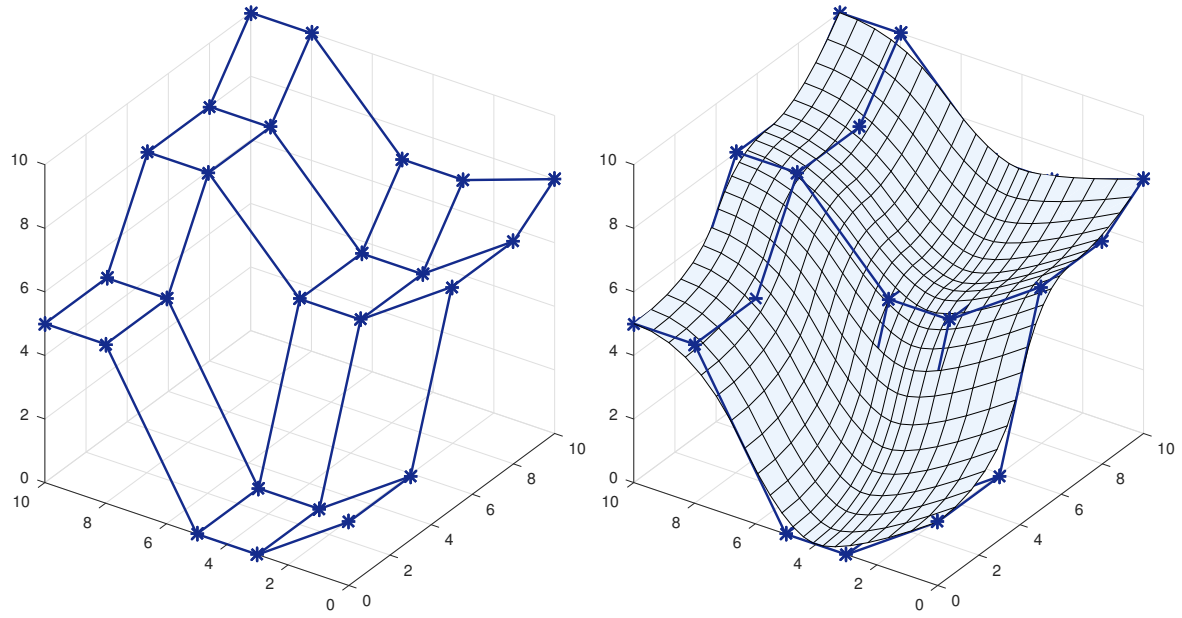


FIGURE 3.3 – Surface B-Spline de degré 2 et son réseau de points de contrôle

3.2 Approximation par courbe B-Spline

L'approximation par courbe B-Spline, ou *fitting* en anglais, consiste à construire une courbe qui passe un ensemble de points donnés [Piegl and Tiller, 1997]. Si la courbe passe exactement par les points donnés on parle *d'interpolation*, si la courbe approche les points on parle *d'approximation*.

Considérons l'ensemble des *points cibles* $P = \{P_i : i = 0, ..., n\}$ comme les points que nous cherchons à représenter avec une courbe B-Spline $\sigma(t)$.

Comme pour les méthodes d'interpolation classiques par des polynômes, l'interpolation par courbe B-Spline produit des oscillations pour des degrés élevés et ne permet pas de reproduire des géométries complexes. Nous traitons donc ici uniquement des problèmes d'approximation.

Les figures 3.4 et 3.5 montrent les courbes obtenues respectivement avec une méthode d'interpolation et une méthode d'approximation pour un même ensemble de *points cibles* $P = \{P_i : i = 0, ..., n\}$, pour $n = 7$ à gauche et pour $n = 11$ à droite. Les figures 3.4 mettent en évidence le phénomène oscillatoire de l'interpolation lorsque le nombre de points à interpoler est trop important. Les figures 3.5 montrent la courbe d'approximation et l'erreur relative entre la courbe et les points. Quelque soit le nombre de points à approximer, la courbe ne produit pas d'oscillations.

L'algorithme d'approximation par courbe B-Spline va permettre de déterminer la position optimale des points de contrôle de la courbe $\sigma(t)$ afin qu'elle approche au mieux l'ensemble des points cibles P.

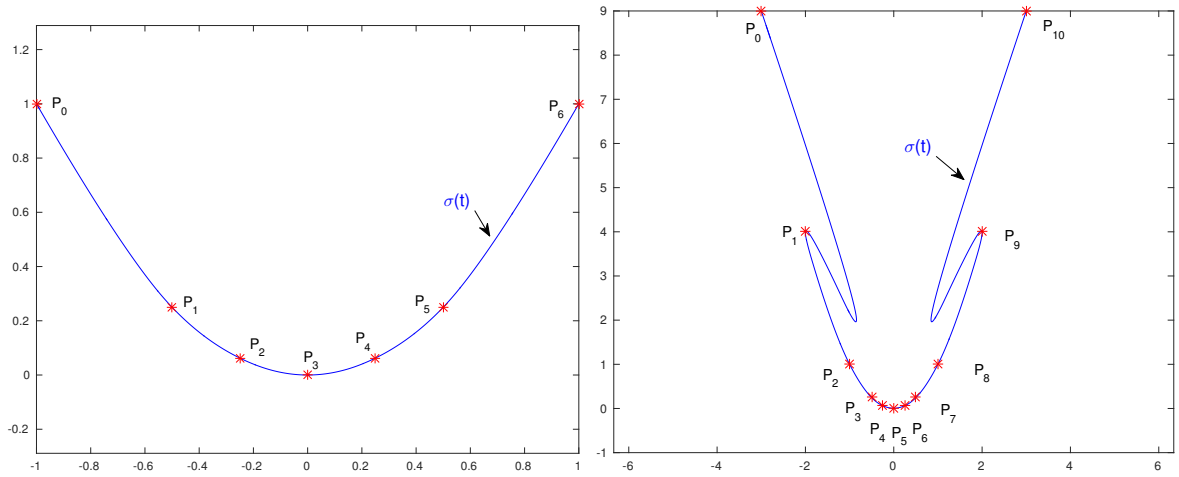


FIGURE 3.4 – Courbes d'interpolation

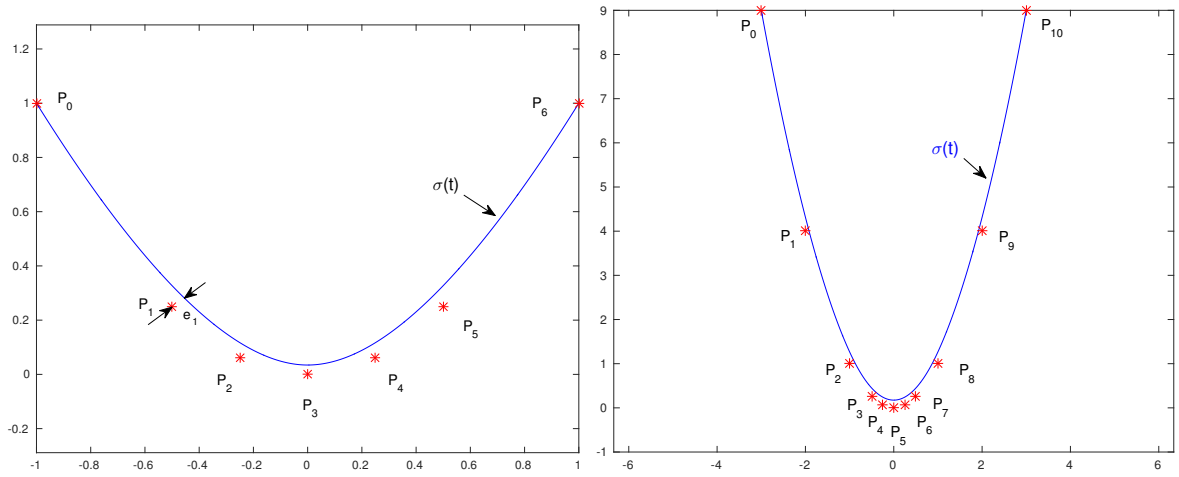


FIGURE 3.5 – Courbes d'approximation

Le degré et le vecteur de nœuds de la courbe sont généralement fixés à priori. Le degré p de σ étant fixé, $B_{j,p}(t)$ est noté $B_j(t)$.

Définition 3.2.1. *Pour approximer un ensemble de points cibles P par une courbe B-spline $\sigma(t)$, nous cherchons la paramétrisation $\sigma(t)$ qui minimise la fonctionnelle suivante :*

$$\begin{aligned} J(\mathbf{c}) &= \sum_{i=0}^{n} d(P_i, \sigma(t_i))^2 + \lambda f_s \\ \text{avec } d(P_i, \sigma(t_i)) &= \min_{t\in[0,1]} \|\sigma(t) - P_i\|, \ i = 0, ..., n \end{aligned} \tag{3.2.1}$$

où les inconnues $\mathbf{c}$ sont les coordonnées des points de contrôle de $\sigma(t)$, $d(P_i, \sigma(t_i))$ représente la distance du point P_i à la courbe σ, f_s est un terme correctif pour assurer le lissage de la courbe et λ une constante positive permettant de pondérer l'influence de f_s.

En général, il est souhaitable que la courbe passe exactement par le premier et le dernier point de P_l. Pour s'assurer que la courbe σ est fixée à P_0 et P_n à ses extrémités nous imposons

$c_0 = P_0$ et $c_d = P_n$, où c_k désigne le $k^{ième}$ point de contrôle de la courbe.

Les paramètres t_i de σ font correspondre un point de la courbe à chacun des P_i. La distance courbe/points sera évaluée en chacun des couples $(P_i, \sigma(t_i))$. Les différentes méthodes de calcul pour obtenir les t_i sont exposées dans la section 3.2.1. Ce calcul appelé *paramétrage* constitue la première étape de l'algorithme d'approximation par courbe B-Spline.

Ensuite, trois approches possibles pour la définition de la distance $d(P, \sigma(t))$ sont utilisées et explicitées dans les sections 3.2.2, 3.2.3 et 3.2.4 :

1. la distance euclidienne, ou Point Distance Minimization (PDM)
2. la distance tangente, ou Tangent Distance Minimization (TDM)
3. la distance quadratique, ou Squared Distance Minimization (SDM)

Enfin, la section 3.2.5 détaille le terme correctif f_s.

3.2.1 Paramétrage

Il s'agit de déterminer les paramètres t_i de façon à ce que $\sigma(t_i)$ soit le plus proche possible de P_i. Nous présentons deux approches pour calculer les valeurs des t_i.

La première approche possible consiste à calculer les t_i indépendamment de la courbe. L'article [Haron et al., 2012] compare les courbes obtenues avec sept différents types de paramétrage sur six différents ensembles de points cibles différents. La méthode la plus simple consiste en une répartition uniforme des t_i [De Boor, 2001]. D'autres méthodes comme chord length [Lü, 2009] ou centripetal [Lee, 1989] prennent en compte la répartition spatial des P_i pour calculer les t_i. Par exemple la méthode *Chord Length* calcule les t_i par :

$$t_0 = 0,\ t_n = 1$$
$$t_k = \frac{\sum_{i=1}^{k} |P_i - P_{i-1}|}{L},\ L = \sum_{i=1}^{n} |P_i - P_{i-1}|$$

Ces méthodes sont rapides mais l'erreur produite $\|P_i - \sigma(t_i)\|$ influe sur la précision d'approximation de la courbe. De plus, chaque paramétrisation produit une allure de courbe différente comme le montre [Haron et al., 2012]. Il est difficile de déterminer si le résultat final est satisfaisant ou non.

La deuxième approche pour calculer les paramètres t_i consiste à projeter orthogonalement les P_i sur la courbe σ. Les valeurs des paramètres de la courbe aux points obtenus sont les t_i. La figure 3.6 illustre cette méthode appelée *Foot point computation* dans [Wang et al., 2006].

Cette technique de calcul nécessite une courbe initiale. En général, elle est utilisée sur une courbe générée grâce à une des techniques de paramétrisation décrites précédemment. Puis, par une méthode itérative les paramètres sont ajustés avec de plus en plus de précision.

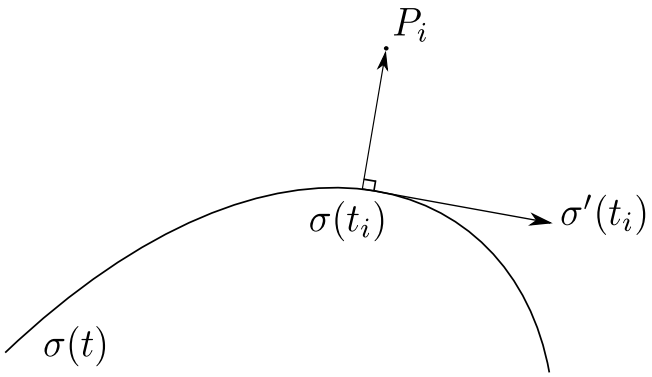


FIGURE 3.6 – Foot Point

3.2.2 Méthode PDM

La courbe B-spline $\sigma(t)$ qui approxime les points cibles P est définie par l'équation (3.2.1) comme minimisant la distance $\sum_{i=0}^{n} d(P_i, \sigma(t_i))^2$.

Il existe plusieurs définitions de la distance $d(P, \sigma(t))$. La méthode la plus simple consiste à utiliser $\|P_i - \sigma(t_i)\|^2$, en assignant à chaque P_i une valeur pré-calculée t_i. Cette méthode appelée *Point Distance Minimization* ou *PDM* a été introduite par [Hoschek, 1988].

Définition 3.2.2. *L'erreur d'approximation au sens PDM est définie par :*

$$J_{PDM}(\mathbf{c}) = \frac{1}{2} \sum_{i=0}^{n} \|P_i - \sigma(t_i)\|^2 \tag{3.2.2}$$

où les inconnues $\mathbf{c}$ *sont les coordonnées des points de contrôle de* $\sigma(t)$.

Proposition 3.2.1. *Les coordonnées des points de contrôle de* $\sigma(t) = \sum_{j=0}^{d} c_j B_j(t)$, *sont solution du système linéaire suivant :*

$$\begin{aligned} AC &= b \text{ avec} \\ A_{lk} &= \sum_{i=0}^{n} B_l(t_i)\, B_k(t_i),\ 1 \le l, k \le d-1 \\ b_k &= \sum_{i=0}^{n} R_i\, B_k(t_i),\ 1 \le k \le d-1 \\ C_k &= c_k,\ 1 \le k \le d-1 \end{aligned} \tag{3.2.3}$$

avec $c_0 = P_0$ *et* $c_d = P_n$ *et* $R_i = P_i - P_0 B_0(t_i) - P_d B_d(t_i)$.

Les trois coordonnées (x, y, z) *des points de contrôles* $\mathbf{c}$ *sont découplées dans la système d'équations* (3.2.3). *Cela nous permet de construire trois matrices indépendantes pour chacune des coordonnées :* A_x, b_x, C_x ; A_y, b_y, C_y ; *et* A_z, b_z, C_z.
Les matrices $A_{x,y,z}$ *sont de taille* $(d-2)\times(d-2)$, *les vecteurs* $b_{x,y,z}$ *et* $C_{x,y,z}$ *sont de taille* $(d-2)$.

Démonstration. L'équation (3.2.2) peut s'écrire de la façon suivante :

$$\begin{aligned}
J_{PDM}(c) &= \frac{1}{2}\sum_{i=0}^{n} \|P_i - \sigma(t_i)\|^2 \\
&= \frac{1}{2}\sum_{i=0}^{n} \|P_i - \sum_{j=0}^{d} c_j B_j(t_i)\|^2 \\
&= \frac{1}{2}\sum_{i=0}^{n} \|P_i - c_0 B_0(t_i) - c_d B_d(t_i) - \sum_{j=1}^{d-1} c_j B_j(t_i)\|^2 \\
&= \frac{1}{2}\sum_{i=0}^{n} \|R_i - \sum_{j=1}^{d-1} c_j B_j(t_i)\|^2 \text{ avec } \quad R_i = P_i - c_0 B_0(t_i) - c_d B_d(t_i)
\end{aligned}$$

Rappelons que $c_0 = P_0$ et $c_d = P_n$, donc $R_i = P_i - P_0 B_0(t_i) - P_d B_d(t_i)$.

Les t_i étant fixés, J_{PDM} est une fonction quadratique et son minimum est déterminé par :

$$\frac{\partial J_{PDM}(c)}{\partial c} = 0$$

Avec la réécriture proposée, on obtient :

$$\begin{aligned}
&\frac{\partial J_{PDM}(c)}{\partial c} = 0 \\
\Leftrightarrow &\sum_{i=0}^{n}(R_i - \sum_{j=1}^{d-1} c_j B_j(t_i)) * B_k(t_i) = 0 \\
\Leftrightarrow &\sum_{i=0}^{n}\sum_{j=1}^{d-1} c_j (B_j(t_i) * B_k(t_i)) = \sum_{i=0}^{n} R_i * B_k(t_i) \\
\Leftrightarrow &\; AC = b
\end{aligned}$$

avec

$$\begin{aligned}
A_{jk} &= \sum_{i=0}^{n} B_j(t_i)\, B_k(t_i),\ 1 \le j,k \le d-1 \\
b_k &= \sum_{i=0}^{n} R_i\, B_k(t_i),\ 1 \le k \le d-1 \\
C_k &= c_k,\ 1 \le k \le d-1
\end{aligned}$$

□

La méthode PDM donne une définition simple de la distance $d(P, \sigma(t))$, en remplaçant $\sigma(t)$ par $\sigma(t_i)$. En calculant les t_i par Foot Point (3.2.1), il est possible d'itérer en recommençant plusieurs fois l'algorithme PDM avec une mise à jour des t_i à chaque itération.

$$\sigma_{k,PDM}(t) = \min_c J_{PDM} = \min_c \frac{1}{2}\sum_{i=0}^{n} \|P_i - \sigma(t_{i,k-1})\|^2$$

[Wang et al., 2006] montre que cette technique est proche d'un algorithme de descente par gradient et donc converge linéairement.

3.2.3 Méthode TDM

Pour améliorer la convergence de la méthode PDM, l'approche *Tangent Distance Minimization* ou *TDM* a été introduite par [Blake and Isard, 1998]. Cette méthode consiste à définir la distance $d(P_i, \sigma(t_i))$ comme la distance entre P_i et la projection normale de P_i sur la tangente de $\sigma(t)$ à t_i. [Wang et al., 2006] montre que la méthode TDM est équivalente à un algorithme de Gauss-Newton.

Définition 3.2.3. *L'erreur d'approximation au sens TDM est définie par :*

$$J_{TDM}(\mathbf{c}) = \frac{1}{2}\sum_{i=0}^{n}[(P_i - \sigma(t_i))^T N_i]^2 \tag{3.2.4}$$

où les inconnues $\mathbf{c}$ *sont les coordonnées des points de contrôle de* $\sigma(t)$*,* N_i *est la normale unitaire à* σ *au point de paramètre* t_i *et* ^{T} *désigne la transposée.*

La méthode TDM se base sur une courbe initiale $\sigma_0(t)$, généralement calculée avec la méthode PDM. TDM est une méthode itérative, où chaque étape k se base sur la courbe $\sigma_{k-1}(t)$ et les paramètres t_i peuvent être recalculés.

$$\sigma_{k,TDM}(t) = \min_c J_{TDM} = \min_c \frac{1}{2}\sum_{i=0}^{n}[(P_i - \sigma_{k-1}(t_{i,k-1}))^T N_i]^2$$

Un repère de Frenet est associé à la courbe en tout points :

$$\begin{array}{lrl} \text{La tangente au point de paramètre } t_i \ : & T_i & = \frac{d\sigma(t_i)}{dt} \\ \text{La binormale au point de paramètre } t_i \ : & B_i & = T \wedge \frac{d^2\sigma(t_i)}{dt^2} \\ \text{La normale au point de paramètre } t_i \ : & N_i & = T_i \wedge B_i \end{array} \tag{3.2.5}$$

Proposition 3.2.2. *Les coordonnées des points de contrôle de* $\sigma(t) = \sum_{j=0}^{d} c_j B_j(t)$*, sont solution du système linéaire suivant :*

$$\begin{aligned} &AC = b \text{ avec} \\ &A_{lk} = \sum_{i=0}^{n} B_l(t_i)\, B_k(t_i)\, N_i \otimes N_i^T,\ 1 \le l, k \le d-1 \\ &b_k = \sum_{i=0}^{n} \langle R_i, N_i \rangle B_k(t_i)\, N_i,\ 1 \le k \le d-1 \\ &C_k = c_k,\ 1 \le k \le d-1 \end{aligned} \tag{3.2.6}$$

avec $c_0 = P_0$ *et* $c_d = P_n$*,* $R_i = P_i - P_0 B_0(t_i) - P_d B_d(t_i)$
et où $\langle \cdot, \cdot \rangle$ *désigne le produit scalaire et* $\otimes$ *désigne le produit tensoriel.*

La matrice A *est de taille* $3\,(d-2) \times 3\,(d-2)$*, les vecteurs* b *et* C *sont de taille* $3\,(d-2)$*. La notation* A_{lk} *désigne le bloc* 3×3 *indexé aux indices* l*,* k *dans la matrice* A*. La notation* b_k *(respectivement* C_k*) désigne le bloc* 3×1 *indexé à l'indice* k *dans le vecteur* b *(respectivement* C*).*

Démonstration. L'équation (3.2.4) peut s'écrire de la façon suivante :

$$\begin{aligned} J_{TDM}(c) &= \frac{1}{2}\sum_{i=0}^{n}[(P_i - \sigma(t_i))^T N_i]^2 \\ &= \frac{1}{2}\sum_{i=0}^{n}\langle P_i - \sum_{j=0}^{d} c_j B_j(t_i), N_i\rangle^2 \end{aligned}$$

Les t_i étant fixés, J_{TDM} est une fonction quadratique et son minimum, est déterminé par :

$$\frac{\partial J_{TDM}(c)}{\partial c} = 0$$

Avec la réécriture proposée, on obtient :

$$\begin{aligned} &\frac{\partial J_{TDM}(c)}{\partial c} = 0 \\ \Leftrightarrow &\sum_{i=0}^{n}\langle (P_i - \sigma(t_i), N_i\rangle \, (B_k(t_i)N_i) = 0 \\ \Leftrightarrow &\sum_{i=0}^{n}\langle P_i, N_i\rangle \, (B_k(t_i)N_i) - \langle \sum_{j=0}^{d} c_j B_j(t_i), N_i\rangle \, (B_k(t_i)N_i) = 0 \\ \Leftrightarrow &\sum_{i=0}^{n}\sum_{j=1}^{d-1} B_j(t_i)\langle c_j, N_i\rangle \, (B_k(t_i)N_i) = \sum_{i=0}^{n}\langle R_i, N_i\rangle \, (B_k(t_i)N_i) \\ &\text{avec } R_i = P_i - c_0 B_0(t_i) - c_d B_d(t_i) \\ \Leftrightarrow &\sum_{i=0}^{n}\sum_{j=1}^{d-1} B_j(t_i)B_k(t_i)c_j \, N_i \otimes N_i^T = \sum_{i=0}^{n}\langle R_i, N_i\rangle \, (B_k(t_i)N_i) \\ &\text{car } \langle c_j, N_i\rangle \, N_i = c_j \, N_i \otimes N_i^T \end{aligned}$$

On obtient donc un système linaire $AC = b$ avec :

$$\begin{aligned} A_{jk} &= \sum_{i=0}^{n} B_j(t_i) \, B_k(t_i) \, N_i \otimes N_i^T, \; 1 \leq l, k \leq d-1 \\ b_k &= \sum_{i=0}^{n}\langle R_i, N_i\rangle B_k(t_i) \, N_i, \; 1 \leq k \leq d-1 \\ C_k &= c_k, \; 1 \leq k \leq d-1 \end{aligned}$$

Pour s'assurer que la courbe σ est fixée à P_0 et P_n aux extrémités, nous imposons $c_0 = P_0$ et $c_d = P_n$ donc $R_i = P_i - P_0 B_0(t_i) - P_d B_d(t_i)$.

□

3.2.4 Méthode SDM

La méthode TDM ne donne pas une bonne estimation de l'erreur proche de zones à grande courbure. Pour surmonter cette limitation, [Wang et al., 2006] propose une méthode appelée *Squared Distance Minimization* ou *SDM*. [Wang et al., 2006] montre que la méthode SDM est équivalente à un algorithme de Quasi-Newton.

Définition 3.2.4. *L'erreur d'approximation au sens SDM est définie par :*

$$J_{SDM}(\mathbf{c}) = \begin{cases} \frac{1}{2}\sum_{i=0}^{n} \frac{d_i}{d_i-\rho_i}[(P_i - \sigma(t_i))^T T_i]^2 + [(P_i - \sigma(t_i))^T N_i]^2, & si\ d \geq \rho \\ J_{TDM}(\mathbf{c}), & si\ d < \rho \end{cases} \tag{3.2.7}$$

où les inconnues $\mathbf{c}$ *sont les coordonnées des points de contrôle de* $\sigma(t)$*,* N_i *est la normale unitaire à* σ *au point de paramètre* t_i*,* T_i *est la normale unitaire à* σ *au point de paramètre* t_i*,* d *représente la plus petite distance entre* P_i *et* $\sigma(t_i)$*,* $\rho = \frac{1}{\kappa}$ *représente l'inverse de la courbure* κ *et* ^{T} *désigne la transposée.*

Les vecteurs N_i et T_i sont calculés par un repère de Frenet décrit par l'équation (3.2.5). d et ρ sont calculés comme :

$$\begin{aligned} d &= \|P_i - \sigma(t_i))\| \\ \rho &= \frac{1}{\kappa} = \frac{\|\sigma'(t_i)\|^3}{det(\sigma'(t_i), \sigma''(t_i))} \end{aligned}$$

Comme pour la méthode TDM, SDM se base sur une courbe initiale $\sigma_0(t)$, généralement calculée avec la méthode PDM. SDM est aussi une méthode itérative, où chaque étape k se base sur la courbe $\sigma_{k-1}(t)$ et les paramètres t_i peuvent être recalculés.

Proposition 3.2.3. *Les coordonnées des points de contrôle de* $\sigma(t) = \sum_{j=0}^{d} c_j B_j(t)$*, sont solution du système linéaire suivant :*

$$\begin{aligned} &AC = b \text{ avec} \\ &A_{lk} = \sum_{i=0}^{n} \frac{d_i}{d_i} B_l(t_i)\, B_k(t_i)\, T_i \otimes T_i^T + B_l(t_i)\, B_k(t_i)\, N_i \otimes N_i^T,\ 1 \leq l, k \leq d-1 \\ &b_k = \sum_{i=0}^{n} \frac{d_i}{d_i} \langle R_i, T_i \rangle B_k(t_i)\, T_i + \langle R_i, N_i \rangle B_k(t_i)\, N_i,\ 1 \leq k \leq d-1 \\ &C_k = c_k,\ 1 \leq k \leq d-1 \end{aligned} \tag{3.2.8}$$

avec $c_0 = P_0$ *et* $c_d = P_n$*,* $R_i = P_i - P_0 B_0(t_i) - P_d B_d(t_i)$
et où $\langle \cdot, \cdot \rangle$ *désigne le produit scalaire et* $\otimes$ *désigne le produit tensoriel.*

La matrice A *est de taille* $3\,(d-2) \times 3\,(d-2)$*, les vecteurs* b *et* C *sont de taille* $3\,(d-2)$*. La notation* A_{lk} *désigne le bloc* 3×3 *indexé aux indices* l*,* k *dans la matrice* A*. La notation* b_k *(respectivement* C_k*) désigne le bloc* 3×1 *indexé à l'indice* k *dans le vecteur* b *(respectivement* C*).*

Démonstration. La démonstration est similaire à celle de la proposition 3.2.2. □

3.2.5 Terme correctif

La formulation de l'algorithme d'approximation par courbe B-Spline donnée par l'équation (3.2.1) inclut un terme correctif λf_s qui permet de contrôler la régularité de la courbe. Ce terme

correctif permet d'inclure l'énergie de la courbe dans la fonctionnelle à minimiser. L'énergie de la courbe se traduit généralement par une combinaison des énergies :

$$E_1 = \int \|\sigma'(t)\|^2 \, \mathrm{d}t, \qquad E_2 = \int \|\sigma''(t)\|^2 \, \mathrm{d}t$$

En pratique, ces énergies sont utilisées dans leur forme discrète, soit :

$$f_1 = \frac{1}{2} \sum_{i=1}^{n} \|\Delta c_i\|^2, \qquad \Delta c_i = c_i - c_{i-1} \tag{3.2.9}$$

$$f_2 = \frac{1}{2} \sum_{i=2}^{n} \|\Delta^2 c_i\|^2, \qquad \Delta^2 c_i = c_{i+1} - 2c_i + c_{i-1} \tag{3.2.10}$$

Les termes correctifs f_1 et f_2 sont des formes quadratique des coefficients $\mathbf{c}$. Ils sont exprimés dans leur forme matricielle F_1 et F_2 pour être inclus dans les différents systèmes linéaires $AC = b$ décrit dans les équations 3.2.3 (PDM), 3.2.6 (TDM) et 3.2.8 (SDM). Ces systèmes sont réécrits comme :

$$(A + \lambda(F_1 + F_2))C = b$$

avec λ de l'ordre de 10^{-2} à 10^{-5}.

Par exemple, pour f_1, la matrice F_1 s'écrit comme :

$$f_{1,ij} = \begin{cases} deg(c_i), & \text{si } i = j \\ -1, & \text{si } i \neq j \text{ et } i \text{ relié à } j \text{ par une arête} \\ 0, & \text{sinon} \end{cases}$$

Le degré de c_i correspond au nombre d'arêtes connectées à c_i, soit 1 pour les points aux extrémités et 2 pour les autres.
La matrice correspondant à F_1 a donc la forme suivante :

$$F_1 = \begin{pmatrix} 1 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & 0 & 0 & -1 & 1 \end{pmatrix}$$

De même, la matrice correspondant à F_2 est de la forme suivante :

$$F_1 = \begin{pmatrix} 1 & 0 & -4 & 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & -4 & 0 & 1 & \ddots \\ -4 & 0 & 5 & 0 & -4 & 0 & \ddots \\ 0 & -4 & 0 & 5 & 0 & -4 & \ddots \\ 1 & 0 & -4 & 0 & 6 & 0 & \ddots \\ 0 & 1 & 0 & -4 & 0 & 6 & \ddots \\ 0 & 0 & 1 & 0 & -4 & 0 & \ddots \\ 0 & 0 & 0 & 1 & 0 & -4 & \ddots \\ 0 & 0 & 0 & 0 & 1 & 0 & \ddots \\ 0 & 0 & 0 & 0 & 0 & 1 & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

3.2.6 Contraintes de tangence

Il est utile de pouvoir intégrer des contraintes de tangence le long des courbes obtenues par approximation.

Définition 3.2.5. *Considérons un ensemble de contraintes $G(c) = 0$ sur la courbe $\sigma(t)$. Alors, le système de minimisation* (3.2.1) *devient :*

$$\begin{cases} \min_c J(c) = \sum_{i=0}^{n} d(P_i, \sigma(t_i))^2 + \lambda f_s \\ G(c) = 0 \end{cases} \tag{3.2.11}$$

Les contraintes $G(c) = 0$ sont intégrées à la fonctionnelle avec des multiplicateurs de Lagrange. Résoudre (3.2.11) équivaut à résoudre :

$$J(c) = \sum_{i=0}^{n} d(P_i, \sigma(t_i))^2 + \lambda f_s + \sum_{j=0} \gamma_j G_j(c) \tag{3.2.12}$$

avec γ_j les multiplicateurs de Lagrange.

En écriture matricielle, le système (3.2.12) s'écrit :

$$\begin{cases} (A + \lambda F)C + M^T \Gamma = b \\ MC = T \end{cases} \tag{3.2.13}$$

ou encore

$$\begin{pmatrix} A_F & M^T \\ M & 0 \end{pmatrix} \begin{pmatrix} C \\ \Gamma \end{pmatrix} = \begin{pmatrix} b \\ T \end{pmatrix}$$

avec $A_F = (A + \lambda F)$ la matrice contenant les termes correctifs introduits dans la section 3.2.5, $MC - T = 0$ correspondant aux contraintes $G(c) = 0$ et Γ le vecteur contenant les multiplicateurs de Lagrange γ_j.

La deuxième forme d'écriture montre qu'il suffit d'étendre les matrices des systèmes linéaires 3.2.3 (PDM), 3.2.6 (TDM) et 3.2.8 (SDM) avec les matrices M et T pour inclure des contraintes de tangence $G(c)$.

Dans nos applications, nous utilisons des contraintes de tangence aux extrémités des courbes. En plus des points cibles, des vecteurs donnant la direction de la tangente aux extrémités sont définis, $D = \{d_0, d_n\}$. Les points de contrôle $\mathbf{c}$ de la courbe doivent vérifier : $c_0c_1 \wedge d_0 = 0$ et $c_{n-1}c_n \wedge d_n = 0$.

3.2.7 Résumé de l'algorithme

L'algorithme 1 expose le fonctionnement de la méthode d'approximation par courbe B-Spline de façon générale. Nous avons implémenté ces algorithmes en C++, en utilisant la librairie open source ***openNURBS*** qui implémente les fonctions d'opération de base sur les NURBS : évaluation, dérivée etc.

Pour adapter automatiquement la courbe B-Spline à l'ensemble des points cibles traités, un système itératif avec une incrémentation du nombre de points de contrôle en fonction de l'erreur commise est appliqué. L'erreur est surveillée sur trois itérations successives. Si elle ne diminue pas alors que le nombre de points de contrôle augmente, l'algorithme est arrêté pour éviter d'ajouter des degrés de liberté superflus.

L'erreur commise entre la courbe et les points cibles se calcule en fonction de la méthode choisie : PDM, TDM ou SDM, en appliquant l'une des formules décrites dans les équations 3.2.3 (PDM), 3.2.6 (TDM) ou 3.2.8 (SDM).

L'algorithme de construction du système est détaillé ci-dessous : algorithme 2 pour la méthode PDM et algorithme 3 pour la méthode SDM.

Il est important de noter que pour la méthode PDM, les coordonnées (x, y, z) sont indépendantes, et donc 3 systèmes de taille $d - 2$, où d est le nombre de points de contrôle, sont construits pour résoudre chacune des coordonnées. Les points de contrôle des extrémités étant fixés, ils ne sont pas inclus dans le système linéaire. En revanche, pour les méthodes TDM et SDM, il faut résoudre les 3 coordonnées dans un unique système, qui sera donc de taille $3 \times (d-2)$.

Pour l'inversion du système $AC = b$, nous utilisons la fonction *colPivHouseholderQr* de la librairie *Eigen*.

Algorithme 1 : Algorithme général de l'approximation par courbe B-Spline

Data : Maillage avec des arêtes reliant les sous-ensembles de points à approximer avec une courbe

Result : Ensemble des courbes BSpline issues de l'approximation

numéro de l'arête étudié n=0

seuil = erreur minimale entre les points cibles et la courbe souhaitée

while *Arêtes non traités dans le maillage* **do**

- P_i = points de l'edge n
- nombre de points de contrôle = minimum fixé par l'utilisateur
- Erreur de distance entre les points cible et la courbe = *error_curve_Pi*
- **while** *pas d'amélioration de error_curve_Pi sur 3 itérations* **do**
  - Calcul du vecteur de nœuds
  - Calcul paramétrage u_i
  - Résolution du système $AC = b$ (voir algorithme 2 et 3)
  - Calcul de *error_curve_Pi*
  - **if** *error_curve_Pi* $<$ *seuil* **then**
    - break
  - **else**
    - *nbr pts controle* $++$
  - **end**
- **end**
- n=n+1

end

Algorithme 2 : Construction du système PDM

Data : Les points cibles $P = \{P_i : i = 0, ..., n\}$, le nombre de point de contrôle (d), le vecteur de noeud, le vecteur de paramétrage (u_i)
Result : Les matrices A et b du système $AC = b$
$A = Matrix(d-2, d-2) = 0$
$b_{x,y,z} = Vector(d-2) = 0$
$c_{x,y,z} = Vector(d-2) = 0$
for $i = 0$ **to** n **do**
  for $k = 1$ **to** $d-1$ **do**
    $B_k(t_i) =$ BasisFunctionEvaluation($u_i(i)$, k)
    for $j = 1$ **to** $d-1$ **do**
      $B_j(t_i) =$ BasisFunctionEvaluation($u_i(i)$, j)

      $A(k,j) = A(k,j) + B_k(t_i) * B_j(t_i)$
    end
    $B_0(t_i)$=BasisFunctionEvaluation($u_i(i)$, 0)
    $B_d(t_i)$=BasisFunctionEvaluation($u_i(i)$, d)
    $R_{x,y,z} = P_{i\ x,y,z} - (P_{0\ x,y,z}\, B_0(t_i) + P_{n\ x,y,z}\, B_d(t_i))$
    $b_{x,y,z}(k) = b_{x,y,z}(k) + B_k(t_i)\, R_{x,y,z}$
  end
end
/* Préparation du système pour la fixation des points aux extrémités
  $c_0 = P_0$ et $c_d = P_n$ */
$C(0) = P_0$
$C(d) = P_n$

Algorithme 3 : Construction du système SDM

Data : Les points cibles $P = \{P_i : i = 0, ..., n\}$, le nombre de point de contrôle (d), le vecteur de noeud, le vecteur de paramétrage (u_i), une courbe initiale (issue de PDM)

Result : Les matrices A et b du système $AC = b$

$A = Matrix(3 \times (d-2), 3 \times (d-2)) = 0$

$b = Vector(3 \times (d-2)) = 0$

$c = Vector(3 \times (d-2)) = 0$

for $i = 0$ **to** n **do**

  /* Calcul du repère de Frenet */

  $tangente_{x,y,z}$, $binomale_{x,y,z}$, $nomale_{x,y,z}$

  /* Calcul des constantes de SDM, ρ et d */

  d, ρ

  for $k = 3$ **to** $d-4,\ k = k+3$ **do**

    $B_k(t_i)$ = BasisFunctionEvaluation($u_i(i)$, k)

    for $j = 3$ **to** $d-4,\ j = j+3$ **do**

      $B_j(t_i)$ = BasisFunctionEvaluation($u_i(i)$, j)

      if $d < \rho$ **then**

        /* Système TDM */

        $A_{k \to k+2, j \to j+2} += B_k(t_i)\, B_j(t_i)\, N \otimes N$

      else

        /* Système SDM */

        $A_{k \to k+2, j \to j+2} += \frac{d_i}{d_i}\, B_k(t_i)\, B_j(t_i)\, T \otimes T + B_k(t_i)\, B_j(t_i)\, N \otimes N$

      end

    end

    $B_0(t_i)$=BasisFunctionEvaluation($u_i(i)$, 0)

    $B_d(t_i)$=BasisFunctionEvaluation($u_i(i)$, d)

    $R = (P_i - (P_0\, B_0(t_i) + P_n\, B_d(t_i))) \cdot N$

    $b_{k \to k+2, j \to j+2} += [B_k(t_i)\, R]\, N$

  end

end

/* Préparation du système pour la fixation des points aux extrémités $c_0 = P_0$ et $c_d = P_n$ */

$C_{0 \to 2} = P_{0\ x,y,z}$

$C_{3d-2 \to 3d} = P_{n\ x,y,z}$

3.2.8 Exemples d'approximation de courbes B-Splines

Nous illustrons ci-dessous plusieurs exemples appliqués a des formes fréquemment utilisées dans ce travail de thèse.

Dans les exemples suivants, l'erreur d'approximations relative à chaque courbe est calculée par :

$$E_{average} = \frac{[\frac{1}{n}\sum_{i=1}^{n} d_i^2]^{1/2}}{L_{ref}} \tag{3.2.14}$$

avec n le nombre de points cibles, $d_i = \|P_i - \sigma(t_P)\|$, la distance du point P_i à sa projection orthogonale $\sigma(t_P)$ sur la courbe σ et L_{ref} la longueur de référence de la courbe.

Exemple 1 - Approximation d'un profil

Le but de cet exemple est de montrer l'application des différentes méthodes d'approximation de courbes (PDM, TDM, SDM) sur la reconstruction d'un profil NACA4412 à partir d'un nuage de point. L'extrados et l'intrados sont chacun décrit avec un nuage ordonné de points illustrés par la figure 3.7

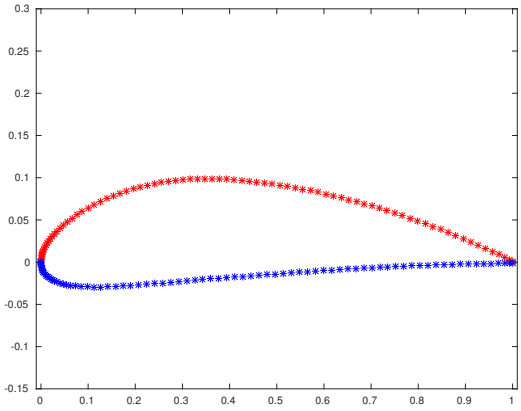


FIGURE 3.7 – Nuages de points représentant l'extrados (bleu) et l'intrados (rouge) d'un profil NACA4412

La méthode PDM est utilisée sur chaque côté du profil avec une paramétrisation calculée selon la méthode centripetal [Lee, 1989], une extension de la méthode Chord Length, pour la première itération. Ensuite, la méthode *Foot Point* est utilisée, avec une mise à jour des paramètres à chaque itération. 5 itérations sont réalisées au total. Pour l'extrados et l'intrados, le temps d'exécution total du code est de $1,85$ secondes.

Pour le calcul de l'erreur donné par l'équation (3.2.14), la longueur de référence du profil est la corde. Dans cet exemple, la corde vaut 1, donc $L_{ref} = 1$

L'erreur finale commise $E_{average}$ pour l'extrados est de $1,07 \times 10^{-4}$ et pour l'intrados de $1,05 \times 10^{-4}$.

La méthode TDM est initialisée avec la courbe produite par la méthode PDM. Les paramètres sont calculés avec une méthode *Foot Point*, sur 5 itérations. Le temps d'exécution total du code est de $3,67$ secondes.

L'erreur finale commise $E_{average}$ pour l'extrados est de $1,08 \times 10^{-4}$ et pour l'intrados de $1,11 \times 10^{-4}$.

La méthode SDM est initialisée avec la courbe produite par la méthode PDM. Tout comme pour l'utilisation de la méthode TDM, les paramètres sont calculés avec une méthode *Foot Point*, sur 5 itérations. Le temps d'exécution total du code est de $3,69$ secondes.
L'erreur finale commise $E_{average}$ pour l'extrados est de $1,09 \times 10^{-4}$ et pour l'intrados de $1,11 \times 10^{-4}$.

Les convergences des méthodes PDM, TDM et SDM sont illustrées dans la figure 3.8.

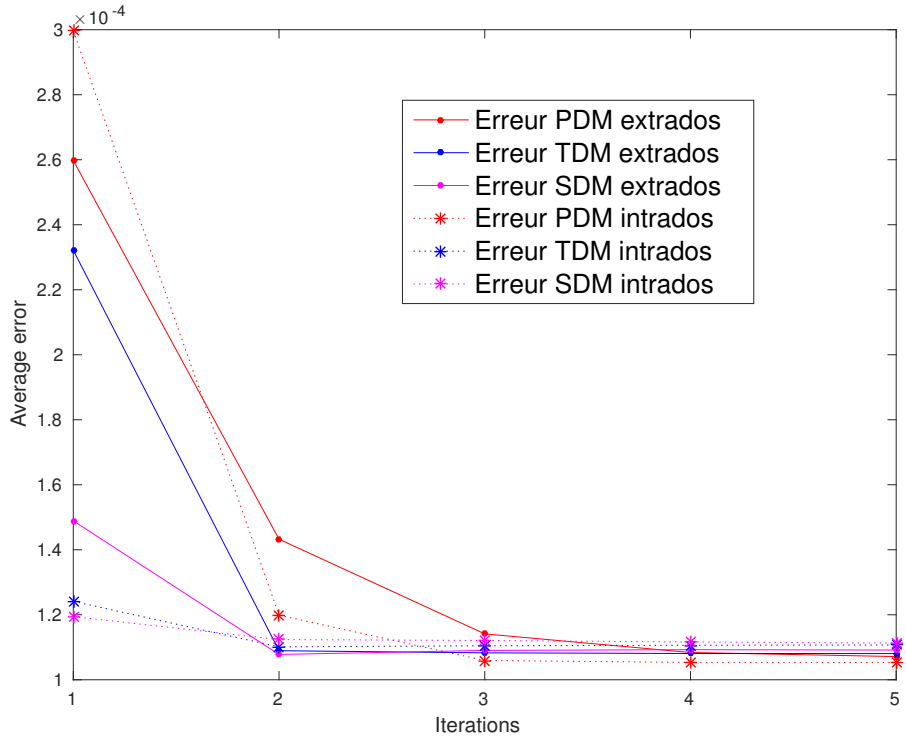


FIGURE 3.8 – Courbes de convergence des méthodes PDM, TDM et SDM pour l'extrados et l'intrados du profil NACA4412

La figure 3.9 montre les courbes obtenues avec les méthodes PDM, TDM et SDM. Pour cet exemple, les courbes obtenues sont très peu différentes les unes des autres comme le montre le graphe de convergence de la figure 3.8.

Exemple 2 - Approximation d'une section de coque de cargo

Le but de cet exemple est de montrer l'application des différentes méthodes de paramétrisation possible sur la reconstruction d'une section d'un cargo à partir d'un nuage de point.

La figure 3.10 montre les courbes obtenues par approximation avec la méthode PDM, utilisées avec la paramétrisation *Foot Point* sur 5 itérations. La coque étant symétrique suivant l'axe Y, seul un des deux côté nécessite d'être reconstruit par approximation. La demi-coque est

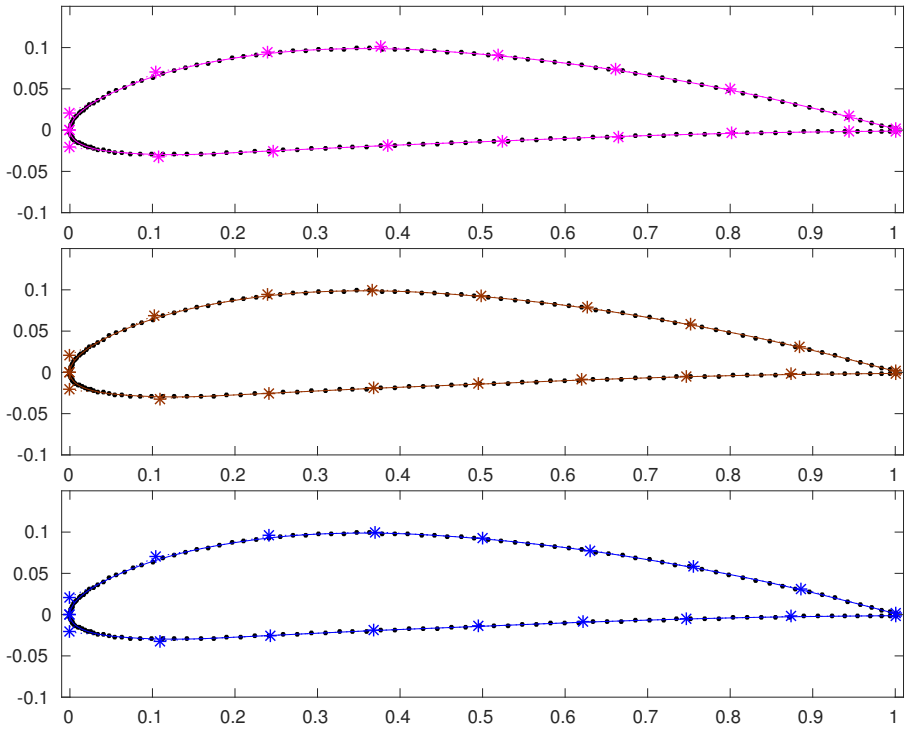


FIGURE 3.9 – Courbes d'approximation du profil NACA4412 : PDM (haut), TDM (milieu), SDM (bas)

composée de $K = 37$ sections. Chaque demi-section est constituée de 10 points de contrôle. Le temps d'exécution pour obtenir les 37 courbes est de $2,82$ secondes.

La moyenne des erreurs $E_{average\ moyen} = \frac{E_{average}}{K}$ est de 5×10^{-4}. La longueur de référence pour la $k^{ième}$, $0 \leq k \leq K$ section utilisée pour calculer $E_{average}$ est :

$$L_{ref,k} = \sum_{i=1}^{n_k} \|P_{i,k} - P_{i-1,k}\|$$

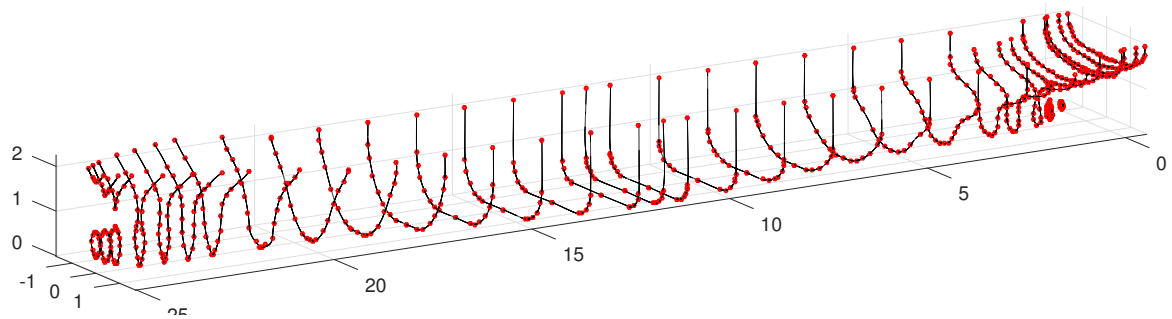


FIGURE 3.10 – Courbes d'approximation du cargo entier

Nous montrons les résultats obtenues pour une section du cargo lorsque la méthode de paramétrisation utilisée est différente. La figure 3.11 montre les différentes courbes obtenues avec la méthode PDM. Pour la méthode *Foot Point*, 5 itérations sont utilisées. Les autres méthodes

ne dépendant pas d'une initialisation de $\sigma(t)$, une seule itération est nécessaire. Nous avons utilisé un petit nombre de points de contrôle, 6, pour accentuer les différences entre les types de paramétrages. La figure 3.11(e) montre la section obtenue pour les paramètres utilisés pour la demi-coque, soit 10 points de contrôle et la méthode *Foot Point*.

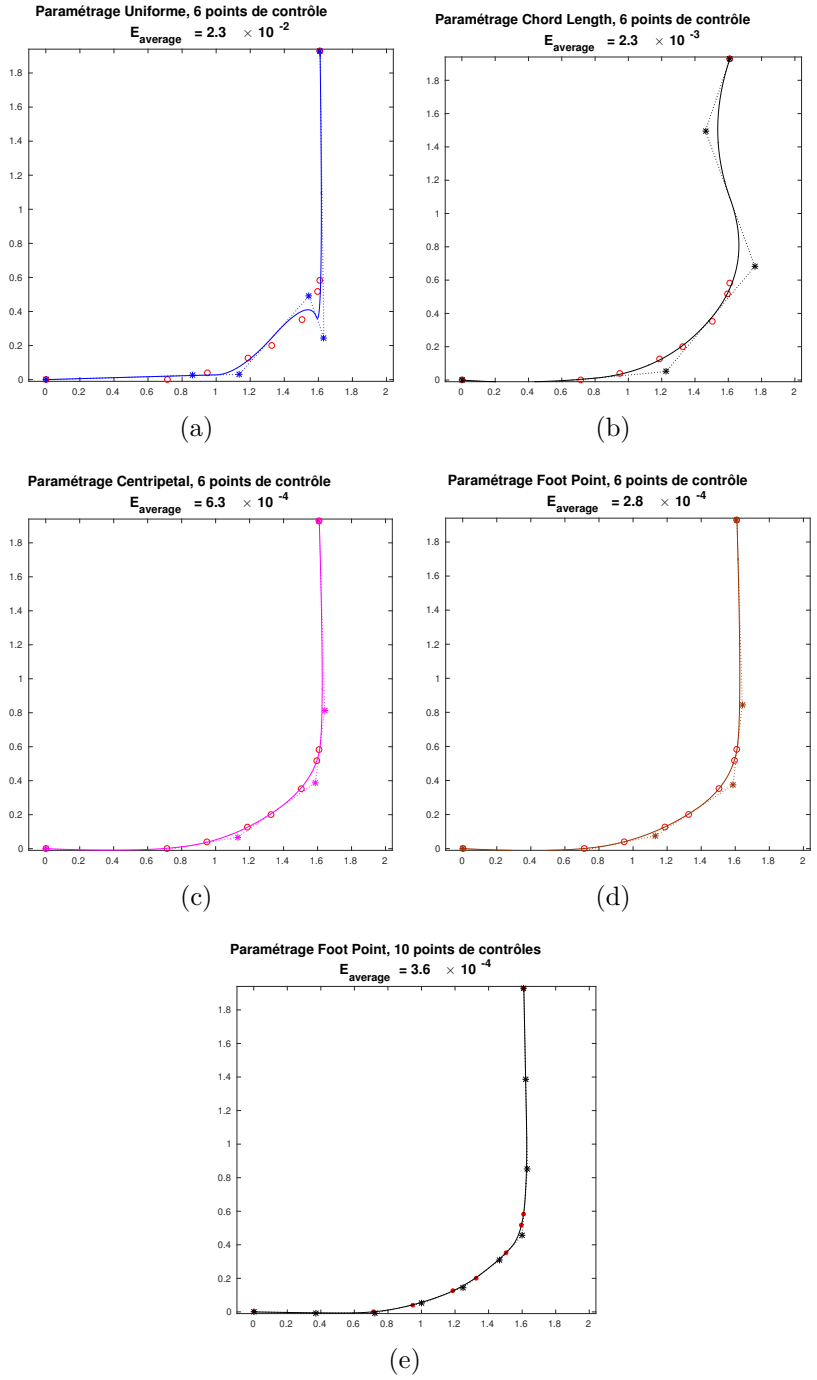


FIGURE 3.11 – Approximation d'une section de cargo avec la méthode PDM en utilisant différent paramétrages

3.3 Fitting de surface B-Spline

De même que pour les courbes, l'approximation de surface consiste à approcher un ensemble de points cibles, $P = \{P_i : i = 0, ..., n\}$ par une surface B-Spline $S(u, v)$, $(u \times v) \in [0, 1] \times [0, 1]$.

La surface B-Spline S de degré p en u, avec le vecteur de nœuds associés u_k, et de degré q en v, avec le vecteur de nœuds associés v_k, est définie comme $S(u, v) = \sum_{l=0}^{m} \sum_{j=0}^{d} c_{l,j} B_{l,p}(u) B_{j,q}(v)$.

Comme pour l'approximation de courbe, le degré et le vecteur de nœuds sont fixés à priori. Les fonctions de bases $B_{l,p}$ sont donc notées B_l, et $B_{j,q}$ sont notées B_j. Notons $N_c = m \times d$, le nombre de points de contrôle total de la surface.

En pratique, nous utilisons généralement $p = q = 3$ et des vecteurs de nœuds u_k, v_k uniformes.

Définition 3.3.1. *Pour approximer un ensemble de points cibles P par une surface B-spline $S(u, v)$, nous cherchons la paramétrisation $S(u, v)$ qui minimise la fonctionnelle suivante :*

$$\begin{aligned} J(\mathbf{c})_{surface} &= \sum_{i=0}^{n} d(P_i, S(u_i, v_i))^2 + \lambda f_s \\ avec\ d(P_i, S(u_i, v_i)) &= \min_{u,v \in [0,1]^2} \|S(u, v) - P_i\|,\ i = 0, ..., n \end{aligned} \tag{3.3.1}$$

où les inconnues $\boldsymbol{c}$ sont les coordonnées des points de contrôle de $S(u, v)$, $d(P_i, S(u_i, v_i))$ représente la distance du point P_i à la surface S, f_s est un terme correctif pour assurer le lissage de la surface et λ une constante positive permettant de pondérer l'influence de f_s.

Ce problème revient à résoudre un système linéaire que l'on peut exprimer en fonction de l'erreur PDM, TDM ou SDM [Bo et al., 2012], comme pour l'approximation de courbe décrite dans la section 3.2.

Paramétrage

A chaque point cible P_i il faut associer un couple de paramètres (u_i, v_i). Il existe plusieurs techniques [Tutte, 1963, Warren, 1996, Floater, 1997, Floater, 2003] pour associer des coordonnées (u_i, v_i) aux points P_i. Celle introduite par [Tutte, 1963] définit l'image (u_i, v_i) de P_i comme une moyenne des coordonnées des points voisins de P_i.

La paramétrisation *Mean Values coordinates* introduite par [Floater, 2003], considère que P_i et ces voisins forment un polygone étoilé centré en P_i. Les angles entre chaque arêtes et les aires des triangles composant le polygone d'origine sont conservés par l'image (u_i, v_i).

Une fois une première paramétrisation obtenue pour initialiser la surface, il est possible d'identifier les paramètres (u_i, v_i) en projetant orthogonalement les P_i sur la surface S, comme pour la technique *Foot Point* décrite dans l'approximation de courbe (voir section 3.2.1).

Si les points P_i sont distribués le long d'iso-courbes de la surface S, il est possible d'assigner une distribution de paramètres calculée pour une courbe dans une direction donnée à toutes les autres courbes de la même direction. Par exemple pour un v_j fixé, une distribution u_i est calculée et ensuite utilisée pour tout les v_i.

Contraintes de bords

Il est souvent préférable qu'une surface obtenue par approximation passe exactement par des bords fixés. Il est possible de déterminer par approximation de courbe B-Spline 4 bords $\sigma_{1,2,3,4}$ par lesquelles la surface passera.

$$\begin{array}{ll} S(0,v) = \sigma_1(v), & S(1,v) = \sigma_2(v) \\ S(u,0) = \sigma_3(u), & S(u,1) = \sigma_4(u) \end{array} \tag{3.3.2}$$

Pour obtenir les courbes $\sigma_{1,2,3,4}$, on résout 4 systèmes linéaires comme décrits dans la section 3.2. Puis, le système linéaire associé à l'approximation de la surface S, décrit par les équations (3.3.4), est réduit pour ne pas inclure les points déjà calculés par les contraintes de bord données par les équations (3.3.2).

Illustration avec la méthode PDM

Le cas de l'utilisation de l'erreur PDM est détaillé ci-dessous. La surface S doit minimiser la fonctionnelle suivante :

Définition 3.3.2.

$$J_{PDM\ surface}(\mathbf{c}) : \sum_{i=0}^{n} \|S(u_i, v_i) - P_i\|^2 \tag{3.3.3}$$

où les inconnues $\mathbf{c}$ *sont les coordonnées des points de contrôle de* $S(u,v)$*, et les couples* (u_i, v_i) *associés aux points* P_i *sont issus du calcul du paramétrage.*

Proposition 3.3.1. *Les coordonnées des points de contrôle de* $S(u,v)$ *sont solution du système linéaire suivant :*

$$\begin{aligned} &AC = b \text{ avec} \\ &A_{lk} = \sum_{i=0}^{n} B_l(t_i)\, B_k(t_i),\ 1 \leq l,k \leq 3\,(N_c - 1) \\ &b_k = \sum_{i=0}^{n} P_i\, B_k(t_i),\ 1 \leq k \leq 3\,(N_c - 1) \\ &C_k = c_k,\ 1 \leq k \leq 3\,(N_c - 1) \end{aligned} \tag{3.3.4}$$

La matrice A *est de taille* $3\,(N_c - 2) \times 3\,N_c$*, les vecteurs* b *et* C *sont de taille* $3\,(N_c - 2)$*. Ces matrices sont des matrices par blocs* 3×3 *pour* A *et* 3×1 *pour* b *et* c*. Chaque bloc correspond aux trois coordonnées* (x, y, z) *d'un point de contrôle.*

Exemples

La section 6.3.3 du chapitre 6 montre des exemples d'utilisation de l'approximation de surfaces appliqué pour une coque de voilier et un bulbe de chalutier.

Chapitre 4

Paramétrisation de forme

L'objectif de ce travail de thèse est de développer un modeleur paramétrique générique capable de déformer des objets de façon cohérente d'un point de vue métier. Notre outil doit pouvoir décrire une large gamme d'objets et s'appliquer à des domaines d'expertises variés tels que la construction navale, l'offshore, l'aéronautique, l'éolien, l'automobile, etc.

Le modeleur paramétrique permet de générer un ensemble de formes filles à partir d'une forme originale, dit forme mère, de façon efficace et intuitive pour les architectes. Les formes obtenues doivent être valides et réalistes. De plus, les géométries générées doivent être adaptées à une utilisation pour la simulation numérique.

Pour obtenir un modèle permettant de répondre à ces contraintes, il faut considérer à la fois la description géométrique de l'objet et ses caractéristiques architecturales.

C'est pourquoi nous construisons un modèle basé sur une double paramétrisation des formes :

1. La *paramétrisation géométrique* : cette partie concerne la description géométrique de l'objet, c'est-à-dire sa définition en terme de courbes et surfaces. Cet aspect de la paramétrisation, concrétisé par le *squelette* est généralisable et peut être utilisé sur un grand nombre d'objets.
2. La *paramétrisation métier* : cette partie concerne les caractéristiques architecturales propres à chaque objet étudié. Nous définissons des ensembles de paramètres architecturaux de la façon la plus générale possible. Par exemple les notions de mesures de longueurs, d'angles, de rayons de courbures, de tangences, etc. sont applicables à un grand nombre d'objets.

   C'est *via* les paramètres métier que l'utilisateur ou l'algorithme d'optimisation de forme va piloter les déformations de l'objet.

Le modèle que nous proposons permet une représentation hiérarchique d'un objet. Les paramètres métier sont les composants élémentaires de la description de la forme. Puis le squelette composé de la génératrice et des sections définissent une couche supplémentaire plus complexe et plus précise. Finalement, la surface enveloppant le squelette, dont des méthodes de reconstructions sont proposées dans le chapitre 6, est le niveau de description le plus haut de la forme. La surface décrit exactement l'objet étudié.

L'intérêt de cette hiérarchisation est de permettre l'application d'algorithmes efficaces et parallélisables aux paramètres ou aux courbes du squelette, qui sont relativement peu complexes.

Puis les transformations subies par le squelette sont transposées à la surface, qui n'est pas modifiée directement.

Les deux aspects de la paramétrisation de formes sont décrits ci-dessous, dans les sections 4.1 et 4.2.

4.1 Paramétrisation géométrique

Nous définissons la paramétrisation géométrique d'un objet par la création d'un *squelette*.

Le squelette est défini comme un ensemble de courbes composé d'une génératrice et de courbes de section. Chaque courbe de section doit être identifiée sur la génératrice : un repère local est associé à chaque section permettant de connaître sa position et son orientation.

L'idée de squelette est basée sur deux considérations :

1. La première concerne la correspondance avec le plan de forme classique utilisé par les architectes pour concevoir un navire [Paulet and Presles, 1999], illustré dans la figure 4.1
2. La deuxième est l'utilisation d'une technique courante en animation 3D, qui consiste à utiliser un axe médian de l'objet pour le déformer [Yoshizawa et al., 2007].

En associant ces deux types de représentation de forme, nous pouvons appliquer à un objet des algorithmes de déformations génériques, tout en assurant leur cohérence architecturale.

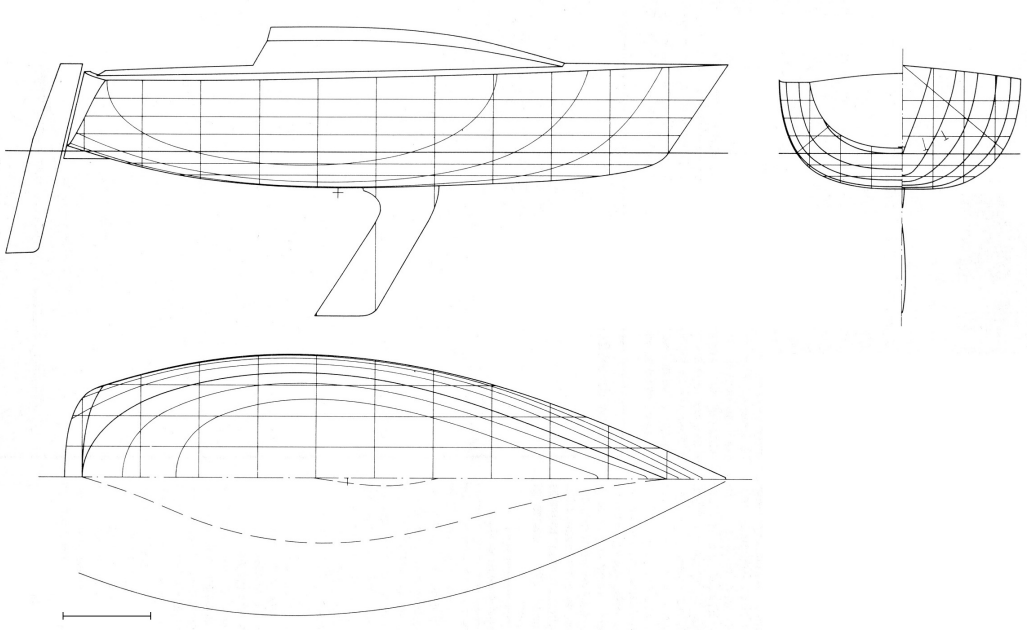


FIGURE 4.1 – Exemple de plan de forme d'une coque de bateau (*Plan Bateaux magazine 1980*)

Cette conception du squelette est suffisamment générique pour être appliquée à un grand nombre d'objets. Par exemple, il est possible de définir un voilier entier avec cette description : un squelette pour la coque, un squelette pour le mat, et un squelette par appendice (safran, foil, etc.). Les hélices, les pales d'éolienne ou d'hydrolienne se prêtent très bien à cette description,

tout comme le fuselage ou les ailes d'un avion. Les sous-marins, les ballons dirigeables, les câbles et *pipes* utilisés en offshore peuvent également être décrits avec un squelette.

4.1.1 Le squelette : génératrice et courbes de section

Le squelette est composé de deux types de courbe : la génératrice et les sections. La génératrice décrit la forme globale de l'objet. Les sections décrivent précisément les contours, tout comme le plan de forme d'un architecte naval.

La génératrice est choisie comme la courbe qui décrit les caractéristiques principales de la géométrie. Déterminer la génératrice dépend de l'objet étudié. Par exemple pour une coque de navire, la ligne de quille représente le mieux la forme générale. Pour les formes profilées, comme les ailes d'avions, les pales, les hélices, les safrans ou les foils, le bord de fuite est le choix le plus approprié.

Les courbes de section sont calculées comme l'intersection de l'objet et d'une famille de plans de coupe $\mathcal{P} : \{\mathcal{P}_i : i = 0, ..., N\}$. Ces plans sont définis comme normaux à la tangente de la génératrice.

Nous faisons exception à cette règle pour les coques de bateau et le bulbe, car dans ce cas il est plus logique de suivre le plan de forme classique des architectes. Cela revient à choisir une génératrice dans le plan XZ (la ligne de quille) et à choisir les sections suivant la direction Y.

Nous associons à chaque section un point sur la génératrice et un système de coordonnées local qui nous permet de déterminer sa position et son orientation.

En pratique, nous représentons la génératrice et les sections comme des courbes B-Splines. Ces courbes B-Splines sont obtenues en utilisant les algorithmes d'approximation de courbes décrits dans la section 3.2. Il est ainsi possible de choisir le nombre de points de contrôle utilisés pour chaque courbes. La géométrie est donc représentée par un ensemble fini de points de contrôle.

Appelons $\mathbf{c}_g$ les points de contrôle de la courbe génératrice définie comme :

$$C_g(t_g) = \sum_{j=0}^{n_g} \mathbf{c}_{g,j} B_{j,p}(t_g),\ t_g \in [0,1]$$

et $\mathbf{c}_i$ les points de contrôle de la i^{ème} section $C_{s,i}(t_c)$, pour $i = 1, \ldots, N$, avec :

$$C_{s,i}(t_c) = \sum_{k=0}^{n_i} \mathbf{c}_{i,k} B_{k,q}(t_c),\ t_c \in [0,1]$$

avec $B_{j,p}$ (respectivement $B_{k,q}$) les fonctions de bases de la courbe B-Spline C_g (respectivement $C_{s,i}$) de degré p (respectivement q) et définies sur le vecteur de nœuds $t_{g,i}$ (respectivement $t_{c,i}$).

4.1.2 Repère local des courbes de section

Les courbes de section sont identifiées sur la génératrice grâce à un point et un système de coordonnées local. La $i^{ème}$ section est associée au point P_i et au repère R_i.

Cette méthode permet de définir la génératrice et les courbes de section de façon indépendante. Les sections sont planaires et sont définies comme des courbes en 2D. Lorsque c'est possible, la génératrice est aussi décrite comme une courbe 2D. Décrire les courbes en 2D permet de réduire le nombre de degrés de liberté du problème de déformation.

A tout instant, le squelette en 3D peut être reconstruit en utilisant les repères locaux.

Nous illustrons un exemple sur le squelette d'un foil. La génératrice est choisie le long du bord de fuite du foil, c'est une courbe en 3D comme le montre la figure 4.3. Les sections sont des courbes 2D, comme illustré par la figure 4.4. La figure 4.2 montre le squelette orignal en 3D du foil.

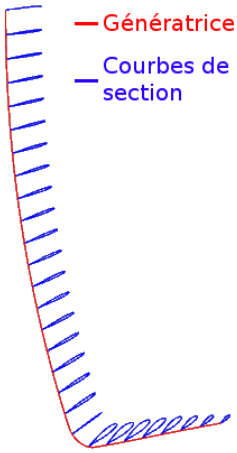


FIGURE 4.2 – Squelette du foil en 3D

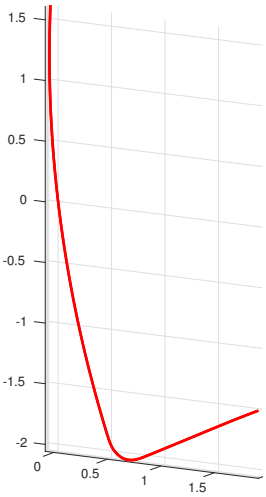


FIGURE 4.3 – Courbe génératrice du foil

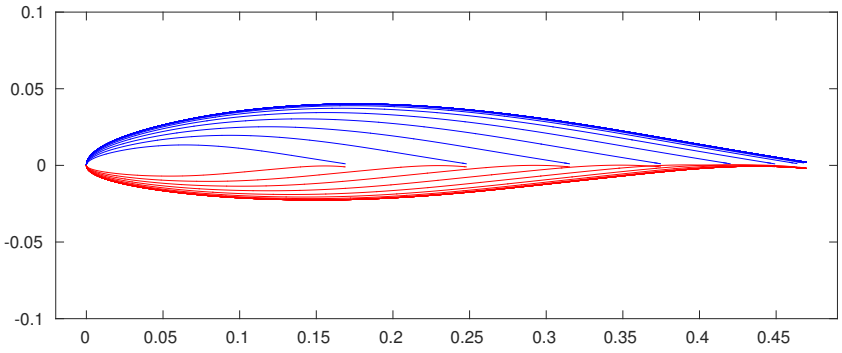


FIGURE 4.4 – Courbes de section du foil, en 2D superposées sur le même repère

Le point P_i associé à la $i^{ème}$ section correspond à une valeur de paramètre $t = t_i$ sur la courbe génératrice $C_g(t)$ où a été définit la section. Cette section se situe au paramètre t_i où la tangente a été calculée, définissant le plan de coupe $\mathcal{P}_i$. Ce point $P_i = C_g(t_i)$ est appelé *point*

d'attache de la section.

Le repère local R_i de la $i_{ème}$ section est associé au plan de coupe $\mathcal{P}_i$ permettant d'obtenir la section.

L'origine de chaque repère local est un point de la section, généralement à l'une des extrémités de la courbe. Il n'est pas nécessairement identique à P_i.

Le premier axe du repère est représenté par la tangente T_i de la génératrice au point d'attache P_i. Le deuxième axe N_i est défini dans le plan de la section, orthogonal au premier axe par construction (nous rappelons que la section est définie par un plan de coupe orthogonal à T_i). La direction de N_i est imposée par les caractéristiques de la section. Par exemple, N_i est la direction de la corde pour une section de type profil. Enfin, la troisième direction S_i est simplement calculée comme étant orthogonale aux deux autres par un produit vectoriel : $S_i = N_i \wedge T_i$.

Les figures 4.5 et 4.6 montrent les repères locaux utilisés pour le foil. Le premier axe T_i est en rouge, N_i en vert et S_i en magenta. N_i est choisi suivant la corde du profil.

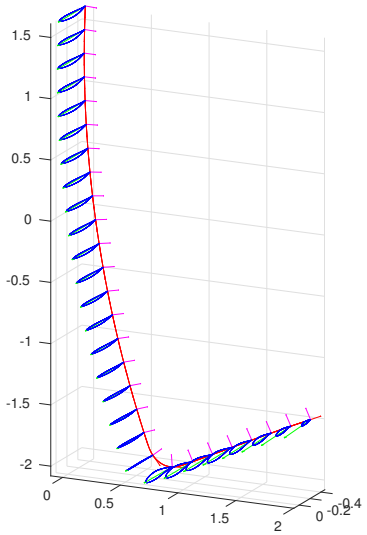


FIGURE 4.5 – Vue générale des repères le long du foil

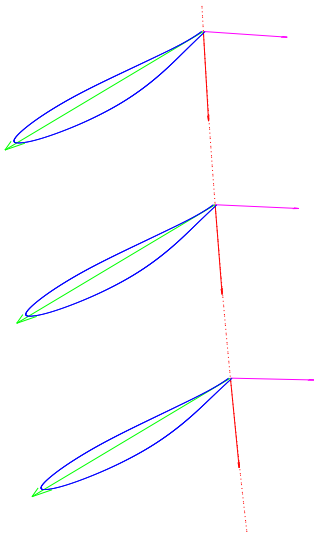


FIGURE 4.6 – Zoom sur les repères locaux de trois sections

Les figures 4.7 et 4.8 montrent les repères locaux utilisés pour le bulbe. Le premier axe T_i est en rouge, N_i en vert et S_i en magenta. Les T_i suivant la génératrice sont dans le plan XZ. Les N_i sont donc choisis suivant la direction Y.

Dans ce cas, les *points d'attache* sont différents de l'origine du repère local. En effet, le bulbe est raccordé à la ligne de quille par une surface plane qu'il n'est pas nécessaire de modéliser pour déformer le bulbe. Les *points d'attache* sont visibles en magenta sur la figure 4.8, et les origines des repères locaux sont en noir.

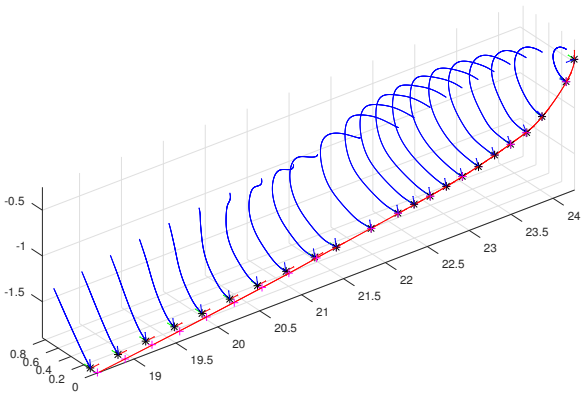


FIGURE 4.7 – Vue générale des repères le long du bulbe

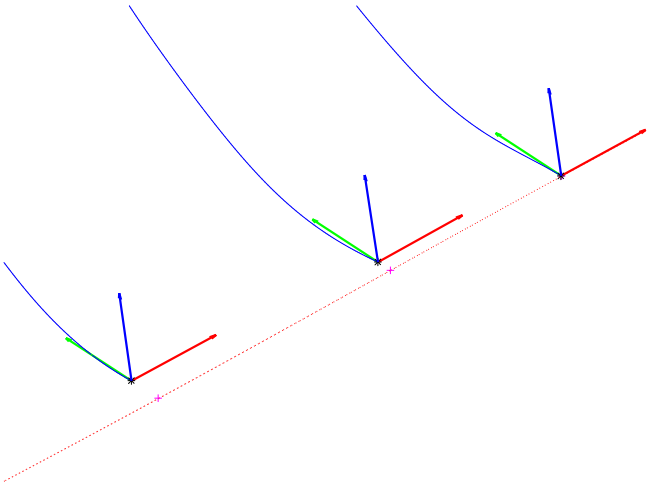


FIGURE 4.8 – Zoom sur les repères locaux de trois sections

4.1.3 Implémentation de l'extraction du squelette

Dans cette section, nous expliquons la technique d'extraction du squelette à partir d'une forme existante.

Nous utilisons le logiciel de CAO Rhinocéros 3D^{TM} qui permet de manipuler des géométries NURBS. Ce logiciel est basé sur une librairie open source ***openNURBS*** qui implémente les fonctions d'opération de base sur le NURBS : évaluation, dérivée etc. Rhinocéros 3D^{TM} propose une surcouche propriétaire qui permet de réaliser des opérations plus complexes : intersections, projections de points sur des courbes ou des surfaces, etc.
openNURBS et la surcouche Rhinocéros 3D^{TM} sont utilisables à partir de programmes C++ ou de scripts Python.

Pour des questions de temps et de facilité d'utilisation, nous avons employé la fonctionnalité de scriptage Python intégrée dans Rhinocéros 3D^{TM}. Des implémentations de fonctionnalités avancées (intersection surface/surface, identification du point le plus proche) proposées par la surcouche propriétaire sont disponibles dans la littérature [Patrikalakis and Maekawa, 2002, Dokken and Skytt, 2006, Thomassen et al., 2005].

Génératrice et courbes de section

Pour obtenir le squelette, l'utilisateur sélectionne la courbe génératrice, illustrée en rouge dans la figure 4.9 pour le foil et 4.10 pour le bulbe.

Ensuite, il choisit le nombre s de sections qui vont être utilisées. s points sont donc échantillonnés sur la génératrice, y compris sur les deux extrémités. Nous avons choisi une répartition uniforme des échantillons.
Ensuite, s plans de coupe sont générés à partir de la tangente à la génératrice aux points échantillonnés, illustrés par la figure 4.11 pour le foil et 4.12 pour le bulbe.

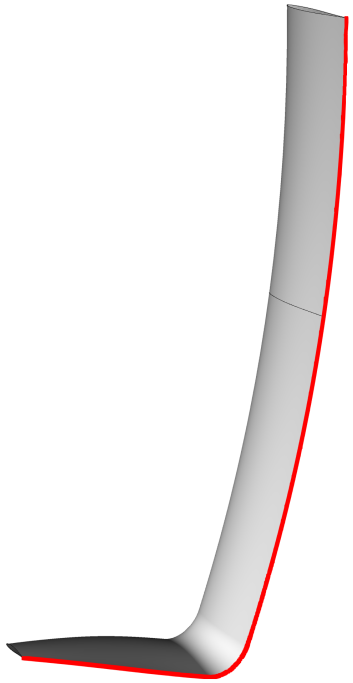


FIGURE 4.9 – Génératrice du foil

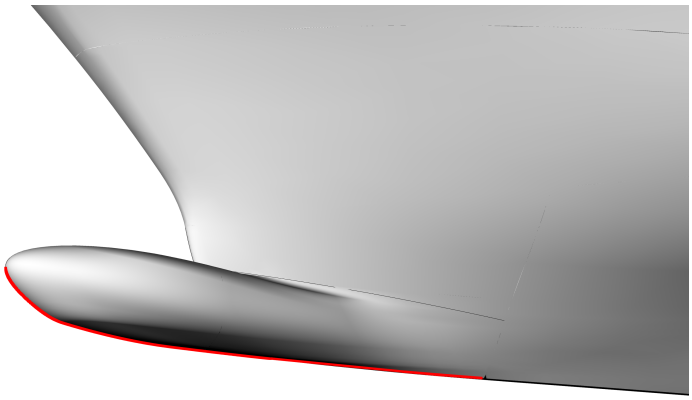


FIGURE 4.10 – Génératrice du bulbe

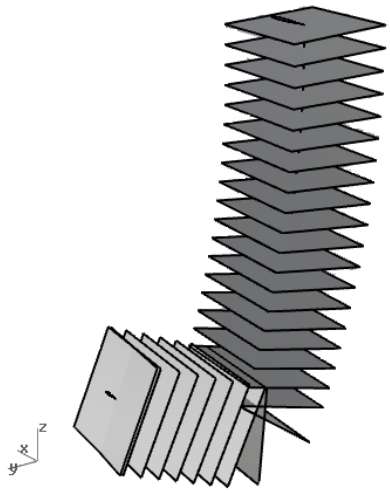


FIGURE 4.11 – Génératrice du foil

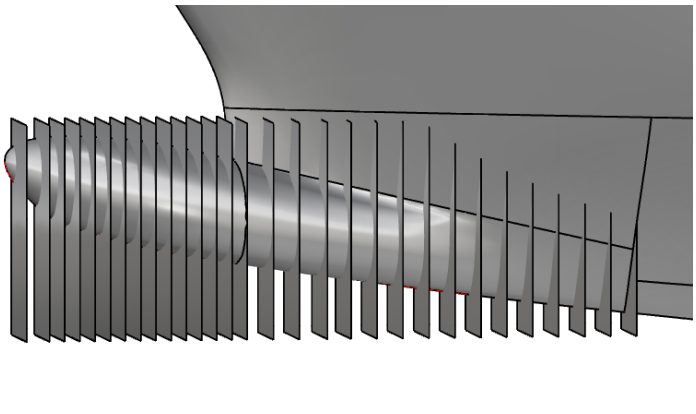


FIGURE 4.12 – Génératrice du bulbe

La fonction d'intersection plan/surface implémentée dans Rhinocéros 3D^{TM} renvoie directement une courbe B-Spline représentant l'intersection. Cette courbe est généralement très complexe. Pour le foil, la génératrice et les courbes de section sont composées d'environ 500 points de contrôle chacune, voir figure 4.13, alors que nous n'en utilisons que 20 avec l'approximation de courbes. Les sections du bulbe sont composées en moyenne de 32 points de contrôle, voir figure 4.14, alors que nous n'en utilisons que 10.

Une fois les courbes d'intersection obtenues avec Rhinocéros 3D^{TM}, nous échantillons ces courbes pour pouvoir créer des courbes B-Splines avec les algorithmes d'approximation.

Les algorithmes d'approximation de courbe B-Spline présentés dans la section 3.2 permettent d'obtenir des courbes de qualité moins complexes que celles générées par Rhinocéros 3D^{TM}. Les algorithmes d'approximation de courbes ont été implémentés en C++, et les format de fichier d'entrée sont compatible avec les sorties du script Python que nous décrivons ici.

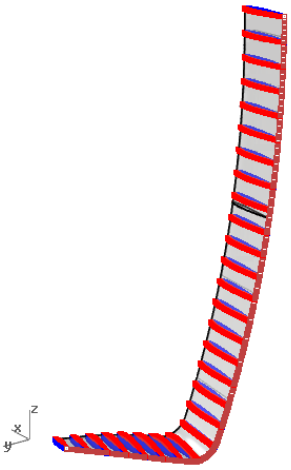


FIGURE 4.13 – Points de contrôle des sections du foil générés par Rhinocéros 3D^{TM}

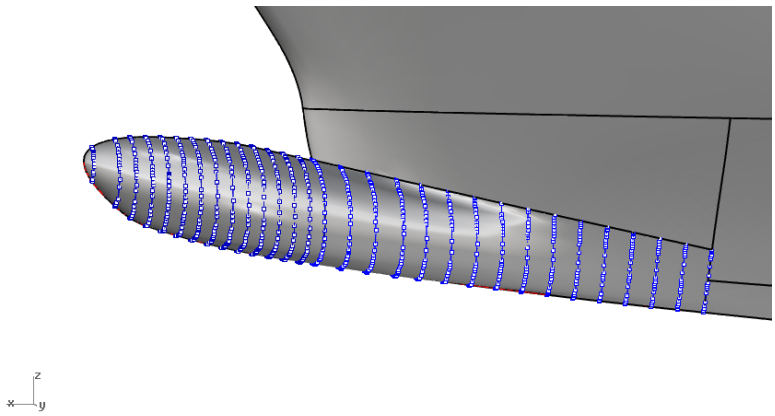


FIGURE 4.14 – Points de contrôle des sections du bulbe générés par Rhinocéros 3D^{TM}

Repères locaux

Les tangentes sont extraites directement de la génératrice aux points échantillonnés.

Le deuxième axe, défini dans le plan de la section, est choisi par l'utilisateur. Il est choisi en fonction du type de section, par exemple la direction qui relie le bord de fuite au bord d'attaque d'un profil. Dans le cas d'une coque, on choisit généralement la direction Y, car il s'agit de la définition usuelle utilisée dans le plan de forme des architectes.

Le troisième axe est calculé comme le produit vectoriel des deux autres.

Les figures 4.15 et 4.16 montrent les squelettes finalement obtenus pour le foil et le bulbe. Nous illustrons aussi le squelette d'une coque de voilier dans la figure 4.17.

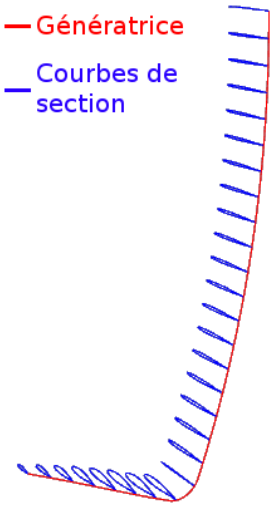


FIGURE 4.15 – Squelette d'un foil AC45

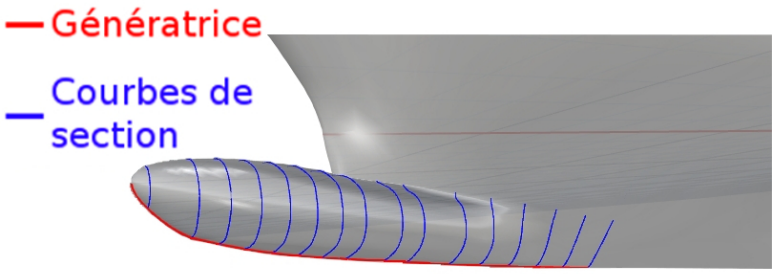


FIGURE 4.16 – Squelette d'un bulbe de chalutier

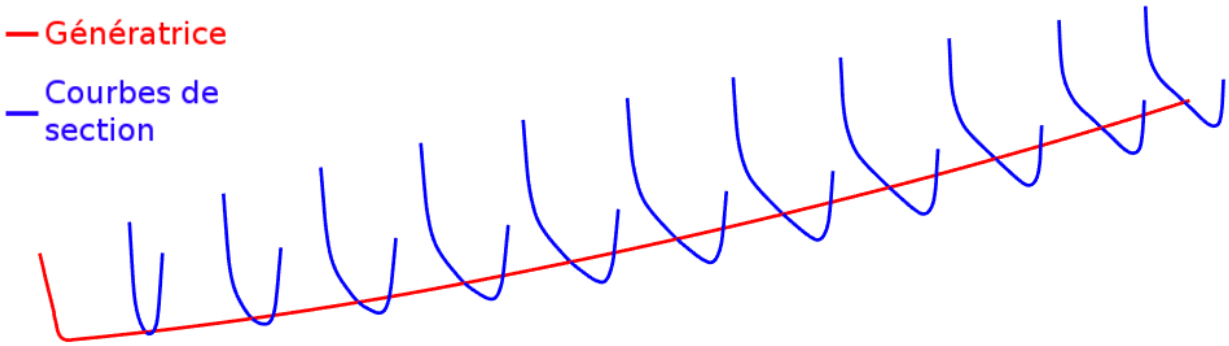


FIGURE 4.17 – Squelette d'une coque de voilier

4.2 Paramétrisation métier : paramètres architecturaux

La paramétrisation métier complète la paramétrisation géométrique décrite précédemment. Les paramètres métier vont être définis sur le squelette et vont permettre de piloter les déformations de l'objet.

Définir et utiliser les paramètres métier pour déformer un objet, plutôt que les points de contrôle de ses courbes ou surfaces, a plusieurs avantages. Les déformations correspondent à des valeurs physiques des paramètre métier. Cela permet de s'assurer de la validité d'un point de vue architectural de la forme générée. Enfin, le nombre de degrés de liberté du problème d'optimisation de forme est considérablement réduit. En effet, en 3D, chaque point de contrôle possède 3 degré de liberté.

Par exemple, les surfaces B-Splines qui composent le foil AC45 contiennent au total 22164 points, soit 66492 degrés de liberté. Comme exposé dans la section 4.2.2, nous définissons sur le foil seulement 14 paramètres métier, c'est-à-dire 14 degrés de liberté.

La surface du bulbe est composée de 57 points de contrôle, soit 171 degrés de liberté. Avec notre paramétrisation, nous considérons seulement 4 paramètres métier, donc 4 degrés de liberté.

Cet aspect de la paramétrisation permet aussi d'assurer la cohérence architecturale des formes générées.

Les paramètres architecturaux décrivent les caractéristiques de l'objet. Ils sont choisis soit pour leur importance dans le design, soit pour l'influence qu'ils ont sur les performances de l'objet.

Fonction *observer*

Considérons la fonction ϕ qui associe à une géométrie G l'ensemble de ses paramètres métier P :

$$\phi : G \longrightarrow P \tag{4.2.1}$$

ϕ caractérise la paramétrisation métier, et nous l'appelons fonction *observer*.

Nous associons différents ensembles de paramètres à la génératrice et aux sections. ϕ peut être décomposée en une fonction *observer* pour la génératrice $\phi_g : G \longrightarrow P_g$ et une fonction *observer* pour l'ensemble des sections $\phi_s : G \longrightarrow P_s$.

Les paramètres P_g associés à la génératrice sont un ensemble fini de valeurs réelles. Les paramètres P_s associés aux sections sont des fonctions continues, dépendant du paramètre t le long de la génératrice. En choisissant un nombre de sections fini, la fonction des paramètres P_s est discrétisée en les t_i, i=0, ..., N.

Pour manipuler en pratique les paramètres des sections, nous introduisons les courbes de *répartition*, décrites ci-après.

4.2.1 Courbe de répartition

La distribution des paramètres des sections P_s le long de génératrice est une fonction continue, qui est discrétisée par le nombre de sections choisies.

Il est possible de contrôler les valeurs des paramètres métier de chaque section indépendamment, mais cela représente un nombre important de degrés de liberté pour l'utilisateur ou l'algorithme d'optimisation de forme. Aussi, si les sections sont traitées de façon indépendantes les unes des autres, il est possible d'impacter la cohérence architecturale globale de l'objet. Par exemple, une variation importante de la valeur d'un paramètre métier d'une section à l'autre ne produit probablement pas une forme valide.

Nous proposons d'approximer la fonction des paramètres P_s par une courbe B-Spline, appelée courbe de *répartition*.

Cette courbe est obtenue en utilisant les algorithmes d'approximation de courbes décrits dans la section 3.2.

Comme nous maîtrisons le nombre de points de contrôle utilisés pour créer la courbe, nous nous assurons que le nombre de degrés de liberté de la courbe est plus petit que le nombre de sections.

De plus, la modification des valeurs de paramètres métier via une courbe B-Spline assure une répartition homogène de la valeur des paramètres métier le long des sections, assurant un lissage de l'objet.

Une courbe de *répartition* est illustrée dans la figure 4.18. Il s'agit de la valeur de la corde des sections le long du foil. La valeur de la corde des 28 sections est illustrée par un point bleu. Nous avons choisi de construire une courbe de *répartition* avec seulement 5 points de contrôle.

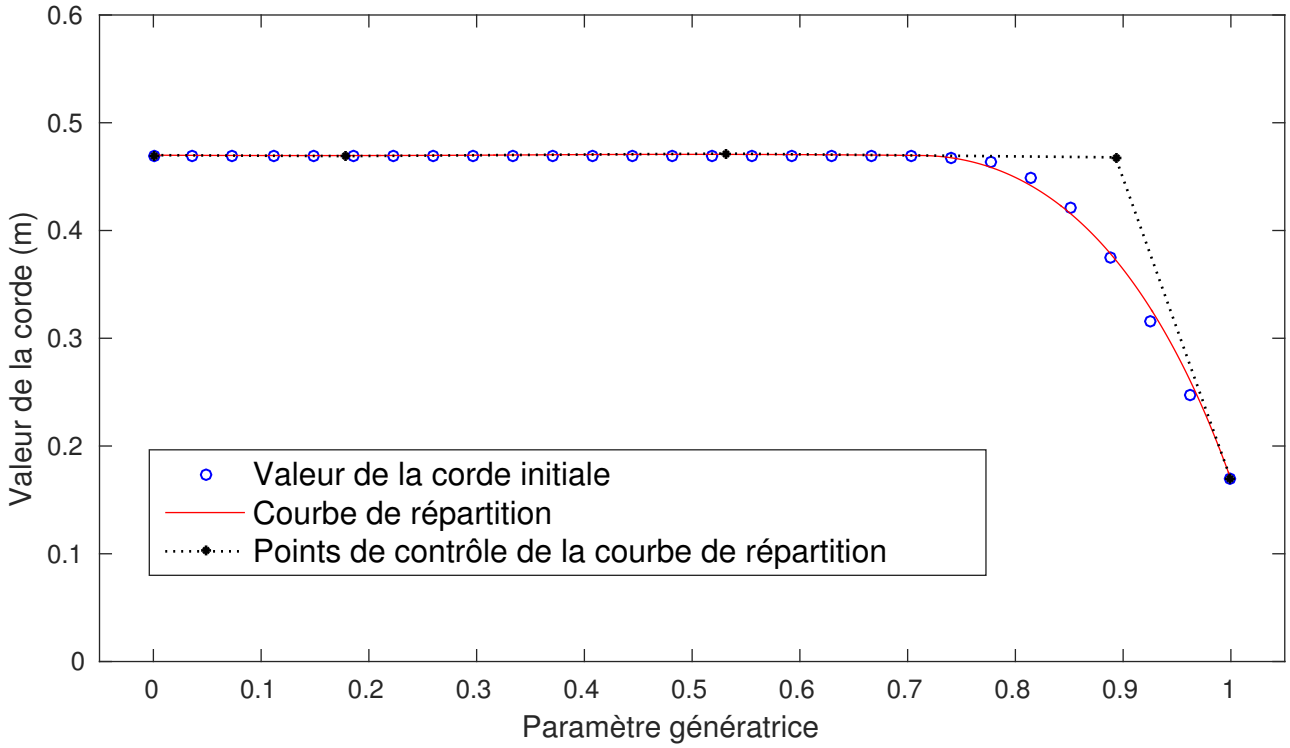


FIGURE 4.18 – Distribution du paramètre de corde le long de la génératrice du foil

4.2.2 Exemples

Exemple 1 - Foil

Dans cet exemple nous considérons un foil en L d'un voilier de course. Ce type de foil est composé de trois parties illustrées dans la figure 4.19 : le *shaft*, le *tip* et le coude qui relie les deux.

Les caractéristiques principales du foil sont la longueur du *shaft*, la longueur du *tip*, l'angle du coude et l'angle de *cant* (angle du *shaft* avec la verticale). Ces paramètres sont les paramètres métier de la génératrice. Ils sont illustrés dans la figure 4.20.

Les sections du foil sont des profils bidimensionnels. Les profils sont composés de deux parties : l'extrados au dessus et l'intrados au dessous. Nous utilisons les caractéristiques classiques de ce

type de forme [Sobieczky, 1999, Kostas et al., 2016] :

- la corde
- l'angle d'attaque
- la hauteur relative de chacun des deux côtés (extrados et intrados), par rapport à la corde
- la position en x de la hauteur de chacun des deux côtés, projetée sur la corde
- la rayon de courbure au bord d'attaque
- la pente de la courbe au bord de fuite

Ces paramètres sont illustrés sur la figure 4.21.

Des paramètres supplémentaires peuvent être considérés, notamment l'aire du profil ou la position de son centre de gravité.

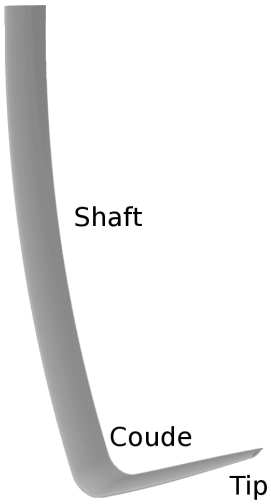


FIGURE 4.19 – Foil en L de voilier de course

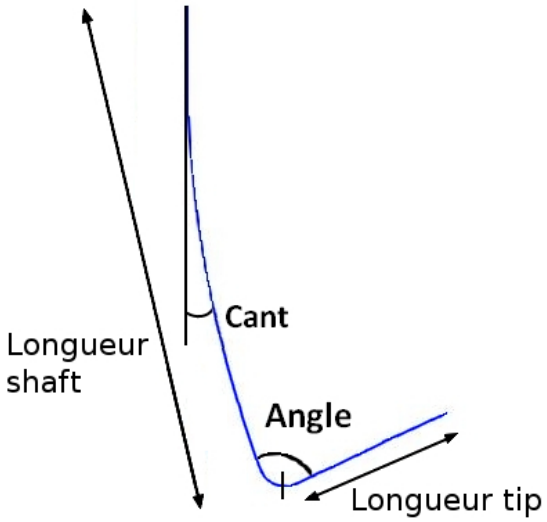


FIGURE 4.20 – Paramètres de la génératrice du foil

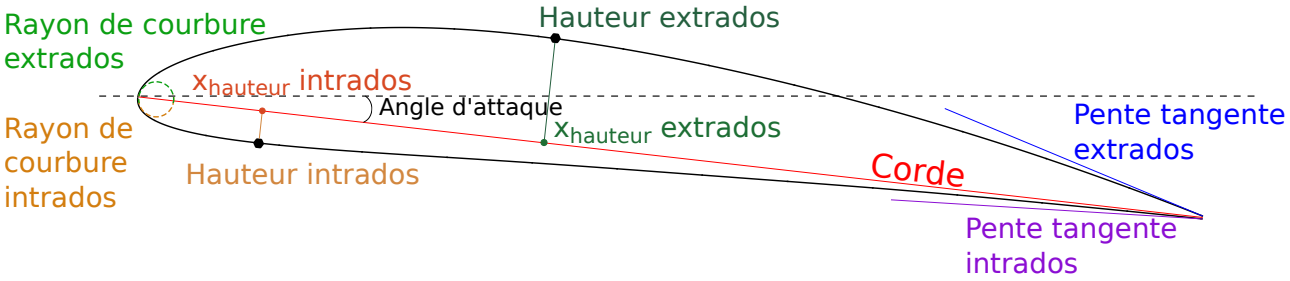


FIGURE 4.21 – Paramètres des sections du foil (profil)

Exemple 2 - Bulbe de chalutier

Dans cet exemple nous considérons un bulbe de chalutier, illustré dans la figure 4.22.

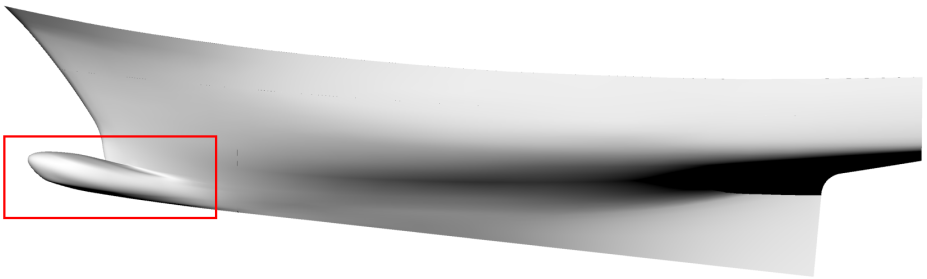


FIGURE 4.22 – Bulbe d'un chalutier

La génératrice du bulbe est identifiée par la ligne de quille de la coque. Les paramètres usuels d'un bulbe sont décrits dans [Kracht, 1978].

Nous avons identifié la longueur et l'angle comme paramètres de la génératrice, illustrés dans la figure 4.23. La hauteur et l'épaisseur sont choisis comme paramètres des sections, illustrés dans la figure 4.24.

D'autres paramètres pourront être aussi considérés comme l'aire des sections, la position verticale du centre de gravité des sections, le type de bulbe (delta, ovale, nabla) ou le volume total.

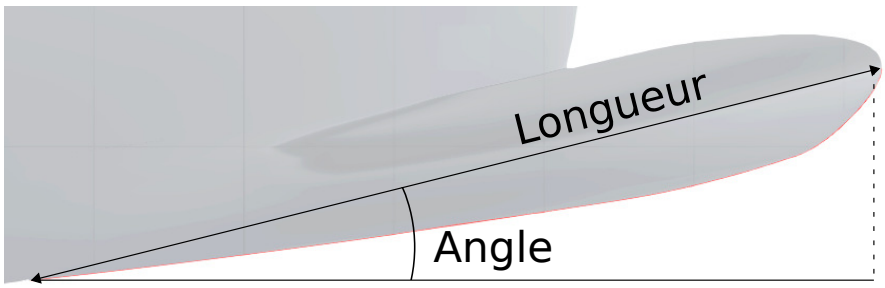


FIGURE 4.23 – Paramètres de la génératrice du bulbe

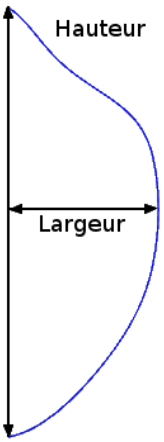


FIGURE 4.24 – Paramètres des sections du bulbe

Exemple 3 - Coque de voilier

Dans cet exemple nous considérons la coque d'un voilier, illustrée dans la figure 4.25.

FIGURE 4.25 – Coque de voilier

Pour la génératrice nous considérons comme paramètres de la ligne de quille la longueur et les pentes de la courbe aux extrémités. Pour l'étrave, nous considérons la longueur et l'angle formé avec la ligne de quille. Les paramètres sont illustrés par la figure 4.26.

Pour les sections, nous considérons la longueur, la largeur et le rayon central, comme illustré par la figure 4.27. Nous illustrons les courbes de répartition associées à chacun de ces paramètres dans les figures 4.28, 4.29, 4.30.

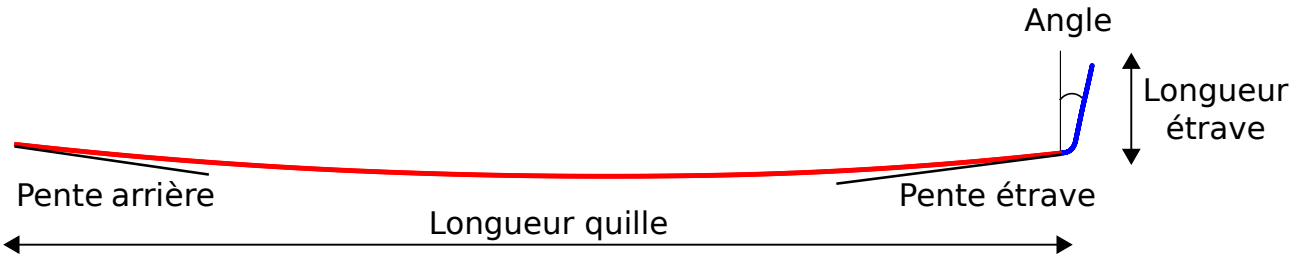


FIGURE 4.26 – Paramètres de la génératrice de la coque de voilier

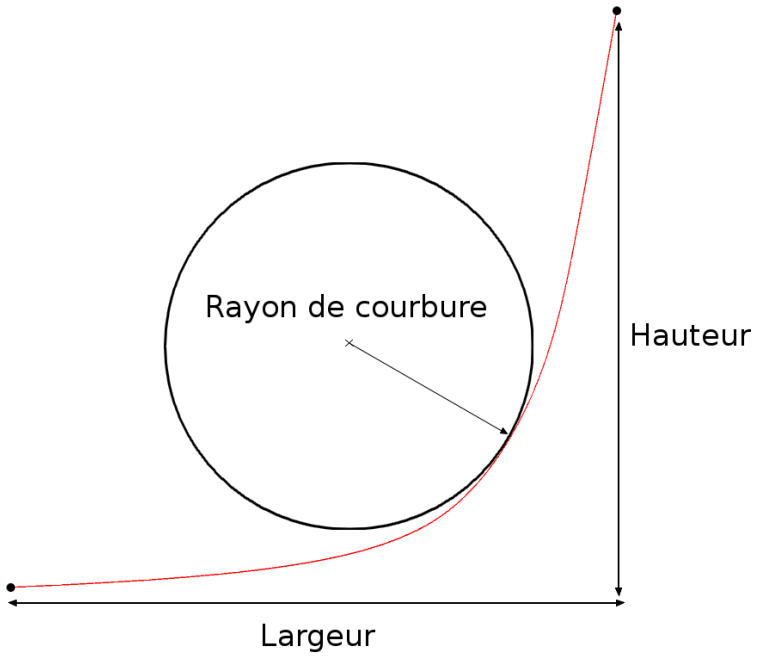


FIGURE 4.27 – Paramètres des sections de la coque de voilier

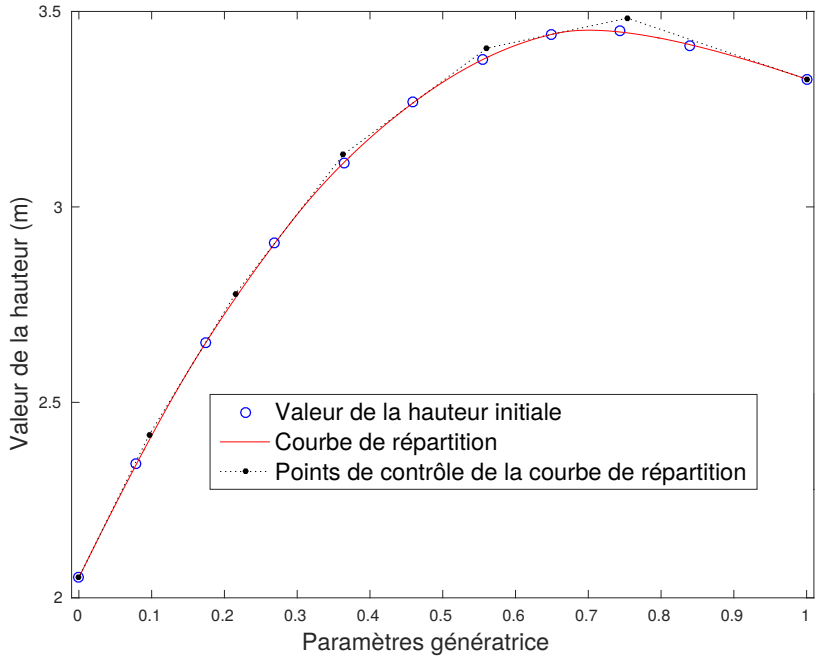


FIGURE 4.28 – Courbe de répartition associée au paramètre de la hauteur de la coque de voilier (de l'arrière vers l'étrave)

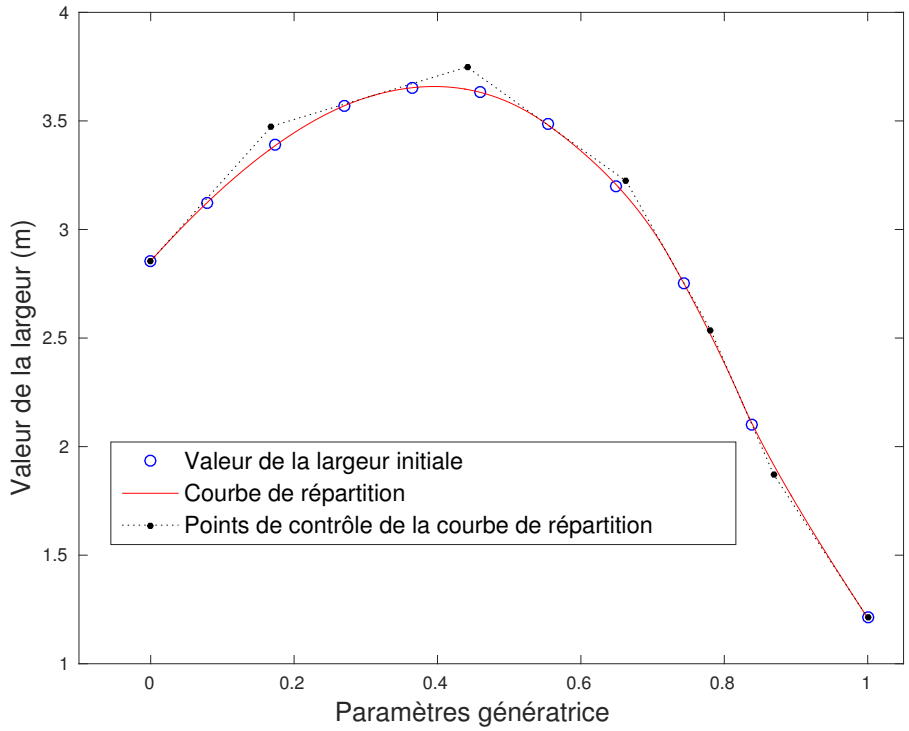


FIGURE 4.29 – Courbe de répartition associée au paramètre de la largeur de la coque de voilier (de l'arrière vers l'étrave)

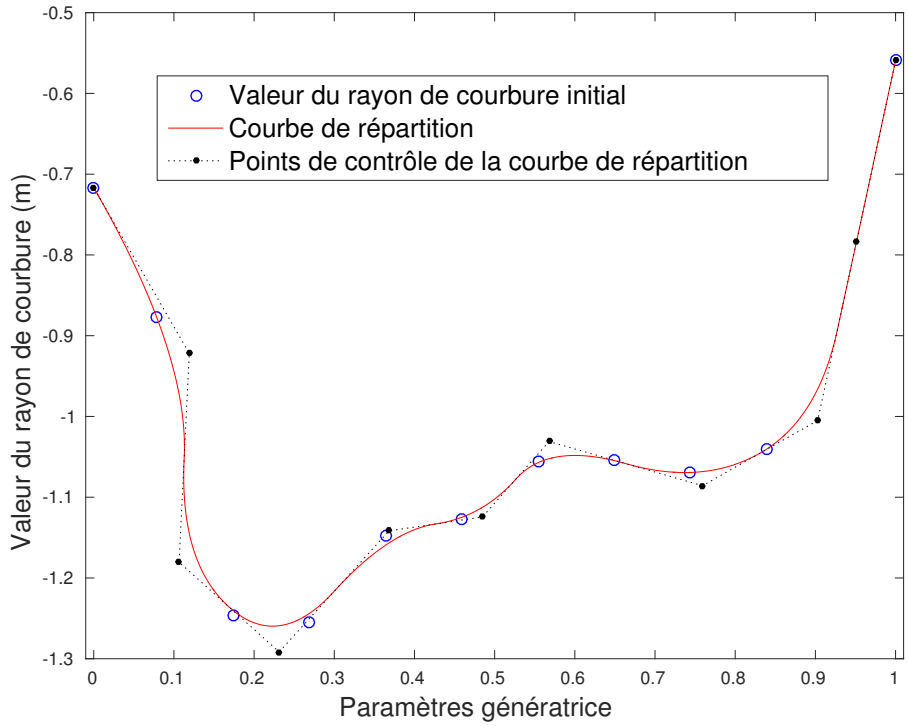


FIGURE 4.30 – Courbe de répartition associée au paramètre du rayon de courbure de la coque de voilier (de l'arrière vers l'étrave)

Chapitre 5

Méthode de déformation

Dans le chapitre précédent, nous avons introduit les notions de paramétrisation de forme géométrique et métier. La paramétrisation géométrique se base sur la description de l'objet par un squelette composé de courbes B-Splines, avec une courbe génératrice et des courbes de section. La paramétrisation métier consiste à définir sur ce squelette un ensemble de paramètres caractéristiques de l'objet, pertinents d'un point de vue architectural. Les fonctions *observer* introduites par la section 4.2 permettent d'obtenir l'ensemble des paramètres métier P à partir d'une géométrie G. Cette opération peut être vue comme un *problème direct*.

Dans cette thèse, notre but est de piloter les déformations de l'objet avec l'ensemble des paramètres métier. Le squelette de l'objet est modifié en fonction de ces paramètres pour obtenir une nouvelle géométrie G'.

Un ensemble de nouvelles valeurs des paramètres métier P' est fixé pour les sections et la génératrice. Le but est de déterminer les courbes du squelette correspondant à ces nouvelles valeurs.

Il s'agit donc de traiter le *problème inverse* c'est-à-dire de déterminer la géométrie G' associée à un ensemble de paramètres P'.

Nous proposons ici une méthode pour déterminer et résoudre ce *problème inverse*. Cette méthode permet de :

- lier les déformations de l'objet aux valeurs des paramètres métier,
- s'assurer de la validité de la forme obtenue d'un point de vue architectural,
- réduire le nombre de degrés de liberté du problème d'optimisation de forme,
- introduire des variations de formes non-linéaires par rapport aux point de contrôle des courbes du squelette, et plus généralement par rapport aux points de contrôle de l'objet initial.

Pour déformer un objet, de nouvelles valeurs des paramètres métier doivent être données. L'utilisation de fonctions d'influence permet de générer des distributions complexes de valeur des paramètres le long de l'objet, et ainsi d'atteindre un grand nombre de formes possibles.

La méthode de déformation proposée, et donc la résolution du *problème inverse*, repose sur la construction d'un problème d'optimisation décrit dans la section 5.1. La technique de résolution numérique de ce problème est discutée en section 5.2, puis nous illustrons la méthode

avec plusieurs exemples.

5.1 Définition du problème

Pour déformer un objet, le but est de trouver la géométrie G correspondant à un ensemble de paramètres donnés P, défini comme *problème inverse*.

La fonction *observer* $\phi : G \longrightarrow P$ introduite en section 4.2 est définie comme la fonction associant un ensemble de paramètres métier à une géométrie. Notre but est de contrôler une forme à partir de ces paramètres. Cela revient à calculer la fonction correspondant au problème inverse $\phi^* : P \longrightarrow G$.

La géométrie G est décrite par un squelette composé de courbes B-Splines. Nous proposons une méthode qui va modifier les coordonnées des points de contrôle de ces courbes jusqu'à obtenir une géométrie correspondant aux paramètres P fixés.

Les coordonnées des points de contrôle sont alors solution d'un système de minimisation, dont les termes sont décrits ci-après.

La géométrie discrète, représentée par un nombre fini de courbes de section et une courbe génératrice est appelée ξ. ξ^0 désigne la géométrie initiale composée de la courbe génératrice et des courbes de section initiales.

La courbe B-Spline de la génératrice, appelée ξ_g, est paramétrisée par $t \in [0, 1]$. Ses points de contrôle sont notés $\mathbf{c}_g$.

La i^{ème} section, correspondant au paramètre t_i sur ξ_g est appelée ξ_i, pour $i = 1, \ldots, N$. Les courbes ξ_i sont paramétrisées par $s \in [0, 1]$ et leurs points de contrôle sont notés $\mathbf{c}_i = (\mathbf{c}_{i,0}, \ldots, \mathbf{c}_{i,M})$.

Les points de contrôle du squelette complet, notés $\mathbf{c}$ sont définis comme l'ensemble des points de contrôle de la génératrice et de toutes les sections. $\mathbf{c} = \{\mathbf{c}_g, \mathbf{c}_{i,m}, i = 0, ..., N; m = 0, ..., M\}$.

5.1.1 Terme de distances des paramètres

Le premier terme du système de minimisation mesure la distance entre la valeur des paramètres courants $\phi(\xi)$ et la valeur des paramètres cibles V :

$$E_{param} = \|\phi(\xi) - V\|^2 \tag{5.1.1}$$

Les fonctions *observer* de la génératrice et des sections sont indépendantes. E_{param} est donc la somme des termes d'erreur de la génératrice et des sections. Le terme d'erreur correspondant à la i^{ème} section (respectivement courbe génératrice) est noté $E_{param,i} = \|\phi_i(\xi_i) - V_i\|^2$ (respectivement $E_{param,g} = \|\phi_g(\xi_g) - V_g\|^2$).

La fonction *observer* ϕ est non linéaire sur les coordonnées des points de contrôle $\mathbf{c}$. Par exemple, comme décrit dans la figure 5.6, le paramètre de hauteur d'un profil est déterminé en cherchant la valeur du paramètre s pour lequel la tangente de la courbe ξ_i est parallèle à une

direction donnée.

Le terme de distances des paramètres est donc non linéaire.

5.1.2 Terme de consistance de forme

Ce terme est introduit pour assurer la consistance de l'objet en mesurant la distance des courbes de section courantes ou de la génératrice courante avec les courbes originales.

Avant de commencer la résolution du problème de minimisation qui réalise la déformation, les courbes originales sont modifiées avec des transformations affines, permettant de "rapprocher" la géométrie des paramètres cibles.

Par exemple, ces transformations peuvent être des homothéties ou des rotations pour atteindre des longueurs ou des angles donnés.

Nous introduisons aussi des pré-transformations non linéaires, dépendant explicitement des paramètres cibles V. Par exemple, la modification de la hauteur d'un profil.

Ces transformations de la géométrie initiale, appelées D_V^i (ou D_V^g), sont explicitement calculées à partir de ξ_i^0 (ou ξ_g^0). Elles permettent de trouver une géométrie correspondant aux paramètres cibles plus rapidement et plus facilement pour de grandes déformations.

C'est la géométrie pré-transformée $D_V^i(\xi_i^0)$ (ou $D_V^g(\xi_g^0)$) qui est utilisée comme point de départ de l'algorithme d'optimisation.

Le terme de consistance de forme est donc défini comme :

$$E_{shape,i} = \|\xi_i - D_V^i(\xi_i^0)\|^2 \tag{5.1.2}$$

respectivement pour la courbe génératrice : $E_{shape,g} = \|\xi_g - D_V^g(\xi_g^0)\|^2$.

De part la définition de D_V^i (respectivement D_V^g), le terme de de consistance de forme est non linéaire.

5.1.3 Terme de contraintes métier

Ce terme permet de prendre en compte des contraintes architecturales F_k sur l'objet étudié. Il s'agit en général de contraintes de position ou de tangence, à déterminer au cas par cas en fonction de l'objet étudié.

Ces contraintes sont définies pour chaque section $\xi_i, i = 1, ..., N$ indépendam ment et pour la génératrice ξ_g.

Par exemple, un profil doit avoir une connexion lisse entre l'extrados et l'intrados au bord d'attaque, assurée par une contrainte de position et de tangence. Les coordonnées du point de contrôle au bord d'attaque pour l'extrados et l'intrados doivent être identiques, et les tangentes à ces deux points doivent avoir une direction opposée, perpendiculaire à la corde.

Pour simplifier l'expression de la contrainte, nous pouvons fixer la position du bord d'attaque par rapport au profil initial. Le bord d'attaque est paramétrisé comme étant à $s = 0$ sur les courbes de section.

Les contraintes métier F_k pour ce cas peuvent se définir de façon identique pour l'extrados et l'intrados comme :

$$\begin{aligned} F_0 :&\xi_i(s=0) - \xi_i^0(s=0) = 0 \\ F_1 :&\frac{\partial \xi_i}{\partial s} \cdot \overrightarrow{chord} = 0 \\ F_2 :&signe(\frac{\partial \xi_i}{\partial s} \times \overrightarrow{chord}) = signe(\frac{\partial \xi_i^0}{\partial s} \times \overrightarrow{chord}) \end{aligned}$$

Les contraintes sont quadratiques en les points de contrôle $\mathbf{c}$, mais nous pouvons envisager des contraintes plus générales qui ne sont *a priori* pas linéaires ou quadratiques. Le terme de contraintes métier est donc considéré comme non-linéaire.

5.1.4 Terme de lissage

Ce terme permet de contrôler le lissage global des courbes déformées. Cette définition est similaire à celle du terme correctif de l'approximation par courbes B-Splines, introduit dans la section 3.2.5. Ce terme inclut l'énergie de la courbe dans la fonctionnelle à minimiser. L'énergie de la courbe se traduit généralement par une combinaison des énergies :

$$H_0(\xi_i) = \frac{1}{2} \sum_{m=1}^{M} \|\Delta c_m\|^2, \qquad \Delta c_m = c_m - c_{m-1} \tag{5.1.3}$$

$$H_1(\xi_i) = \frac{1}{2} \sum_{m=2}^{M} \|\Delta^2 c_m\|^2, \qquad \Delta^2 c_m = c_{m+1} - 2c_m + c_{m-1} \tag{5.1.4}$$

respectivement $H_0(\xi_g)$ et $H_1(\xi_g)$ pour la génératrice.

Le terme de lissage est quadratique en les points de contrôle $\mathbf{c}$.

5.1.5 Système complet

Finalement, le système de minimisation non-linéaire proposé est le suivant, découplé pour la génératrice ξ_g et les sections ξ_i :

$$\min_{\mathbf{c}_i} E_{param,i} \; + \varepsilon E_{shape,i} + \sum_k \lambda_k F_k^2(\mathbf{c}_i) + \sum_{l=0}^{1} \mu_l \, H_l(\mathbf{c}_i) \tag{5.1.5}$$

Respectivement, le système est résolu pour la génératrice en $\mathbf{c}_g$.

ε est un poids permettant de pondérer l'influence du terme de contrôle de consistance de forme. λ_i est le poids pour les contraintes de forme métier. μ_i est le poids pour les termes correctifs. λ_i et μ_i sont très petits, en général en dessous de 10^{-4}.

Si ε est trop grand, le système va converger vers une solution proche de la solution originale et ne sera pas en mesure de respecter les paramètres métier fixés, potentiellement éloignés des paramètres originaux.

ε peut être vu comme un coefficient de pénalisation, qui diminue à chaque itération pour que le point de départ soit dégradé, et que le système puisse converger vers une forme

correspondant aux paramètres architecturaux fixés. Évidement, si ε est trop petit, la forme obtenue peut être de mauvaise qualité d'un point de vue architectural, même si elle respecte les paramètres métier fixés. A chaque itération qui diminue la valeur de ε, la géométrie calculée à l'itération précédente est utilisée comme point de départ de la résolution du système (5.1.5).

La pré-transformation de la géométrie initiale D_V introduite dans la section 5.1.2 (terme de consistance de forme) permet de rapprocher la forme initiale de la forme cible. Nous rappelons que la géométrie pré-transformée $D_V^i(\xi_i^0)$ (ou $D_V^g(\xi_g^0)$) est utilisée comme premier point de départ de l'algorithme d'optimisation.

Cela revient à initialiser le système de minimisation avec un point qui n'est pas trop éloigné de la solution optimale. En pratique, cette initialisation proche d'un optimum est importante pour la convergence de l'algorithme numérique utilisé.

5.2 Résolution numérique

5.2.1 L'algorithme SQP

La méthode d'optimisation quadratique successive (OQS), ou Sequential Quadratic Programming (SQP) en anglais, est une approche particulièrement adaptée pour la résolution de problèmes d'optimisation non linéaires.

Le principe de l'algorithme SQP consiste à approcher le modèle initial par une suite de problèmes quadratiques [Nocedal and Wright, 2006]. L'intérêt de cette méthode réside dans la rapidité de convergence de ces sous problèmes locaux.

Problèmes avec contraintes d'égalité

Considérons le problème d'optimisation suivant :

$$\begin{cases} \min_{x\in\mathbb{R}^n} & f(x) \\ & h_i(x) = 0, \quad i = 0, ..., m \end{cases} \tag{5.2.1}$$

où $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ et $h_i : \mathbb{R}^n \longrightarrow \mathbb{R}, i = 0, ..., m$ sont des fonctions continues et au moins deux fois dérivables.
Notons $h : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ la fonction qui regroupe les contraintes h_i.

Définition 5.2.1. *Le Lagrangien associé au problème* (5.2.1) *est la fonction* $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \longrightarrow \mathbb{R}$ *définie par :*

$$\mathcal{L}(x, \lambda) = f(x) + \lambda^T h(x) = f(x) + \sum_{i=0}^{m} \lambda_i h_i(x) \tag{5.2.2}$$

où $\lambda \in \mathbb{R}^m$ *est le vecteur des multiplicateurs de Lagrange associé aux contraintes d'égalité* $h(x)$.

Notons $A(x)$ le vecteur Jacobien de h contenant les dérivées des contraintes d'égalité, $A(x) = [\nabla h_1(x), \nabla h_2(x), ..., \nabla h_m(x)]$.

Définition 5.2.2. *Les conditions d'optimalité du premier ordre Karush, Kuhn et Tucker (KKT) du problème* (5.2.1) *peuvent se réécrire sous la forme d'un système d'équation de taille $n+m$ et dont les inconnues sont x et λ [Nocedal and Wright, 2006] :*

$$F(x,\lambda) = \begin{pmatrix} \nabla f(x) - \lambda\, A(x)^T \\ h(x) \end{pmatrix} = 0 \tag{5.2.3}$$

Un couple (x^*, λ^*) qui vérifie (5.2.3) *est dit solution primale-duale, avec x^* la solution optimale du problème primal* (5.2.1) *et λ^* un multiplicateur optimal de* (5.2.2)*.*

La méthode SQP se base sur l'utilisation d'un algorithme de Newton pour résoudre le problème (5.2.3).

Le Jacobien de (5.2.3) s'écrit :

$$F'(x,\lambda) = \begin{pmatrix} \nabla^2_{xx}\mathcal{L}(x,\lambda) & -A(x)^T \\ A(x)^T & 0 \end{pmatrix} \tag{5.2.4}$$

où $\nabla^2_{xx}\mathcal{L}(x,\lambda)$ est le Hessien du Lagrangien.

Le pas de Newton à l'itération (x_k, λ_k) est donné par :

$$\begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} + \begin{pmatrix} d^x_k \\ d^\lambda_k \end{pmatrix} \tag{5.2.5}$$

où d^x_k et d^λ_k sont solution du système :

$$\begin{pmatrix} \nabla^2_{xx}\mathcal{L}(x,\lambda) & -A(x)^T \\ A(x)^T & 0 \end{pmatrix} \begin{pmatrix} d^x_k \\ d^\lambda_k \end{pmatrix} = \begin{pmatrix} -\nabla f(x) + \lambda\, A(x)^T \\ -h(x) \end{pmatrix} \tag{5.2.6}$$

L'itération de Newton est bien définie si la Jacobienne du système (5.2.6) n'est pas singulière. Cette propriété est vérifiée si x_k est proche de la solution x^* et si les hypothèses suivantes sont validées :

- $A(x)$ est de rang m
- $p^T\, \nabla^2_{xx}\mathcal{L}(x,\lambda)\, p > 0$, pour tout $p \neq 0$ tel que $A(x)p = 0$

Si ces hypothèses sont vérifiées, alors l'itération de Newton (5.2.6) revient à résoudre le problème de minimisation quadratique suivant :

$$\begin{cases} \min_{d^x \in \mathbb{R}^n} & f(x_k) + \nabla f(x_k)^T\, d^x + \frac{1}{2}(d^x)^T\, \nabla^2_{xx}\mathcal{L}(x_k,\lambda_k)\, d^x \\ & A(x_k)\, d + h(x_k) = 0 \end{cases} \tag{5.2.7}$$

Le système (5.2.7) admet une unique solution sous les hypothèses données, qui initialise les valeurs de l'itéré suivant (x_{k+1}, λ_{k+1}).

Problèmes avec contraintes d'inégalité

Considérons le problème d'optimisation suivant :

$$\begin{cases} \min_{x\in\mathbb{R}^n} & f(x) \\ & h_i(x) = 0, \quad i \in E \\ & h_i(x) \geq 0, \quad i \in I \end{cases} \tag{5.2.8}$$

En suivant les mêmes étapes que pour un problème sans contraintes, le sous problème quadratique correspondant à (5.2.8) à l'itération k est défini comme [Nocedal and Wright, 2006] :

$$\begin{cases} \min_{d^x\in\mathbb{R}^n} f(x_k) + \nabla f(x_k)^T\, d^x + \frac{1}{2}(d^x)^T\, \nabla^2_{xx}\mathcal{L}(x_k,\lambda_k)\, d^x \\ \nabla h_i(x_k)^T\, d + h_i(x_k) = 0,\ i \in E \\ \nabla h_i(x_k)^T\, d + h_i(x_k) \geq 0,\ i \in I \end{cases} \tag{5.2.9}$$

Les problèmes d'inégalités sont généralement traités avec des *contraintes actives*. Une contrainte $h_i(x)$ est dite active en x^* si $h_i(x^*) = 0$. $H(x^*)$ est l'ensemble des indices i où la contrainte $h_i(x)$ est active en x^* :

$$H(x^*) = \{i,\ h_i(x^*) = 0\}$$

Alors x^* est une solution de (5.2.9) si et seulement si x^* est une solution de :

$$\begin{cases} \min_{d^x\in\mathbb{R}^n} f(x_k) + \nabla f(x_k)^T\, d^x + \frac{1}{2}(d^x)^T\, \nabla^2_{xx}\mathcal{L}(x_k,\lambda_k)\, d^x \\ \nabla h_i(x_k)^T\, d + h_i(x_k) = 0,\ i \in H(x^*) \end{cases} \tag{5.2.10}$$

[Nocedal and Wright, 2006] donne les conditions pour que l'ensemble des contraintes actives $H(x^*)$ du problème (5.2.8) soit conservé par l'approximation quadratique (5.2.9). Il faut notamment que (x_k, λ_k) soient assez proches de la solution optimale (x^{**}, λ^{**}) de (5.2.8).

[Nocedal and Wright, 2006] décrit aussi des techniques efficaces pour résoudre le problème (5.2.9) par la méthode des contraintes actives.

5.2.2 Application au problème de minimisation pour la déformation

Approche avec SQP

Le problème de minimisation (5.1.5) qui définit la déformation du squelette est un problème non-linéaire a priori sans contraintes.

En pratique, les limites de variation des coordonnées de points de contrôle sont définies comme contraintes d'inégalité sur le système. Ces limites sont imposées au cas par cas en fonction de l'objet, et empêchent d'obtenir des formes qui n'ont aucun sens architectural.

La méthode SQP décrite dans la section 5.2.1 est bien adaptée pour résoudre le problème (5.1.5).

Nous utilisons l'algorithme SQP disponible dans la Toolbox "Optimization" de MATLAB, accessible avec la fonction *fmincon* [MATLAB, 2015]. Les matrices Jacobienne et Hessienne du

système sont calculées par différences finies.

Le problème (5.1.5) a un nombre relativement important de degrés de liberté (DDL). En général, les courbes de section et la génératrice sont planaires, chaque point de contrôle a donc 2 DDL. Pour une géométrie avec s courbes de sections composées de m points de contrôle chacune et une courbe génératrice composée de n points de contrôle, le problème de déformation consiste en s problèmes de $2 \times m$ DDL et un problème de $2 \times n$ DDL.

La géométrie initiale pré-transformée $D_V^i(\xi_i^0)$ est le point de départ de l'algorithme SQP. Une valeur de ε est choisie, en général $\varepsilon = 1$. Puis, un système itératif est mis en place, avec une diminution de la valeur de ε à chaque itération. L'ensemble des courbes obtenues à l'itération n est utilisé comme point de départ de l'itération $n + 1$.

L'algorithme s'arrête lorsque la valeur de l'erreur globale calculée par (5.1.5) atteint un seuil fixé.

Remarque

Le calcul direct des dérivées du système n'étant pas possible, nous utilisons les différences finies pour obtenir une approximation. Nous avons également essayé la différentiation automatique afin d'obtenir la dérivée des fonctionnelles écrites en code MATLAB. Le logiciel "Automatic Differentiation for MATLAB (ADiMat)" [Coleman and Verma, 2000] a été utilisé pour obtenir les codes dérivés.

Avec cette technique, nous avons obtenus des résultats d'optimisation similaires à ceux obtenus avec les différences finies, mais avec un temps d'exécution beaucoup plus long. Par exemple, le tableau 5.1 présente les résultats des temps d'exécution en secondes pour une même déformation d'un profil donné.

SQP avec différences finies (sec.)	SQP avec différentiation automatique (sec.)	Différence
3	58	+1933 %

TABLE 5.1 – Temps d'exécutions pour la déformation d'un profil avec différences finies ou différentiation automatique

Méthode *"pas à pas"*

Les grandes déformations peuvent être difficile à atteindre, même à partir de la géométrie pré-transformée $D_V^i(\xi_i^0)$. Pour résoudre ce problème, nous introduisons l'idée de la méthode *"pas à pas"*, permettant d'atteindre l'objet cible petit à petit à partir de l'objet initial.

Le principe de cette méthode consiste à obtenir des géométries successives correspondant à de plus petites variations des paramètres cibles jusqu'à converger vers la géométrie cible.

Cela revient à se déplacer le long d'une courbe regroupant les formes d'énergie minimal dans le sens (5.1.5) dans l'espace des formes possibles.

En pratique, il est possible par exemple d'échantillonner N valeurs du vecteur de paramètres entre les paramètres originaux et les paramètres cibles. A chaque itération de la méthode *"pas à pas"*, le but est d'obtenir la géométrie correspondant à la fraction du vecteur de paramètres, avec

la méthode décrite précédemment. La géométrie obtenue est utilisée pour initialiser l'itération suivante.

Soit $V_0 = [v_{0,1}, ..., v_{0,p}]$ le vecteur de paramètres de la géométrie G_0. Soit $V_F = [v_{F,1}, ..., v_{F,p}]$ le vecteur de paramètres de la géométrie cible G_F. Fixons N itérations pour la méthode *"pas à pas"*, pour passer de G_0 à G_F. Alors, l'itération n de la méthode *"pas à pas"* calcule la géométrie $G_n,\ n = 1, ..., N$, avec comme point de départ la géométrie G_{n-1}, et le vecteur cible à atteindre V_n défini par :

$$V_n = [v_{n,1}, ..., v_{n,p}]$$
$$v_{n,i} : v_{0,i} < ... < v_{n-1,i} < v_{n,i} < v_{n+1,i} < ... < v_{F,i},\ i = 1, ..., p$$

5.2.3 Modification des repères locaux

Dans la section 4.1.2, nous avons introduit les repères locaux qui permettent de positionner et d'orienter une section sur la génératrice. Le repère local de la $i^{ème}$ section R_i est complété par un point d'attache P_i. Grâce à cette méthode, il est possible de définir la génératrice et les sections comme des courbes en 2D, puis de les replacer correctement les unes par rapport aux autres en 3D. Cela permet notamment de réduire le nombre de degrés de liberté du problème de déformation.

Lors du processus de déformation, la position et l'orientation relative de la génératrice et des sections peut être modifiée. Le point d'attache et le repère local associé à chaque section permet de la repositionner et de l'orienter correctement sur la génératrice après les déformations.

Appelons $C_g(t)$ la génératrice originale et $C'_g(t)$ la génératrice déformée.

Le nouveau point d'attache P'_i est facilement recalculé, étant donné qu'il correspond à un paramètre t_i sur la génératrice : $P'_i = C'_g(t_i)$.

L'un des axes du repère local R_i est la tangente T_i à la courbe génératrice au point d'attache correspondant au paramètre t_i. Lorsque la génératrice est déformée, une nouvelle tangente T'_i est calculée au paramètre t_i. La transformation (translation et rotation) f qui amène T_i sur T'_i est appliquée aux deux autres axes du repère local pour obtenir le nouveau repère local R'_i.

La figure 5.1 montre une déformation de la génératrice d'un foil. Les points d'attache originaux et les tangentes originales sont illustrés en bleu. Les nouveaux points d'attache sont illustrés en noir sur la nouvelle génératrice, et les tangentes recalculées en ces points sont en rouge.

La figure 5.2 montre les repères locaux complets, ceux originaux et ceux recalculés suite à la déformation.

La figure 5.3 illustre les anciens profils (bleu) et les nouveaux profils (rouge) repositionnés correctement grâce aux repères locaux.

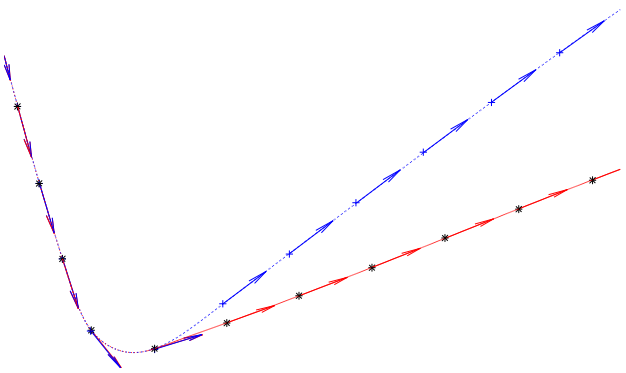


FIGURE 5.1 – Anciens et nouveaux points d'attache ; anciennes et nouvelles tangentes

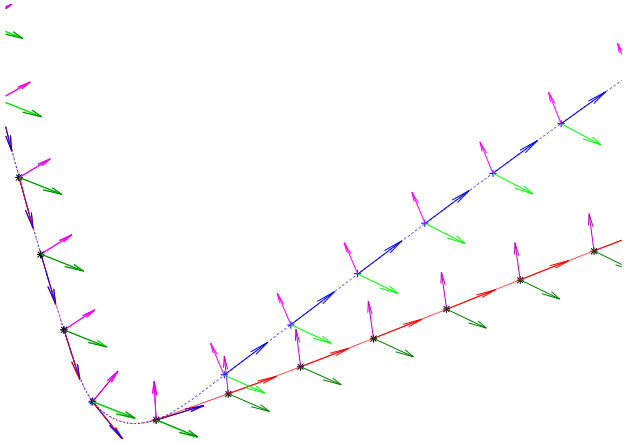


FIGURE 5.2 – Anciens et nouveaux repères

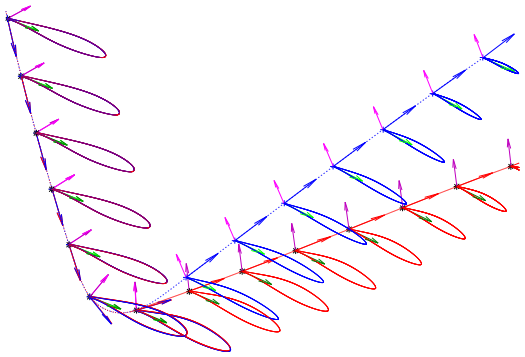


FIGURE 5.3 – Placement des sections avant et après déformation grâce au repères locaux

5.2.4 Courbes de répartition des paramètres

Dans la section 4.2.1, nous avons introduit les courbes de *répartition* qui approximent la répartition des paramètres le long des sections par des courbes B-Splines.

Ces courbes permettent de contrôler la valeur des paramètres métier des sections en utilisant moins de degrés de liberté.

Lors du processus de déformation, la courbe de répartition est modifiée par l'utilisateur ou l'algorithme d'optimisation. Les nouvelles valeurs des paramètres métier des sections sont calculées en projetant verticalement les points sur la nouvelle courbe. Ensuite, la déformation est effectuée comme décrit précédemment, en résolvant le problème de minimisation donné par l'équation (5.1.5).

La figure 5.4 montre la répartition du paramètre de corde des sections le long de la génératrice d'un foil. Nous proposons une déformation en modifiant les coordonnées des points de contrôle de cette courbe et calculons les nouvelles valeurs de la corde associées.

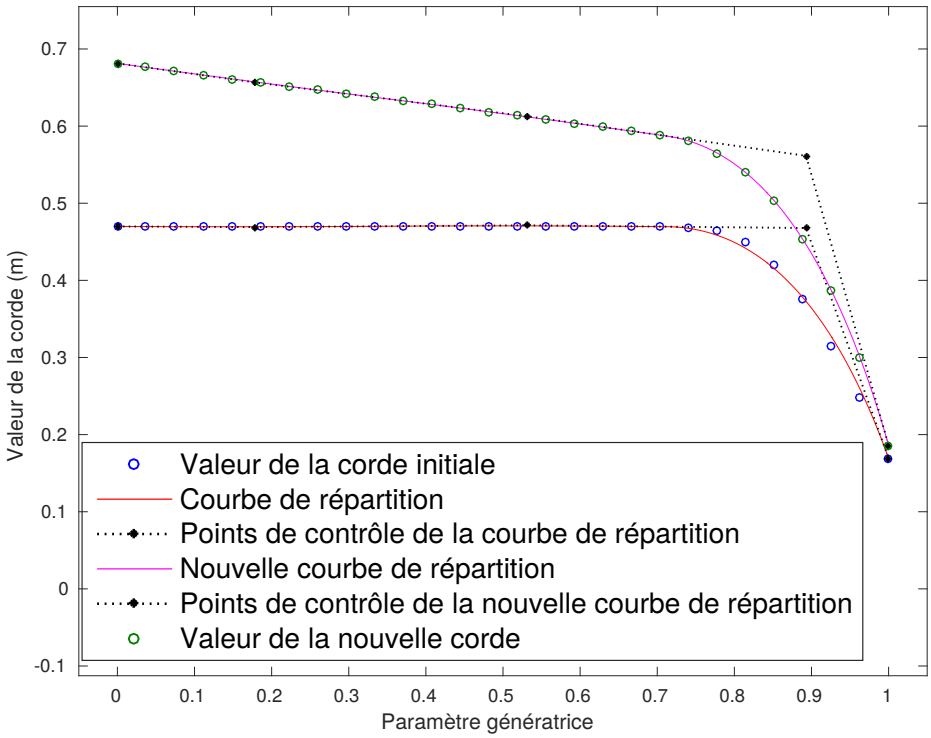


FIGURE 5.4 – Modification de la courbe de répartition de la corde le long du foil, nouvelles valeurs de la corde

5.2.5 Résumé de l'algorithme

La figure 5.5 résume l'algorithme de déformation et les différentes boucles itératives imbriquées. Cette partie du modeleur a été implémentée avec MATLAB. Une attention particulière a été portée à la généricité des modules développés, afin de pouvoir transposer facilement le code actuel en C++.

Nous rappelons ou introduisons les notations suivantes :

- $D_V(G)$ désigne la pré-transformation de la géométrie décrite dans la section 5.1.2.
- $V_0 = [v_{0,1}, ..., v_{0,p}]$ est le vecteur de paramètres de la géométrie G_0.
- $V_F = [v_{F,1}, ..., v_{F,p}]$ est le vecteur de paramètres de la géométrie cible G_F.
- ε est le poids permettant de pondérer l'influence du terme de contrôle de consistance de forme dans le système (5.1.5).
- h est un réel positif qui détermine la vitesse de décroissance de ε pour chaque itération. Dans la plupart des cas, $h = 2$.
- *seuil* est un seuil fixé pour la valeur de l'erreur totale donnée par le système (5.1.5). En général, $seuil = 10^{-6}$ ou $seuil = 10^{-7}$.

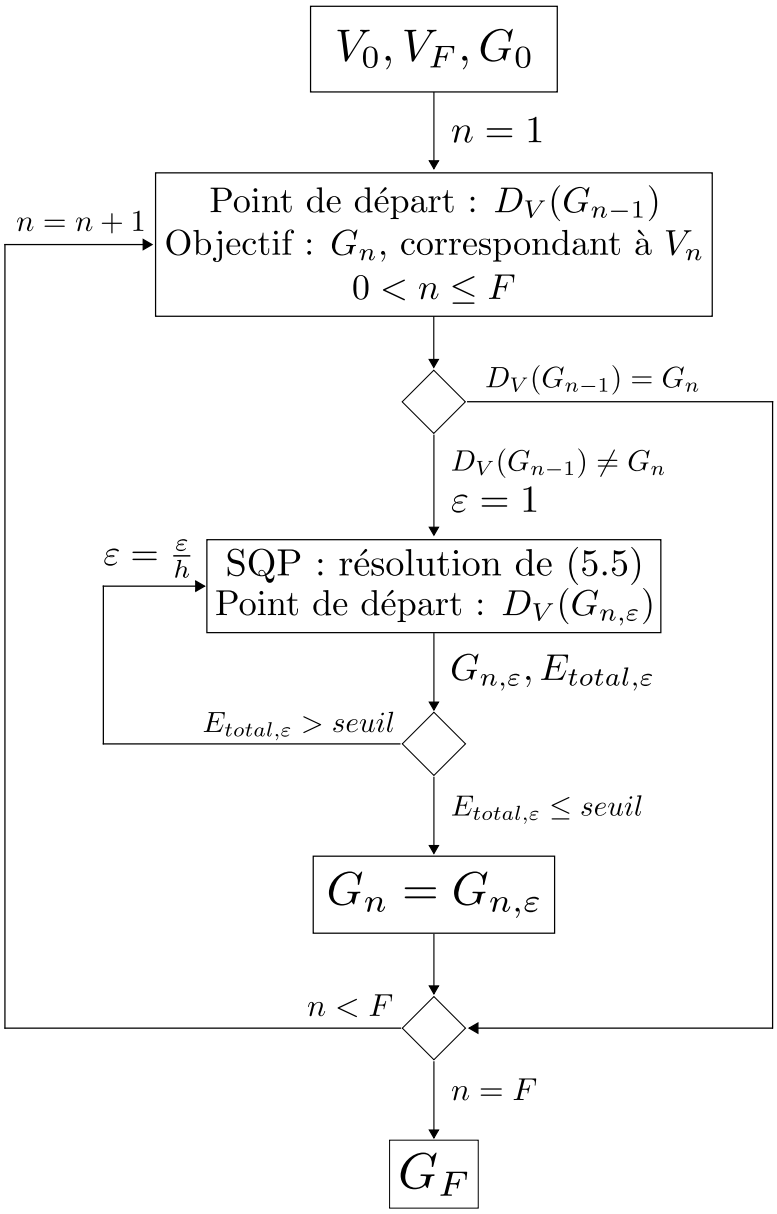


FIGURE 5.5 – Schéma de l'algorithme de déformation

5.3 Exemples de déformation

Exemple 1 - Modification d'un profil

Le but de ce premier exemple est de montrer les possibilités de déformation d'un profil. Le profil initial est une section de foil.

Les paramètres du profil sont illustrés par la Fig.5.6. Les figure 5.7 montrent des exemples de déformation possible à partir d'un profil existant.

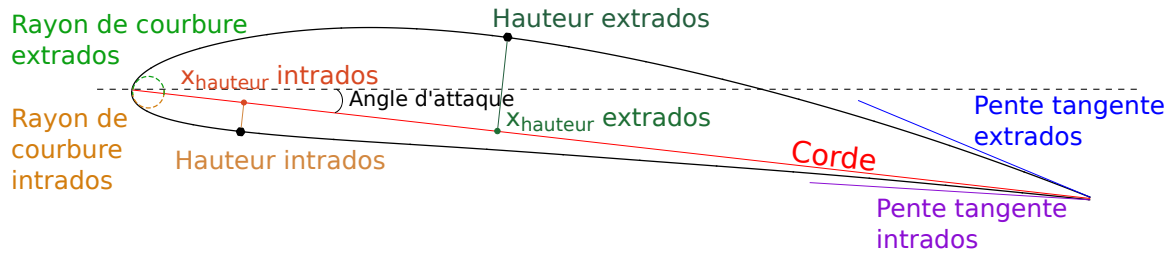


FIGURE 5.6 – Paramètres d'un profil

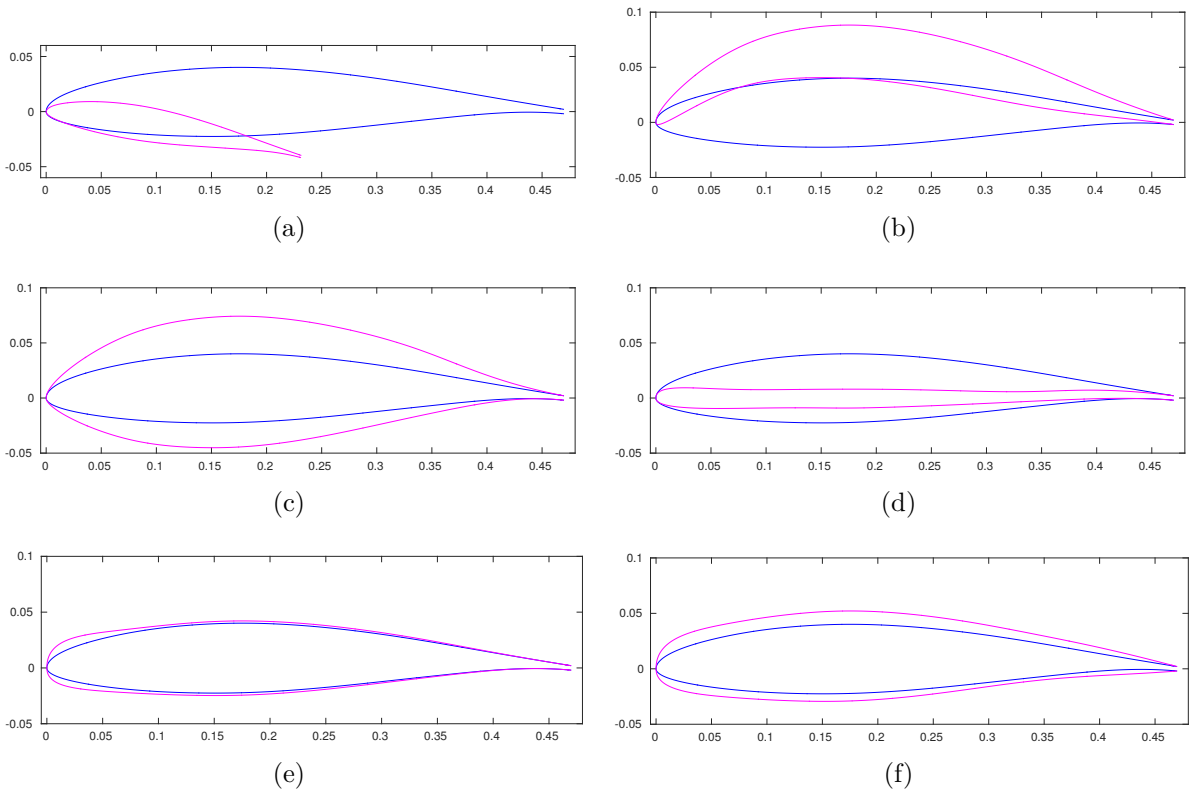


FIGURE 5.7 – Déformations d'un profil

Exemple 2 - Modification des hauteurs d'un profil

Petites déformations

Le but de cet exemple est de modifier uniquement la hauteur de l'extrados et de l'intrados d'un profil. La valeur des autres paramètres doit rester identique au profil original, notamment les rayons de courbure au bord d'attaque et les angles de tangence des bords de fuite doivent être fixes. Les variations de hauteurs sont de faible amplitude.

Le profil initial est un NACA4412. 10 points de contrôle sont utilisés pour décrire l'extrados, et 10 autres pour décrire l'intrados. Les coordonnées des points de contrôle du bord d'attaque et du bord de fuite de l'extrados et de l'intrados peuvent être pré-calculées exactement avec la valeur des paramètres de corde et d'angle d'attaque. Ces points ne sont donc pas inclus dans le système de minimisation. Nous résolvons donc deux problèmes de minimisation avec 16 degrés de liberté chacun.

L'erreur totale du système (5.1.5) pour l'extrados est $E_{total} = 9,8 \times 10^{-8}$. Pour l'intrados l'erreur est $E_{total} = 6 \times 10^{-5}$. La déformation des hauteurs est illustrée par la Fig.5.8.

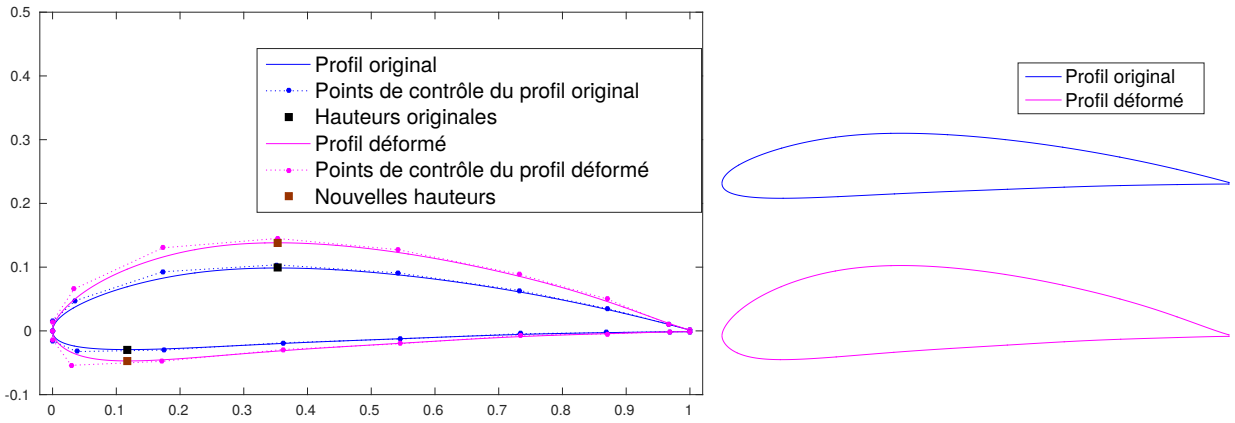


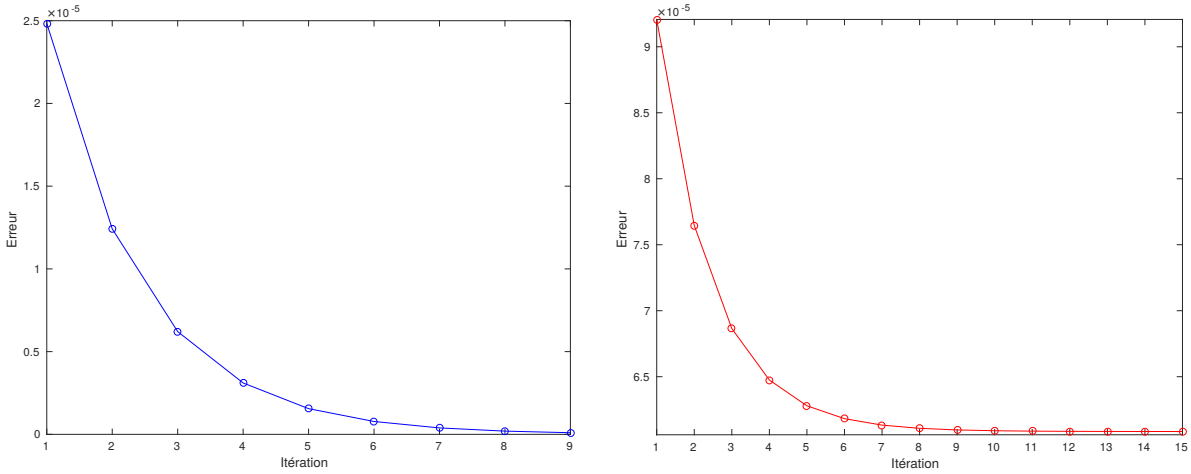
FIGURE 5.8 – Déformation des hauteurs d'un profil (NACA4412), les autres paramètres sont fixes

Pour cet exemple, nous n'avons pas utilisé la méthode *pas à pas*, c'est à dire que nous atteignons la géométrie finale G_F en 1 itération. Les courbes de convergence pour l'extrados et l'intrados sont présentées dans la figure 5.9. L'extrados converge en 9 itérations, et l'intrados en 15 itérations. A chaque itération, la valeur de ε diminue.

Cette déformation prend au total (extrados et intrados) 8 secondes.

Considérons la même déformation, mais utilisons la méthode pas à pas avec $N = 5$ itérations. Le profil obtenu est illustré par la figure 5.10. Cette fois, l'erreur totale du système (5.1.5) pour l'extrados est $E_{total} = 1,9 \times 10^{-7}$. Pour l'intrados l'erreur est $E_{total} = 2,6 \times 10^{-6}$. L'erreur sur la valeur des paramètres cible est plus petites que celle de la méthode directe, mais les deux méthodes convergent vers le même profil.

La déformation avec 5 itération de la méthode pas à pas prend au total (extrados et intrados) 35 secondes.



(a) Courbe de convergence de l'extrados (b) Courbe de convergence de l'intrados

FIGURE 5.9 – Courbes de convergence de l'algorithme pour la déformation du profil NACA4412

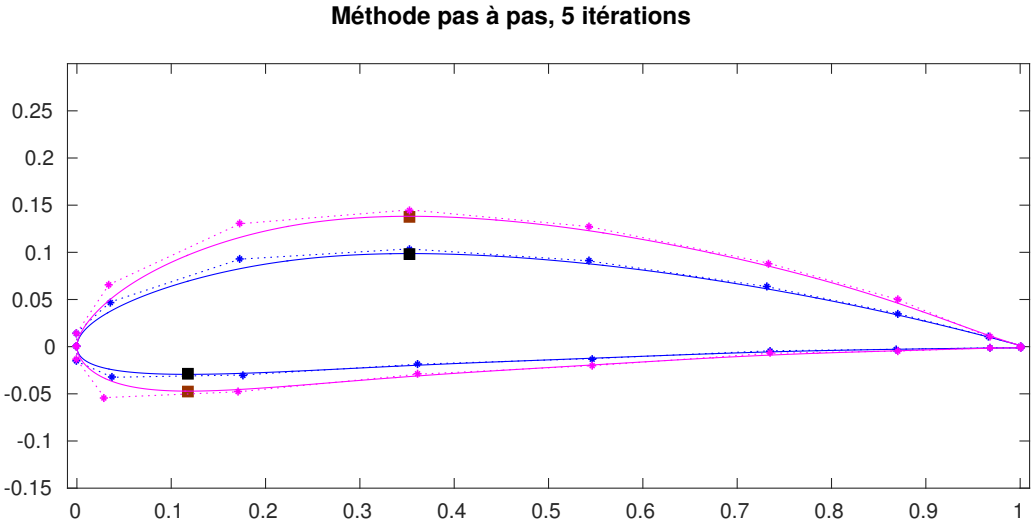


FIGURE 5.10 – Déformation des hauteurs d'un profil (NACA4412), les autres paramètres sont fixes, avec la méthode pas à pas en 5 itérations

Grandes déformations

Considérons à présent une déformation beaucoup plus importante de la hauteur, et modifions également la position en x de celle-ci. Les pentes des tangentes au bord de fuite sont augmentées. La méthode sans *pas à pas*, dite méthode directe, appliquée comme précédemment ne donne pas un bon résultat comme le montre la figure 5.11.

L'erreur totale du système (5.1.5) pour l'extrados est $E_{total} = 3,3 \times 10^{-4}$. Pour l'intrados l'erreur est $E_{total} = 4,6 \times 10^{-4}$.

La déformation permet d'atteindre les hauteurs et les positions en x cibles pour l'extrados, mais les paramètres cibles de l'intrados ne sont pas bien respectés.

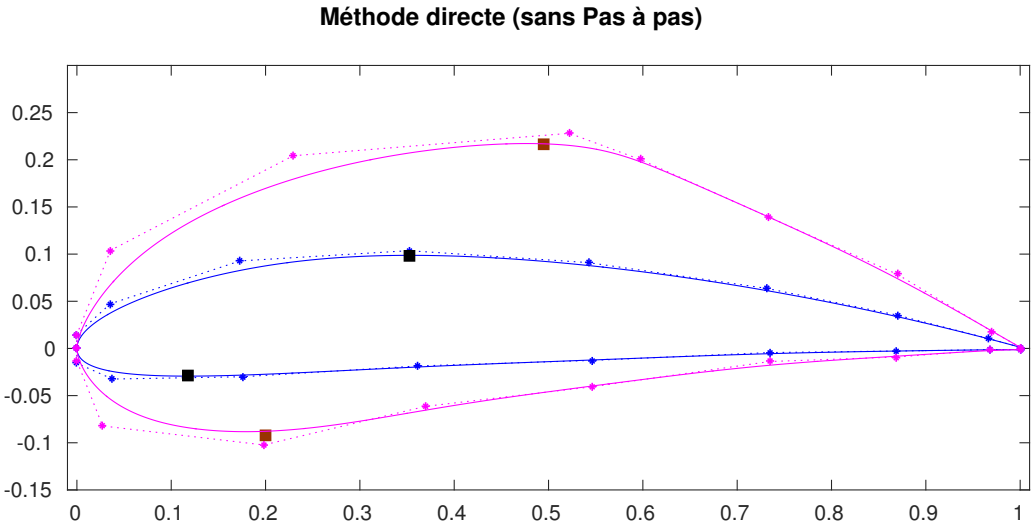


FIGURE 5.11 – Déformation de grande amplitude de la hauteur et de sa position en x sur un profil (NACA4412)

Appliquons la méthode *pas à pas*, avec un nombre de sous-itérations $N = 5$. 4 profils intermédiaires seront donc générés avant d'atteindre la valeur cible des paramètres. La figure 5.12 montre les profils intermédiaires et la figure 5.13 montre le profil final.

Avec $N = 5$ itérations, l'erreur totale finale du système (5.1.5) pour l'extrados est $E_{total} = 9,5 \times 10^{-4}$. Pour l'intrados, l'erreur est de $E_{total} = 4,2 \times 10^{-4}$. L'erreur est similaire à la méthode directe, mais répartie différemment. Le terme d'erreur des valeur des paramètres est plus élevé, mais le terme de consistance de forme est plus petit.

L'allure du profil final obtenu avec la méthode *pas à pas* est légèrement meilleure que celle du profil obtenu avec la méthode directe, on remarque surtout le positionnement des points de contrôle près de la hauteur maximal de l'extrados. Les paramètres cibles de l'intrados sont atteint, mais la position en x de la hauteur de l'extrados n'est un peu moins bien respectée, d'où l'erreur E_{total} plus grande.

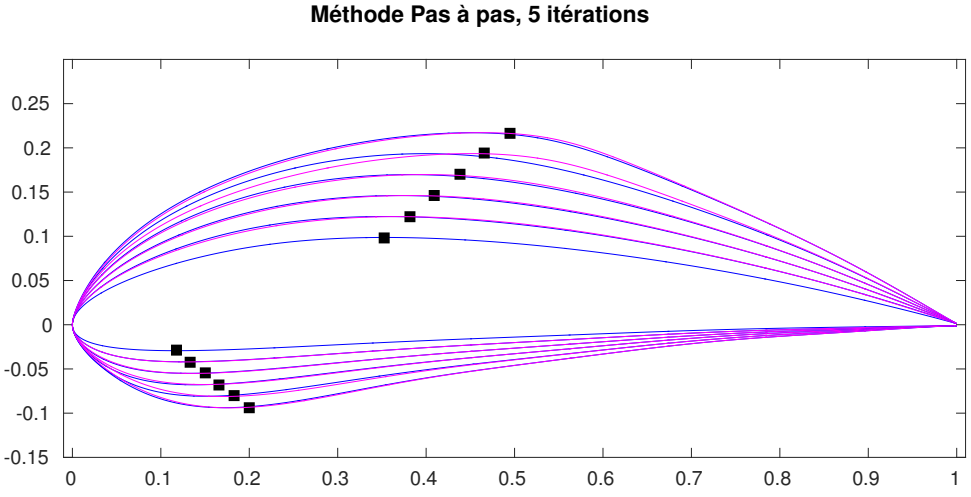


FIGURE 5.12 – Profils intermédiaires générés par la méthode *pas à pas*, avec 5 itérations

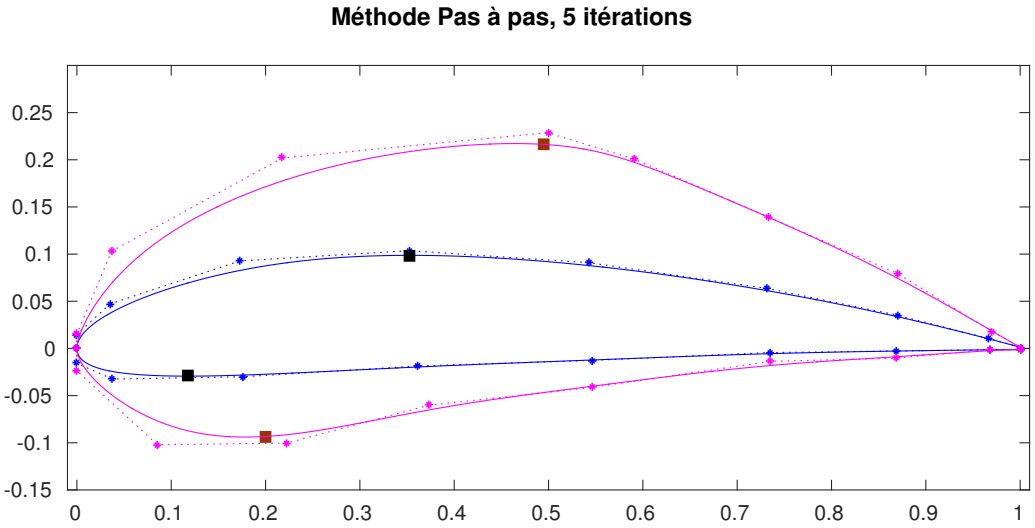


FIGURE 5.13 – Profil final obtenu avec la déformation *pas à pas*, avec 5 itérations

Avec $N = 20$ itérations, l'erreur totale finale du système (5.1.5) pour l'extrados est $E_{total} = 2,5 \times 10^{-4}$. Pour l'intrados, l'erreur est de $E_{total} = 1,1 \times 10^{-4}$. Dans ce cas, illustré dans la figure 5.14, les paramètres cibles sont atteint exactement pour l'extrados et l'intrados. La qualité de la forme est moins bonne, elle se rapproche de celle obtenue avec la méthode directe.

En augmentant le nombre d'itérations de la méthode *pas à pas*, l'erreur totale commise sur les paramètres diminue comme illustré sur les graphes de convergence des figures 5.15 et 5.16. En revanche, la qualité de la forme se dégrade à nouveau.

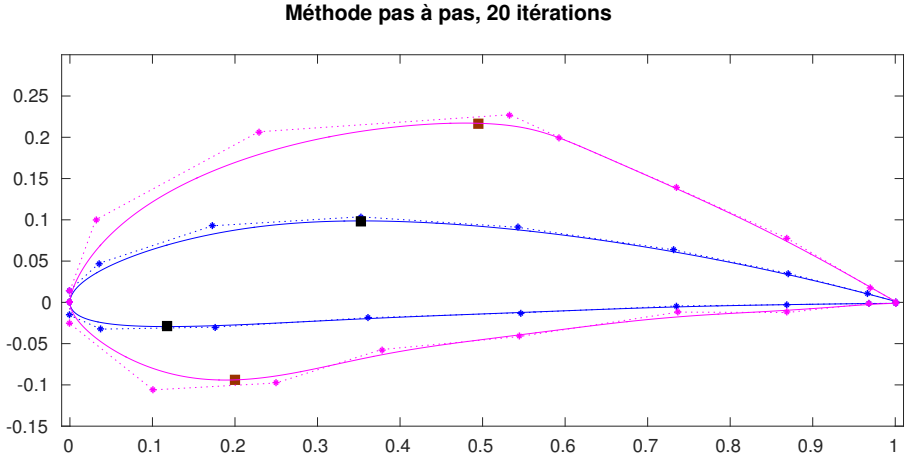


FIGURE 5.14 – Profil final obtenu avec la déformation *pas à pas*, avec 20 itérations

Cette méthode de déformation prend au total (extrados et intrados) 42 secondes pour $N = 5$ itérations de la méthode *pas à pas*, et $2,2$ minutes pour $N = 20$ itérations. La méthode directe prend $14,5$ secondes.

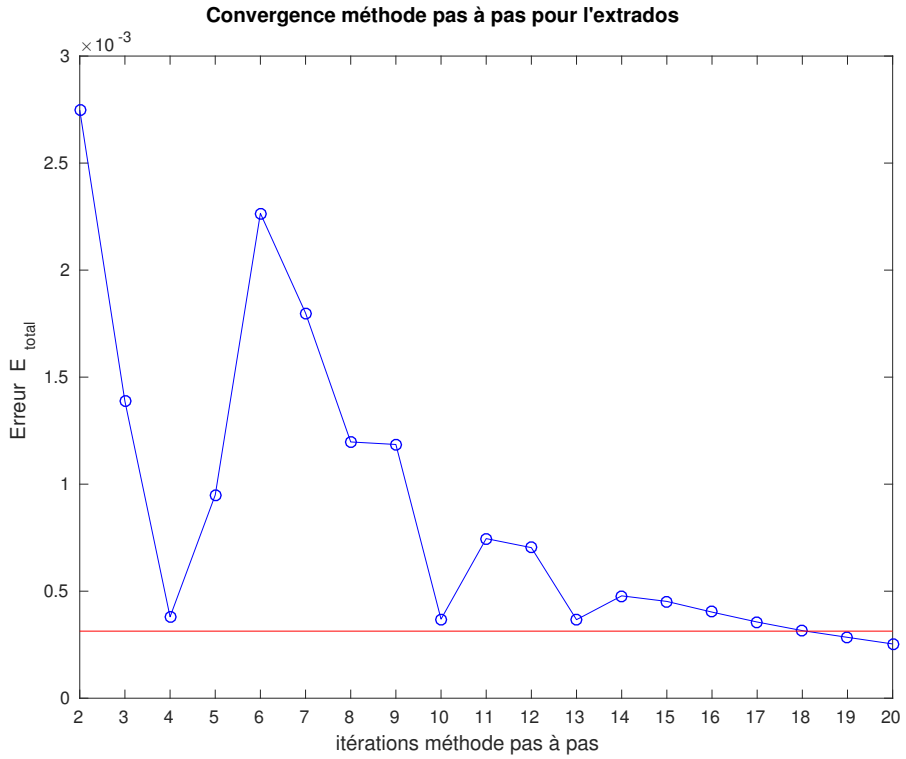


FIGURE 5.15 – Convergence de l'erreur E_{total} de la méthode pas à pas par rapport à la méthode directe pour l'extrados

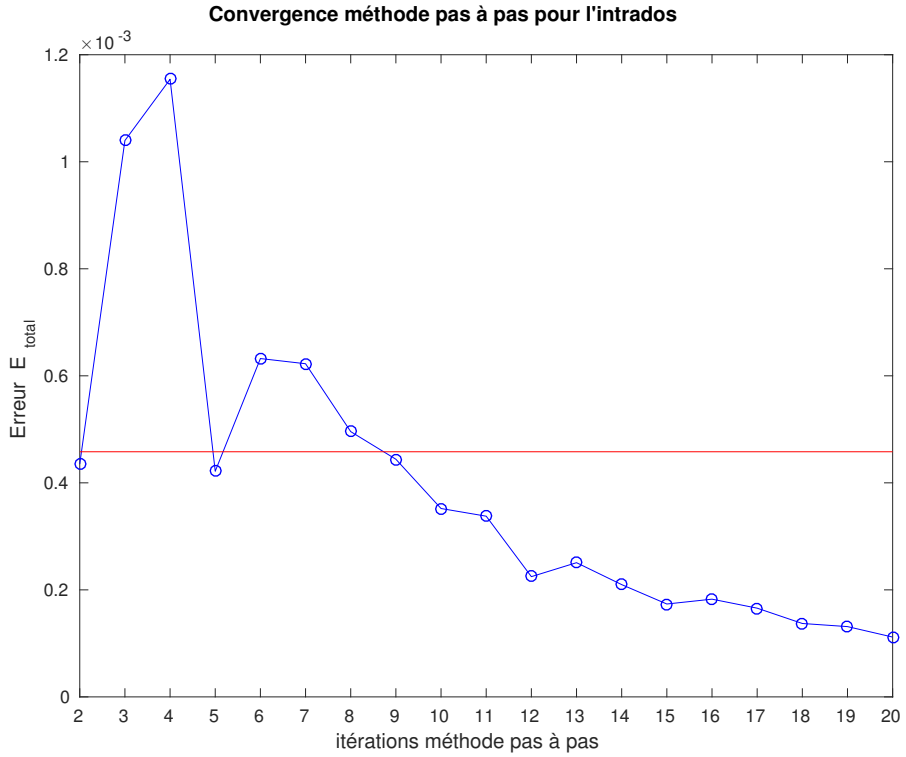


FIGURE 5.16 – Convergence de l'erreur E_{total} de la méthode pas à pas par rapport à la méthode directe pour l'intrados

Ces deux exemples de déformations (petites et grands) permettent de conclure que la fonction $\phi^* : P \longrightarrow G$ n'est que localement injective. Pour de petites déformations, la méthode pas à pas converge vers le résultat de la méthode directe. Donc pour un ensemble de paramètres P', ϕ^* donne une unique géométrie G' correspondante quelque soit la méthode utilisée pour l'atteindre.

En revanche pour de grandes déformations, nous obtenons plusieurs géométries G' à partir d'un même ensemble de paramètres P'. La solution du problème (5.1.5) n'est pas unique et dépend du nombre d'itérations utilisées dans la méthode *pas à pas*.

Exemple 3 - Modification de la longueur et de l'angle d'un foil

Dans cet exemple, nous modifions la longueur et l'angle d'un foil. Les paramètres du foil sont illustrés par la Fig.5.17 et son squelette est illustré par la Fig.5.18. Le but ici est d'augmenter la longueur du tip, et d'augmenter l'angle formé par le shaft et le tip. Le foil initial est un AC45.

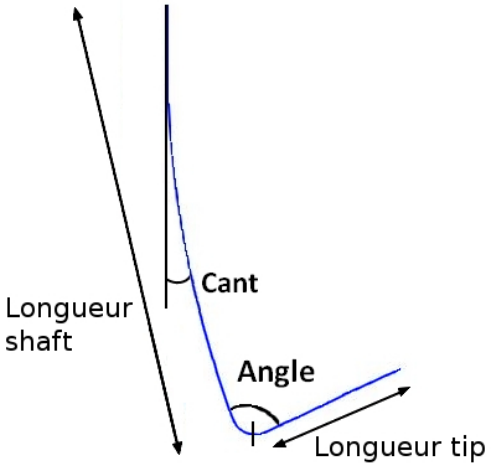


FIGURE 5.17 – Paramètres d'un foil

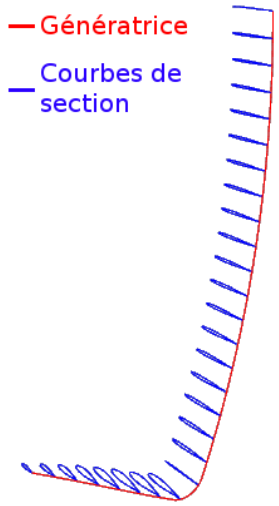


FIGURE 5.18 – Squelette du foil AC45

La déformation de la longueur du tip et de l'angle nécessite de modifier uniquement la génératrice. Les profils sont replacés à l'identique sur la génératrice déformée grâce aux repères locaux décrits dans la section 4.1.2 et 5.2.3.

11 points de contrôle sont utilisés pour décrire le shaft, et 7 autres pour décrire le tip. Les coordonnées des points de contrôle des extrémités de chaque courbe peuvent être pré-calculées exactement avec la valeur des paramètres de longueur et d'angle. Ces points ne sont donc pas inclus dans le système de minimisation. Nous résolvons donc deux problèmes, l'un avec 18 degrés de liberté (shaft) et l'autre avec 10 degrés de liberté (tip).

La Fig.5.8 illustre la déformation du foil.

Ces modifications sont atteintes directement par la géométrie pré-transformée $D_V^g(\xi_g)$, l'algorithme ne passe pas donc par la résolution du système avec SQP.

Réaliser cette déformation prend au total 5 secondes.

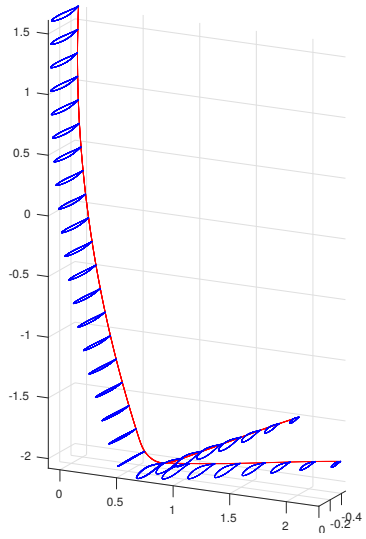


FIGURE 5.19 – Déformation de la longueur du tip et de l'angle d'un foil (AC45), les autres paramètres sont fixes

Exemple 4 - Modification de l'épaisseur le long d'un bulbe

Le but de cet exemple est d'inclure des fonctions d'influence dans le processus de déformation. Le bulbe d'un chalutier est décrit par son squelette composé de 22 sections et d'une génératrice, illustré par la figure 5.20. Seul la largeur des sections va être modifiée. Les paramètres des sections sont illustrés par la figure 5.21.

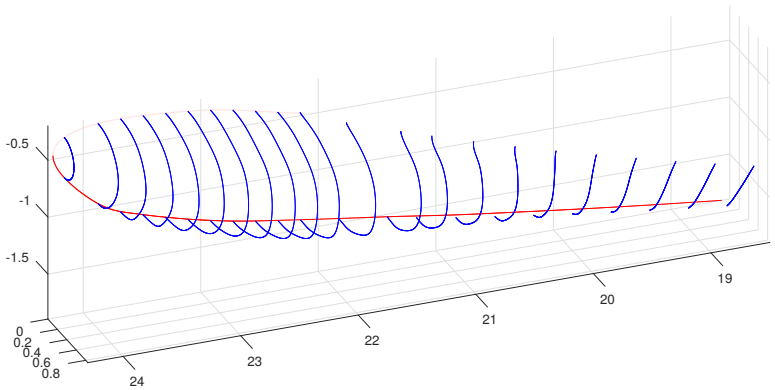


FIGURE 5.20 – Squelette d'un bulbe de chalutier

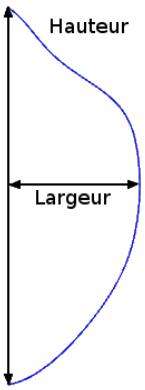


FIGURE 5.21 – Paramètres de la section d'un bulbe

La répartition de la largeur des sections le long de la génératrice est illustrée par la figure 5.22. C'est cette répartition que nous voulons modifier, conduisant à une déformation du bulbe. Le but est de modifier la répartition des largeurs suivant une fonction *d'influence*, illustrée en figure 5.23. Cette fonction est représentée par une courbe B-Spline C_{influ} adimentionnée sur $[0, 1]$. Les ordonnées correspondent aux paramètres de la génératrice, les abscisses correspondent à l'amplitude de déformation de la section au paramètre correspondant. Dans cet exemple la fonction d'influence est une courbe B-spline de degré 2.

La valeur du paramètre de largeur de chaque section est modifié en fonction de la fonction d'influence. Soit $L_{0,i}$ la valeur initiale de la largeur de la section i. La section est tout d'abord identifiée par le paramètre t_i le long de la génératrice auquel elle correspond. L'amplitude de déformation est ensuite déterminée en évaluant la fonction d'influence en t_i.

Finalement la largeur cible de la section i est donnée par :

$$L_i = L_{0,i} + C_{influ}(t_i)\, L_{0,i}$$

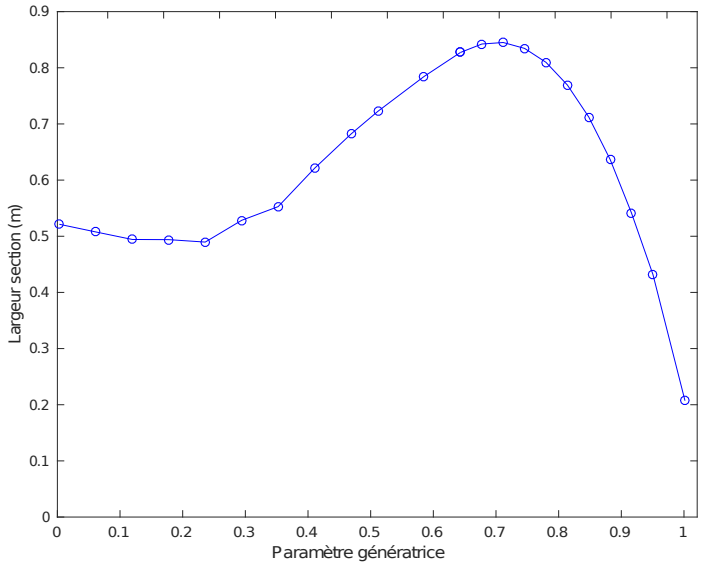


FIGURE 5.22 – Répartition de la largeur des sections le long de la génératrice

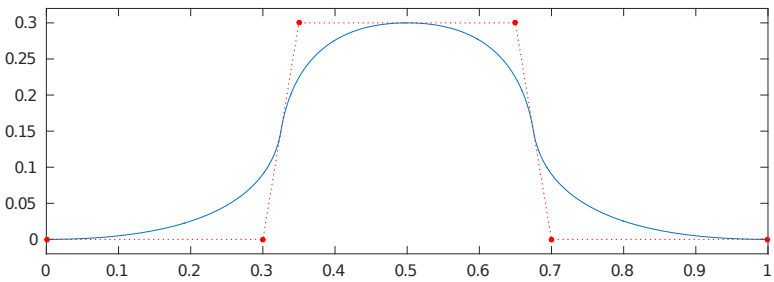


FIGURE 5.23 – Fonction d'influence représentée par une courbe B-spline adimensionnée sur $[0, 1]$

Chaque section est décrite avec 10 points de contrôle. La génératrice est décrite avec deux courbes, l'une de 10 points de contrôle et l'autre de 15 points de contrôle. Les coordonnées des points de contrôle des extrémités de chaque courbe peuvent être pré-calculées exactement avec la valeur du paramètre de hauteur. Ces points ne sont donc pas inclus dans le système de minimisation. La génératrice n'étant pas modifiée, nous résolvons donc 22 problèmes avec 16 degrés de liberté chacun.

Notons qu'il aurait aussi été possible d'utiliser des fonctions de *répartition* introduites dans la section 4.2.1 et 5.2.4. Avec de telles fonctions, la répartition montrée sur la figure 5.22 qui

contient 22 points aurait été approximée par une courbe B-Spline de 12 points de contrôle, comme illustré sur la figure 5.24. Utiliser ce type de fonction permet de réduire le nombre de paramètres liés aux sections pour l'optimisation de forme. Dans cet exemple, 12 paramètres sont utilisés au lieu de 22 pour contrôler la largeur des sections.

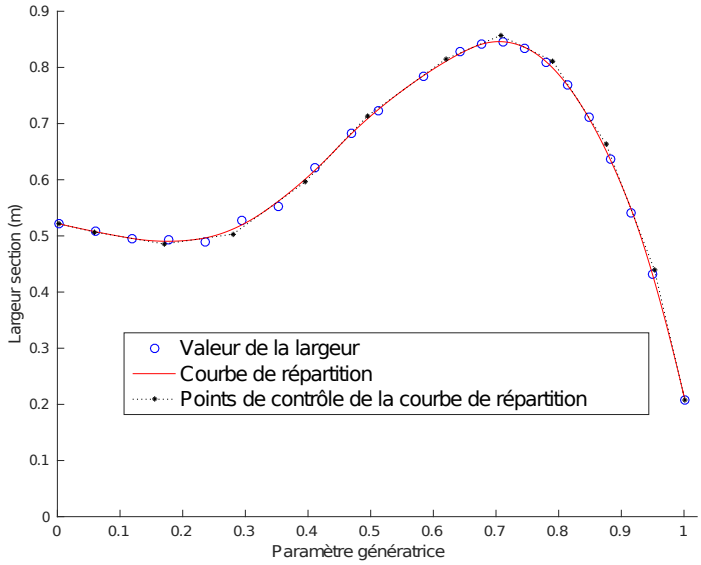


FIGURE 5.24 – Fonction de *répartition* de la largeur : courbe B-Spline approximant la répartition de la largeur des sections le long de la génératrice

La déformation finale du bulbe est illustrée par la figure 5.26, et la nouvelle répartition des largeurs par la figure 5.25. Réaliser cette déformation prend au total 47 secondes.

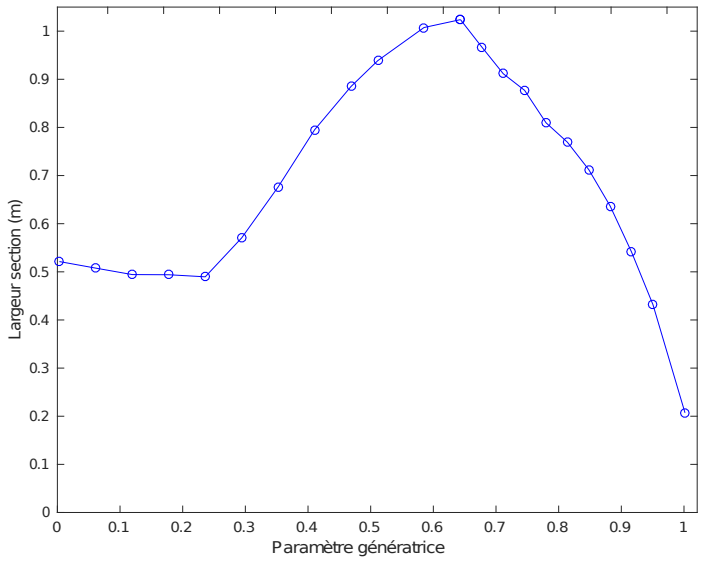


FIGURE 5.25 – Nouvelle répartition de la largeur des sections le long de la génératrice

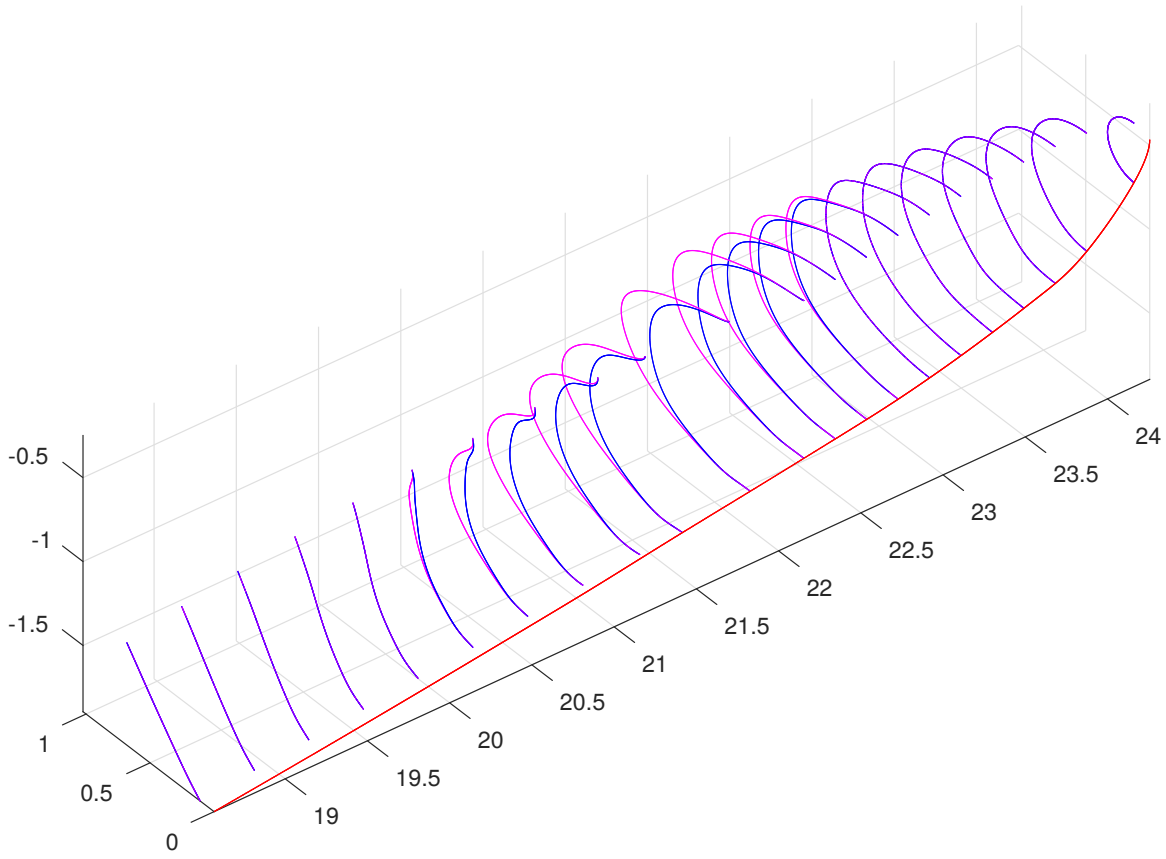


FIGURE 5.26 – Bulbe déformé

Chapitre 6

Reconstruction de surfaces

La méthode de déformation décrite au chapitre précédent produit un nouveau squelette définissant une géométrie déformée.

Ces techniques de reconstruction de surface peuvent aussi être utilisées pour obtenir une géométrie plus propre et plus simple que la géométrie originale, afin de faciliter la génération du maillage.

En effet, les mailleurs surfaciques ou volumiques utilisés pour des solveur numériques sont généralement très sensibles à la qualité des surfaces composant l'objet. La description de la connectivité des surfaces doit être précise et propre, et les surfaces doivent être les plus simples possibles. Par exemple, il s'agit d'éviter que deux surfaces adjacentes se chevauchent ou qu'elles ne soient pas jointives sur tout le long de l'arête commune. Les modèles géométriques ne respectant pas les contraintes du mailleur doivent être corrigés à la main. Cette intervention manuelle n'est pas souhaitable dès lors que l'on veut automatiser le processus d'optimisation de forme, comme décrit dans le chapitre 7.

Dans ce chapitre nous proposons plusieurs techniques de reconstruction de surfaces adaptées aux problèmes liés à la simulation numérique. La première, le *lofting*, est un algorithme rapide et adapté pour des surfaces simples. Pour des objets plus complexes, deux algorithmes sont proposés. Nous exposons d'abord une technique basée sur les *Surface Networks* qui permet de reconstruire des surfaces à partir de grilles de courbes. Puis un algorithme itératif basé sur les techniques de *Form Finding* et d'approximation de surface B-Spline qui permettent de reconstruire des patchs de surfaces continus à partir de nuages de points.

6.1 Méthode de *Lofting*

Le *lofting* est une technique classique pour reconstruire des surfaces à partir d'une série de courbes orientées dans une direction [Piegl and Tiller, 1997].

Nous rappelons brièvement la méthode de calcul d'une surface de type Loft.

Définition 6.1.1. *Soit un ensemble de r courbes dans une direction donnée avec le même degré, le même vecteur de nœuds et le même nombre $n+1$ de points de contrôle :*

$$\sigma_k(u) = \sum_{i=0}^{n} c_{i,k} B_i(u) \qquad u \in [0,1] \qquad k = 0, ..., r$$

Alors on construit $n+1$ courbes $\tilde{\sigma}(v)$ avec un vecteur de nœuds et un degré choisis :

$$\tilde{\sigma_l}(v) = \sum_{j=0}^{l} \tilde{c}_{j,l} B_j(v) \qquad v \in [0,1] \qquad l = 0, ..., n$$

telles que les courbes $\tilde{\sigma}_l$ interpolent les points de contrôle $c_{i,k}$ pour un $0 \leq i \leq n$ fixé.
L'ensemble des points de contrôle $c_{i,k}$ et $\tilde{c}_{j,l}$ des courbes σ_k et $\tilde{\sigma}_l$ constituent le réseau de points de contrôle de la surface loftée.

Les courbes $\tilde{\sigma_l}(v)$ sont construites avec les algorithmes d'approximation de courbe B-Spline décrits dans la section 3.2 afin d'interpoler les points de contrôle $c_{i,k}$ des courbes $\sigma_k(u)$. Il est possible d'inclure des contraintes de tangence aux extrémités des courbes, et ainsi générer des patchs de surfaces C^1.

La technique de *lofting* est un choix satisfaisant pour des géométries simples constituées de peu de surfaces. Le foil est un bon exemple : il est constitué de deux surfaces dont les contraintes de continuité sont entièrement satisfaites par les sections dans le système de minimisation (5.1.5).

La figure 6.1 montre la surface loftée obtenue pour un foil.

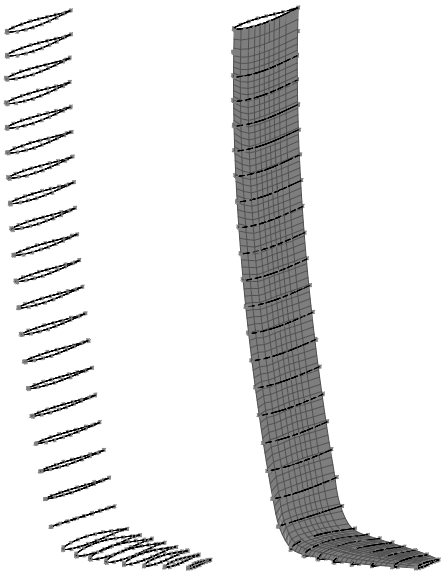


FIGURE 6.1 – Surface loftée obtenue pour l'extrados du foil, et les courbes associées

Par construction, le *lofting* permet de contrôler l'allure de la surface avec précision seulement sur une des deux directions. Pour des objets complexes cette technique n'est pas adaptée.

Comme le montre la figure 6.2 pour la surface loftée obtenue pour une partie du bulbe d'un chalutier, une discontinuité apparaît entre la surface fixe (en haut) et la surface loftée (en bas) : les bords des surfaces ne sont pas les mêmes et un "trou" apparaît dans la coque. Ce phénomène apparaît à cause de la discrétisation en section du squelette. En effet, aucune information ne décrit le comportement de la surface loftée entre deux sections, le long de la jonction avec la surface adjacente.

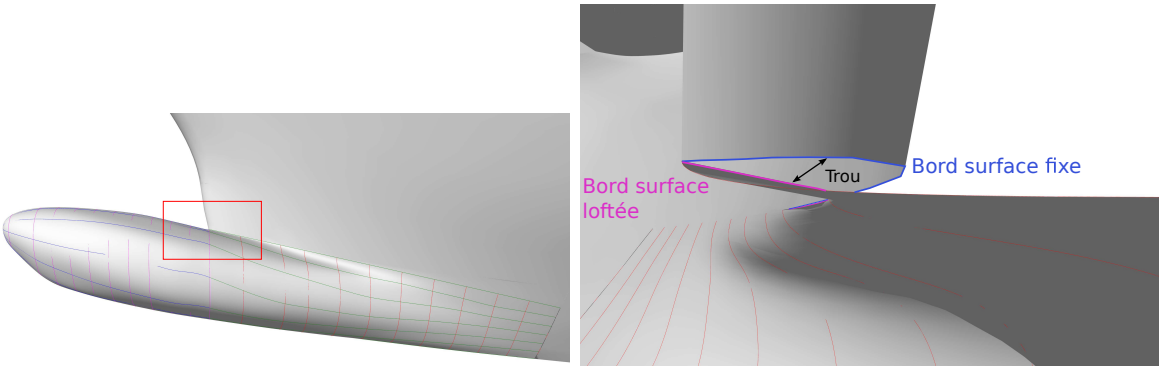


FIGURE 6.2 – Surface loftée obtenue pour une partie du bulbe, présentant une discontinué avec la surface adjacente

6.2 Méthode basée sur la technique de *Surface Network*

Pour des objets complexes, des techniques type multi-patchs sont nécessaires. La continuité entre les différents patchs doit être assurée, au moins G^1. La continuité G^1 signifie que deux surfaces adjacentes doivent avoir les même plans tangents le long du bord commun.

La première technique que nous présentons est basée sur la méthode de *Surface Network* [Piegl and Tiller, 1997]. Nous rappelons brièvement la formule de construction d'une surface avec la méthode *Surface Network*.

Définition 6.2.1. *Soit deux ensembles de courbes :*

$$\sigma_k(u) = \sum_{i=0}^{n} P_{i,k} B_{i,k}(u) \qquad u \in [0,1] \qquad k = 0, ..., r$$
$$\tilde{\sigma}_l(v) = \sum_{j=0}^{m} P_{j,l} B_{j,l}(v) \qquad v \in [0,1] \qquad l = 0, ..., s$$

tels qu'il existe des paramètres $0 = u_0 < u_1 < ... < u_{s-1} < u_s = 1$ *et* $0 = v_0 < v_1 < ... < v_{r-1} < v_r = 1$ *qui vérifient :*

$$Q_{l,k} = \sigma_k(u_l) = \tilde{\sigma}_l(v_k)$$

La surface qui satisfait :

$$S(u_l, v) = \sigma_l(v) \qquad l = 0, ..., s$$
$$S(u, v_k) = \sigma_k(u) \qquad k = 0, ..., r$$

est dite Surface Network *et elle est définie comme :*

$$S(u,v) = \sum_{l=0}^{s} \tilde{\sigma}_l(v)\phi_l(u) + \sum_{k=0}^{r} \sigma_k(u)\psi_k(v) + \sum_{l=0}^{s}\sum_{k=0}^{r} Q_{l,k}\phi_l(u)\phi_l(u) \tag{6.2.1}$$

avec :

$$\phi_l(u_i) = \begin{cases} 0 & if\ l \neq i \\ 1 & if\ l = i \end{cases} \quad et \quad \psi_k(v_i) = \begin{cases} 0 & if\ k \neq i \\ 1 & if\ k = i \end{cases}$$

Un exemple de surface construite par *Surface Network* et son réseau de courbes est illustré par la figure 6.3. La surface a été obtenue avec le logiciel de CAO Rhinoceros, avec la commande "Réseau de Courbes". Dans ce cas, les courbes σ_k (bleu) ont 5 points de contrôle, et $\tilde{\sigma_l}$ (rouge) ont 8 points de contrôle. La surface finale est composée de 96 points de contrôle.

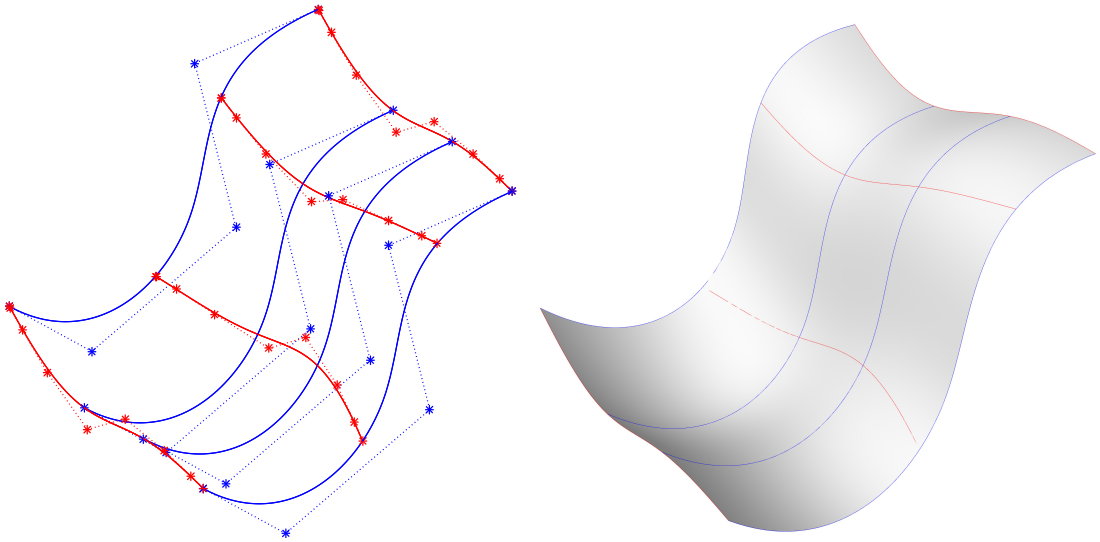


FIGURE 6.3 – Surface obtenue par *Surface Network* et la grille de courbes associée

Pour adapter cette technique à nos applications, nous proposons d'appliquer des contraintes aux courbes $\sigma_k(u)$ et $\tilde{\sigma_l}(v)$ composant la grille de la surface. Ces contraintes permettent de contrôler la forme générale de la surface et la continuité entre deux surfaces adjacentes.

Considérons les courbes de section du squelette comme les courbes σ_k de la grille dans la direction u. La génératrice du squelette correspond à l'une des courbes $\tilde{\sigma_l}$ dans la direction v. Les autres courbes $\tilde{\sigma_l}(v)$ sont construites par approximation de courbes B-Splines (voir section 3.2) à partir d'un échantillonnage régulier des courbes σ_k qui produit $l-1$ nuages de points.

Les courbes $\tilde{\sigma_l}(v)$ sont reconstruites avec un paramétrage uniforme (voir section 3.2.1) afin d'imposer une régularité forte aux courbes quelque soit la distribution des points le long des courbes de section.

Pour la gestion des contraintes de continuité G^1, il est important de noter deux configurations différentes :

1. les surfaces reconstruites adjacentes à des surfaces fixes du modèle original,
2. les surface reconstruites adjacentes à d'autres surfaces reconstruites.

Dans le premier cas, les contraintes de type G^1 sont issues de la surface fixe S_f et imposées aux extrémités des courbes $\sigma_k(u)$ et $\tilde{\sigma_l}(v)$ lors de leur construction. Pour les courbes de section σ_k, ces contraintes sont imposées dans le système de minimisation du processus de déformation (voir section 5.1.5). Lorsqu'une courbe σ_k correspond à un bord d'une surface fixe, le bord de cette surface est directement utilisé et la courbe n'est pas reconstruite par approximation. Pour les courbes $\tilde{\sigma_l}$, les contraintes sont imposées aux extrémités dans l'algorithme d'approximation par courbes B-Splines (voir section 3.2.6).

Par exemple, considérons n_{S_f} la normale à la surface fixe S_f au point où est échantillonné une des courbes $\tilde{\sigma}_l$. Alors pour satisfaire une contrainte de continuité G^1, nous imposons :

$$\tilde{\sigma}_l(v_0) \perp n_{S_f}$$

Dans le deuxième cas, les surfaces sont résolues les unes après les autres. Les contraintes pour une surface donnée sont issues des surfaces adjacentes reconstruites précédemment. Les surfaces adjacentes à des surfaces fixes sont toujours construites en premier.

Par exemple, considérons $n_{\tilde{\sigma}_{l_0,m-1}}$ la normale à l'extrémité libre de la courbe $\tilde{\sigma}_{l_0,m-1}$ construite à l'itération $m-1$. Pour satisfaire une contrainte de continuité G^1, la courbe l_0 à l'itération m doit vérifier :

$$\tilde{\sigma}_{l_0,m}(v_0) \perp n_{\tilde{\sigma}_{l_0,m-1}}$$

Cette méthode de reconstruction basée sur la technique classique de *Surface Network* permet d'appliquer facilement des contraintes issues de surfaces adjacentes. Elle est plus précise que le *lofting* car elle permet de contrôler directement les courbes dans les deux directions u et v.

Cependant, elle impose de choisir un ordre de reconstruction pour passer les contraintes de continuité d'une surface à l'autre. Aussi, il est impossible de prédire si l'allure de la courbe à la jonction est satisfaisante.

Enfin, le nombre de points de contrôle d'une surface obtenue par la technique de *Surface Network* dépend du nombre de points de contrôle des courbes de la grille.

Or pour s'assurer de la connectivité de la grille (intersections entre les courbes), un nombre important de points de contrôle est utilisé. Plus le nombre de points de contrôle est grand, plus l'algorithme d'approximation par courbes est précis.

L'échantillonnage des courbes de section σ_k joue un rôle important dans la régularité des courbes $\tilde{\sigma}_l$. Les courbes σ_k n'ont pas de paramétrisation commune, la répartition des points échantillonnés est donc arbitraire.

Les surfaces résultantes sont donc en général très complexes, avec un nombre important de points de contrôle. Elles sont peu maniables et leur régularité d'un point de vue architectural n'est pas satisfaisante. En effet, lorsqu'une surface est décrite avec beaucoup de points de contrôle, de faibles oscillations dans le réseau de points impactent fortement la lissage globale de la surface. De plus, les termes de lissage utilisé dans l'approximation de courbes B-Spline, introduits dans la section 3.2.5, ne représentent qu'une énergie de tension locale des points de contrôle.

Exemple

Un exemple de reconstruction de la surface d'un bulbe est illustré par la figure 6.4. Dans ce cas, il s'agit de reconstruire deux surfaces. La première surface (déterminée par les courbes en vert et rouge) a trois côtés adjacents à des surfaces fixes du reste de la coque. La deuxième surface (déterminée par les courbes en bleu et magenta) doit simplement se raccorder de façon continue à la première surface.

Notons qu'un point singulier est situé au sommet de la jonction des deux surfaces, montré dans la figure 6.5. A ce point, une surface fixe de la coque se raccorde aux deux surfaces

du bulbe. Les tangentes des courbes de bords des surfaces ne sont pas les mêmes, bien que coplanaires.

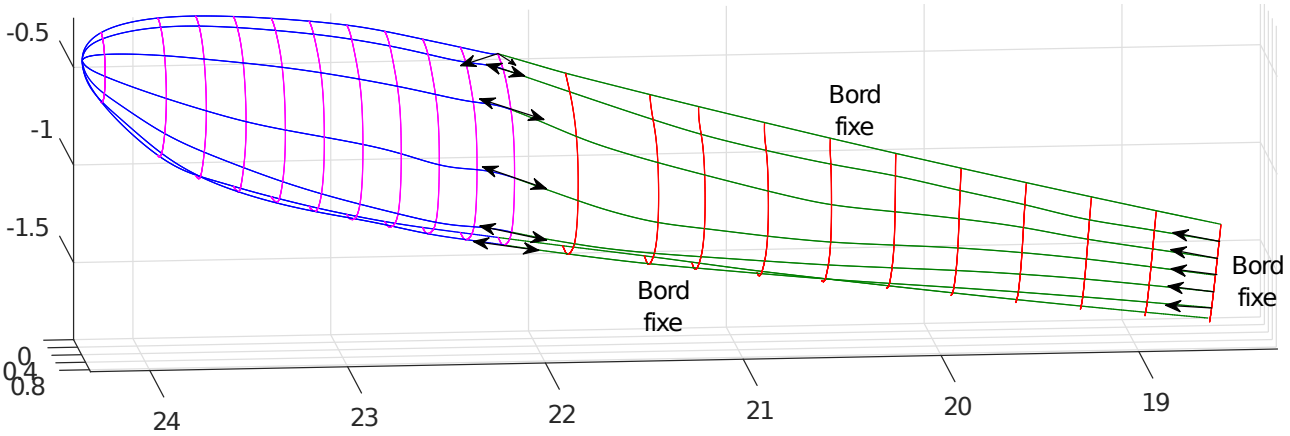


FIGURE 6.4 – Grille de courbes avec contraintes aux bords pour les surfaces reconstruites par *Surface Network*

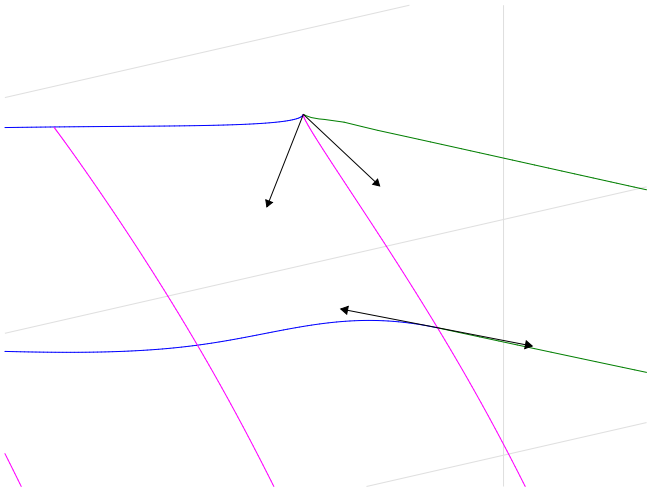


FIGURE 6.5 – Zoom sur le point singulier au sommet de la jonction entre les deux surfaces

Comme défini plus haut, les courbes σ_k sont les sections du squelette (en rouge et magenta). Elles sont définies avec 10 points de contrôle chacune. Les courbes σ_k contiennent déjà les contraintes de raccordement adéquates. En effet, ces contraintes sont prises en compte dans le système de déformation décrit au chapitre précédent.

Les courbes $\tilde{\sigma}_l$ (en vert et bleu) sont calculées avec un algorithme d'approximation par courbes B-Splines. Les contraintes de raccord sont intégrées lors de cette construction. Les flèches noires représentent les directions des contraintes de tangence aux extrémités des courbes σ_k. Pour les courbes $\tilde{\sigma}_l$ de la première surface (vert), 30 points de contrôle ont été utilisés. Les deux courbes des bords fixes (courbes du modèle original) contiennent respectivement 36 points de contrôle pour celle du haut et 10 points de contrôle pour celle du bas. Pour les courbes $\tilde{\sigma}_l$ de

la deuxième surface (bleu), 15 points de contrôle ont été utilisés.

Il est visible sur la figure 6.4 que les courbes $\tilde{\sigma}_l$ (en vert et bleu) ne sont pas régulières, et oscillent entre deux sections successives. C'est une conséquence de la méthode d'échantillonnage des courbes σ_k. Il est difficile de déterminer une règle *a priori* sur le choix de la répartition des points d'échantillonnage. Dans cet exemple, une échantillonnage régulier a été utilisé, c'est-à-dire que les points ont été choisis à intervalles réguliers de u sur les $\tilde{\sigma}_k(u)$.

Les surfaces résultantes sont illustrées par la figure 6.6. Les surfaces ont été obtenues avec le logiciel de CAO Rhinoceros, avec la commande "Réseau de Courbes". La première surface est composée de 220 x 548 points de contrôle. La deuxième surface est composée de 292 x 229 points de contrôle.

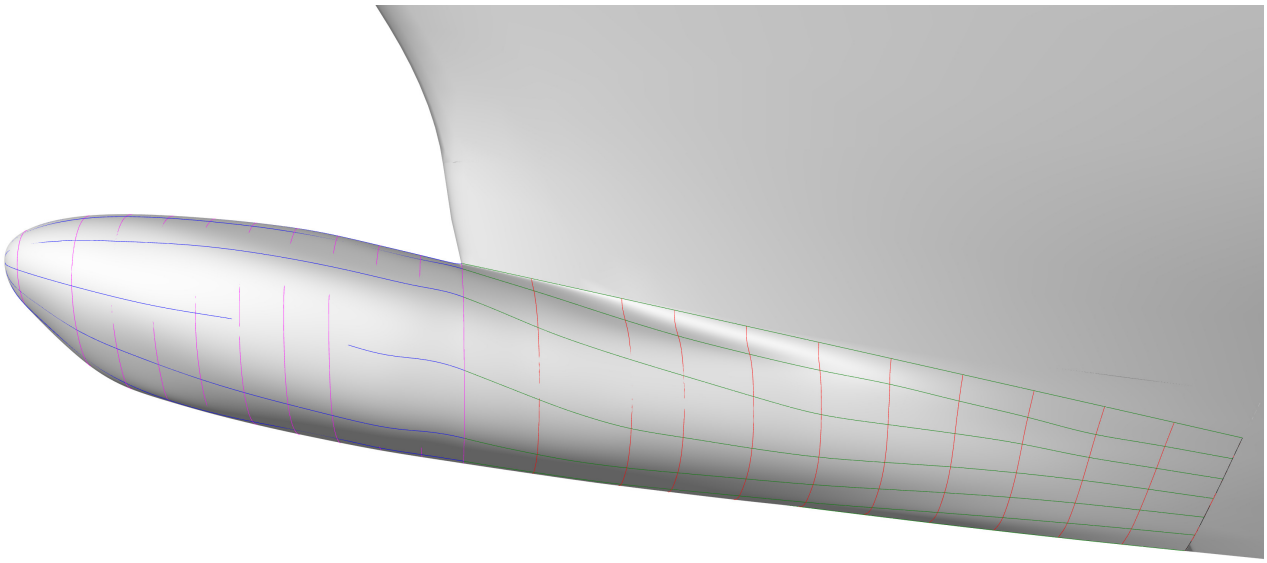


FIGURE 6.6 – Surfaces obtenues avec la méthode *Surface Network* à partir de la grille calculée dans la figure 6.4

6.3 Méthode *Form finding*

La deuxième technique que nous proposons pour générer des surfaces complexes avec des raccords continus est basée sur la technique appelée *form finding* [Tang et al., 2014]. Contrairement à la technique basée sur les *Surface Networks*, la technique de *form finding* permet de maîtriser le nombre de points de contrôle utilisés pour reconstruire les surfaces. Elle permet aussi de mieux contrôler la continuité entre les patchs.

6.3.1 Définition du problème

La technique de *form finding* que nous avons implémentée se base sur le travail de [Tang et al., 2014]. Dans cet article, il est question de trouver la meilleure distribution de faces d'un maillage polyédrique pour satisfaire des contraintes de type coplanarité des faces, passage par

un bord, volume total, lissage, etc. en réduisant le coût de la résolution du problème. [Tang et al., 2014] construit un système linéaire regroupant l'ensemble des contraintes, exprimées sous forme linéaire ou quadratique. La résolution du système se fait par un algorithme itératif.

Nous adaptons cette technique pour construire un réseau de points de contrôle correspondant aux surfaces qui envelopperont le squelette. Dans notre cas, les données initiales sont les courbes de section, la génératrice et les courbes de bords des surfaces fixes adjacentes. En échantillonnant ces courbes nous obtenons un nuage de points $P : \{P_l,\ l = 0, ..., N_P\}$, support des surfaces à reconstruire.

Afin de pouvoir appliquer une méthode de gestion des contraintes similaire à [Tang et al., 2014], il faut obtenir un premier réseau de points de contrôle, et donc une première surface à partir des points échantillonnés. Cela revient à réaliser une approximation de surface B-Spline, comme expliqué dans la section 3.3.

A partir de cette surface initiale, la surface finale $\sigma(u, v)$ est obtenue en résolvant un problème de minimisation sur les coordonnées de ses points de contrôle c_{ij}. Les contraintes de continuité et de lissage sont définies comme étant linéaires ou quadratiques et intégrées dans le système de minimisation.

Certaines contraintes que nous voulons intégrer dans le système ne sont pas linéaires ou quadratiques en les c_{ij}, comme par exemple les contraintes de continuité G^1 entre les patchs. Dans de tels cas, nous utilisons une autre caractéristique géométrique du problème pour établir une contrainte de forme quadratique. Pour la continuité G^1 entre patchs, les contraintes peuvent s'exprimer de façon quadratique sur les normales de la surface. Les normales sont donc ajoutées dans le vecteur des inconnues du système.

Nous décrivons chacune des contraintes que nous avons associée à notre problème ci-après.

Définition 6.3.1. *La forme générale considérée pour l'expression d'une contrainte quadratique est la suivante :*

$$\varphi_i(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T H_i \boldsymbol{x} + b_i^T \boldsymbol{x} + c_i = 0,\ i = 1, ..., N \tag{6.3.1}$$

où H_i est une matrice symétrique, b_i un vecteur et c_i une constante et $\boldsymbol{x}$ le vecteur d'inconnues sur lesquelles la contrainte s'applique.

Rappelons la définition introduite dans la section 3.3 de la fonction de paramétrage f qui associe à chaque point P_l du nuage de points un couple de paramètres (u_l, v_l) sur la surface, telle que :

$$\begin{aligned} &f : \mathbb{R}^3 \longrightarrow [0,1] \times [0,1] \\ &P_l \longrightarrow (u_l, v_l), \qquad\qquad l = 0, ..., N_P \end{aligned}$$

Définition 6.3.2. ***Fitting de la surface***
Cette contrainte permet de s'assurer que la surface $\sigma(u, v)$ passe par les N_P points P_l échantillonnés sur le squelette.

$$E_{fitting} : \sum_l \|\sigma(u_l, v_l) - P_l\|^2 = 0,\ l = 0, ..., N_P \tag{6.3.2}$$

$$avec\ \sigma(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} c_{ij}B_i(u)B_j(v)$$

Définition 6.3.3. ***Contraintes de tangence avec les bords fixes de l'objet***
Soit n_l la normale au point P_l sur la surface fixe adjacente à la surface qui est reconstruite. Dans la direction u, la contrainte de continuité C^1 s'exprime comme :

$$E_{fixT} :\ \langle \sigma_u(u_l, v_l) \cdot n_l \rangle^2 = 0,\ l = 0, ..., N_P \tag{6.3.3}$$
$$où\ \sigma_u(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} c_{ij}B'_i(u)B_j(v)$$

Respectivement, une contrainte similaire est définie pour la direction v.

Définition 6.3.4. ***Contraintes de tangence avec les bords mobiles de la surface***
Aux N_N points P_l situés sur la frontière entre deux surfaces reconstruites σ_1 et σ_2, les valeurs des normales n_k sont de nouvelles inconnues du système, qui doivent satisfaire des contraintes d'égalité. Dans la direction u, la contrainte de continuité C^1 s'exprime comme :

$$\begin{cases} E_{mobileT1} :\ \langle \sigma_{1\ u}(u_{1,l}, v_{1,l}) \cdot n_k \rangle^2 = 0 \\ E_{mobileT2} :\ \langle \sigma_{2\ u}(u_{2,l}, v_{2,l}) \cdot n_k \rangle^2 = 0 \end{cases} \tag{6.3.4}$$
$$l = 0, ..., N_P,\ k = 0, ..., N_N$$

respectivement pour la direction v.

Définition 6.3.5. *De plus, les normales n_k doivent satisfaire :*

$$E_{mobile\ normals} :\ \langle n_k \cdot n_k \rangle = 1,\ k = 0, ..., N_N \tag{6.3.5}$$

Il est important de remarquer que les contraintes de tangence avec les bords mobiles de la surface nécessitent une valeur initiale de σ_1, σ_2 et n_k. La première étape consiste donc à obtenir des surfaces sans tenir compte de ces contraintes (6.3.4) et (6.3.5) puis à utiliser les surfaces obtenues pour initialiser le système complet. Ensuite, de façon itérative nous calculons pour chaque k la valeur moyenne des normales n_{k,σ_1} et n_{k,σ_2} aux frontières entre deux surfaces reconstruites σ_1 et σ_2. Cette moyenne est la valeur cible à atteindre pour l'itération en cours. Cet algorithme itératif est décrit dans la section 6.3.2.

Définition 6.3.6. ***Lissage***
Un terme de régularisation est introduit pour améliorer la qualité des surfaces. Ce terme est similaire à celui introduit pour les courbes dans la section 3.2.5, et correspond à :

$$E_{lissage} = E_{lissage,1} + E_{lissage,2} \tag{6.3.6}$$

avec

$$E_{lissage,1} = \iint (\|\sigma_u(u,v)\|^2 + \|\sigma_v(u,v)\|^2)\, \mathrm{d}u\, \mathrm{d}v$$
$$E_{lissage,2} = \iint (\|\sigma_{uu}(u,v)\|^2 + 2\|\sigma_{uv}(u,v)\|^2 + \|\sigma_{vv}(u,v)\|^2)\, \mathrm{d}u\, \mathrm{d}v$$

6.3.2 Résolution numérique

Soit $\mathbf{x}$ le vecteur des inconnues du système, c'est-à-dire les points de contrôle c_{ij} de la surface et les normales n_k aux frontières entre plusieurs surfaces reconstruites.
Considérons E_{total} comme la somme des contraintes décrites ci-dessus :

$$\begin{aligned} E_{total}(\mathbf{x}) &= \sum_{i=0}^{N} \varphi_i(\mathbf{x})^2 \\ &= E_{fitting} + E_{fixT} + E_{mobileT1} + E_{mobileT2} + E_{mobile\ normals} + E_{lissage} \end{aligned} \tag{6.3.7}$$

Alors pour reconstruire une surface qui satisfasse les contraintes voulues, nous cherchons la valeur $\mathbf{x} = \mathbf{x}^*$ tel que $\varphi_i(\mathbf{x}^*) = 0$, $i = 0, ..., N$. Le vecteur $\mathbf{x}^*$ solution contient les coordonnées des points de contrôle de la surface.

On utilise une méthode de Newton pour déterminer $\mathbf{x}^*$. Soit $\mathbf{x} = \mathbf{x}_0$ la valeur initiale des inconnues. Nous calculons de manière itérative les valeurs $\mathbf{x} = \mathbf{x}_n$ jusqu'à ce que $\varphi_i(\mathbf{x}_n)$ soit plus petit qu'un seuil fixé.

A l'itération n, $\varphi_i(\mathbf{x}_{n+1})$ peut être approché par :

$$\varphi_i(\mathbf{x}_{n+1}) \approx \varphi_i(\mathbf{x}_n) + \nabla\varphi_i(\mathbf{x}_n)^T(\mathbf{x}_{n+1} - \mathbf{x}_n)\ i = 1, ..., N \tag{6.3.8}$$

Rappelons la définition de l'équation (6.3.1),

$$\begin{aligned} &\varphi_i(\mathbf{x}_n) = \frac{1}{2}\mathbf{x}_n^T H_i \mathbf{x}_n + b_i^T \mathbf{x}_n + c_i\ i = 1, ..., N, \\ &\text{donc } \nabla\varphi_i \text{ est donné par } \nabla\varphi_i(\mathbf{x}_n) = H_i\mathbf{x}_n + b_i^T \end{aligned}$$

Comme nous cherchons $\varphi_i = 0$, $i = 1, ..., N$, à partir de l'équation (6.3.8) nous obtenons :

$$\nabla\varphi_i^T \mathbf{x}_{n+1} = \nabla\varphi_i(\mathbf{x}_n)^T \mathbf{x}_n - \varphi_i(\mathbf{x}_n) \tag{6.3.9}$$

qui consiste à résoudre un système linéaire $A_n\mathbf{x}_{n+1} = r_n^T$ pour déterminer $\mathbf{x}_{n+1}$ avec :

$$A_n = \begin{pmatrix} \nabla\varphi_1(\mathbf{x}_n)^T \\ \vdots \\ \nabla\varphi_N(\mathbf{x}_n)^T \end{pmatrix} \text{ et } r_n = \begin{pmatrix} \nabla\varphi_1(\mathbf{x}_n)^T \cdot \mathbf{x}_n - \varphi_1(\mathbf{x}) \\ \vdots \\ \nabla\varphi_N(\mathbf{x}_n)^T \cdot \mathbf{x}_n - \varphi_N(\mathbf{x}) \end{pmatrix} \tag{6.3.10}$$

6.3.3 Exemples

Exemple 1 - Coque de voiler

Un exemple de reconstruction de surface est illustré pour une coque de voilier dans les figures suivantes. Dans cet exemple, nous avons choisi de découper la coque en 4 patchs, deux fixes aux extrémités et deux mobiles au centre. Les deux patchs centraux vont être reconstruits avec la méthode de *form finding*. Ils seront raccordés aux surfaces fixes avec une continuité G^1. Dans ce cas, l'algorithme converge en 3 itérations et les surfaces obtenues sont satisfaisantes.

La figure 6.7 montre les deux surfaces centrales obtenues. Les points en rouge et magenta représentent le nuage de points P_l pour chacune des surfaces. Les deux surfaces ont les caractéristiques suivantes :

- Le nuage de points P_l est composé de 10×11 points

- Le nombre de points de contrôle utilisés pour les surfaces est fixé à 8×5
- Le degré des surfaces est fixé à 2, avec des vecteurs de nœuds uniformes dans les deux directions

Le nombre d'inconnues à résoudre dans le système linéaire (6.3.10) est 135 ($5 * 8 * 3$ (points de contrôles 3D) et $5 * 3$ (normales sur le bord de jonction en 3D)).

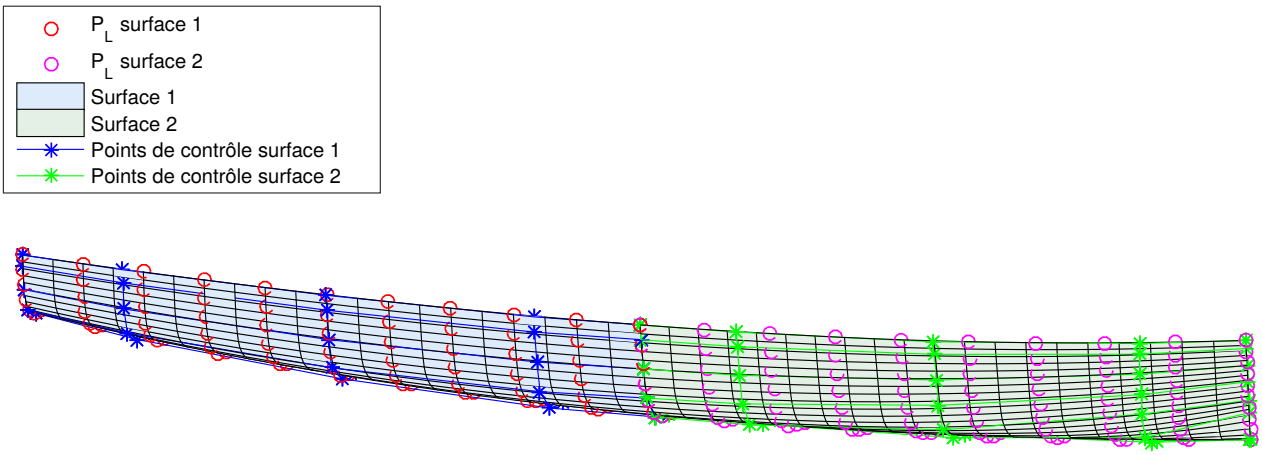


FIGURE 6.7 – Reconstruction de deux surfaces centrales d'une coque de voilier avec la technique *form finding*

La figure 6.8 montre la convergence de l'algorithme itératif.

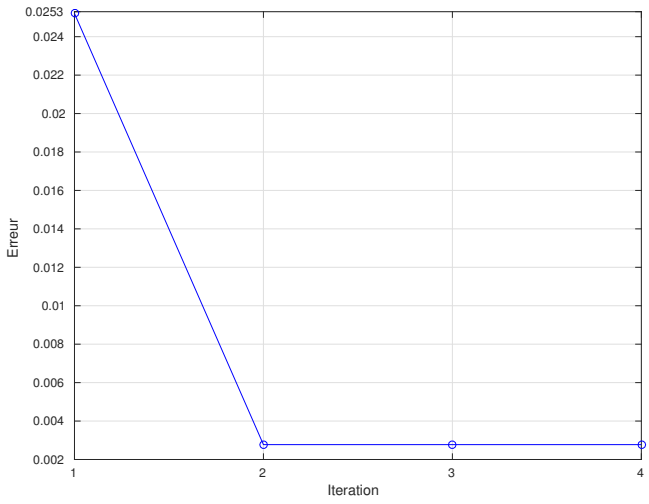


FIGURE 6.8 – Convergence de l'algorithme de reconstruction de surface pour la coque de voilier

Finalement, la figure 6.9 illustre la coque de voilier entière, avec les deux surfaces centrales reconstruites et les deux surfaces fixes aux extrémités.

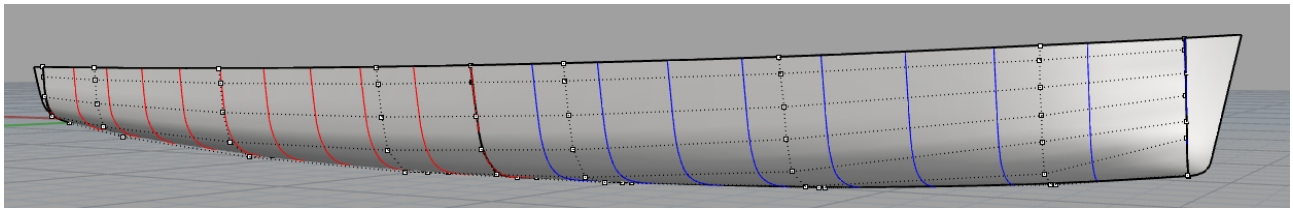


FIGURE 6.9 – Coque de voilier complète, avec les surfaces reconstruites et les surfaces fixes

Exemple 2 - Bulbe de chalutier

L'exemple de reconstruction du bulbe avec la technique *form finding* est illustré pour une coque de voilier dans les figures suivantes. Le bulbe est constitué de deux surfaces. L'une doit se raccorder sur 3 bords avec le reste de la coque restée fixe, et l'autre doit se raccorder à la première surface le long d'un bord. Dans ce cas, l'algorithme converge en 4 itérations et les surfaces obtenues sont satisfaisantes, meilleures et moins complexes que celles obtenues avec la méthode *Surface Network*.

La figure 6.12 montre les surfaces du bulbe obtenues. La figure 6.10 montre le nuage de points P_l pour chacune des surfaces. Les deux surfaces ont les caractéristiques suivantes :

- Le nuage de points P_l est composé de 50×16 points
- Le nombre de points de contrôle utilisés pour les surfaces est fixé à 10×8
- Le degré des surfaces est fixé à 2, avec des vecteurs de nœuds uniformes dans les deux directions

Le nombre d'inconnues à résoudre dans le système linéaire (6.3.10) est 270 ($10 * 8 * 3$ (points de contrôles 3D) et $10 * 3$ (normales sur le bord de jonction en 3D)).

La figure 6.11 montre la convergence de l'algorithme itératif.

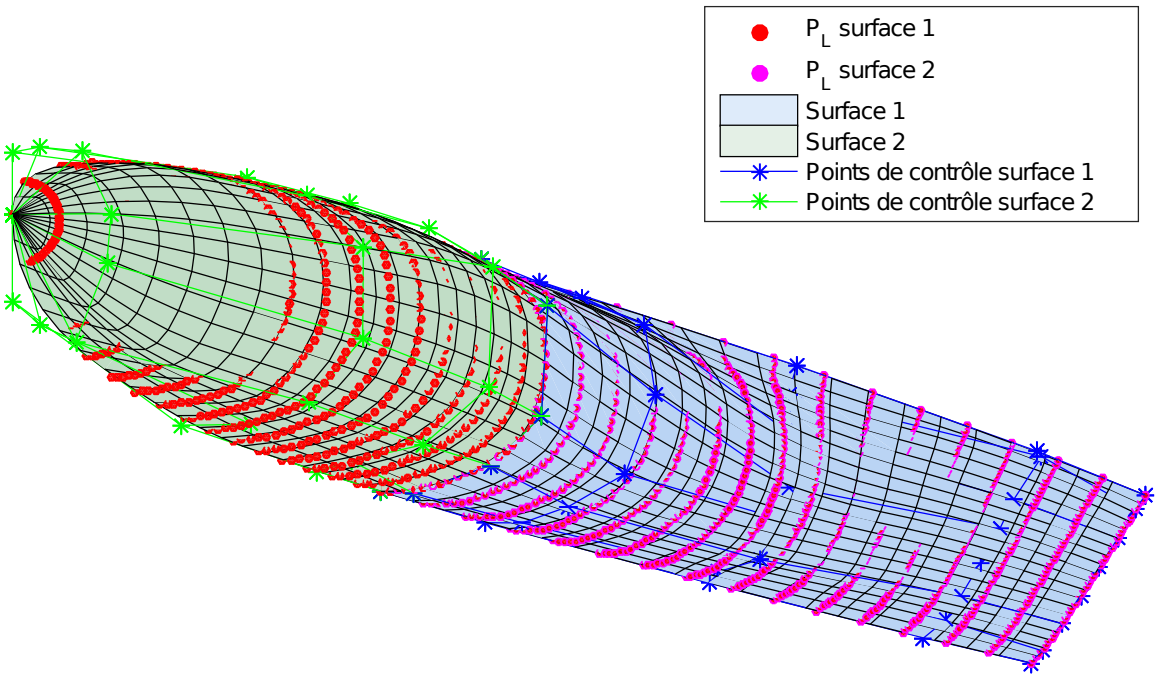


FIGURE 6.10 – Nuage de point utilisé pour reconstruire les surfaces du bulbe de chalutier

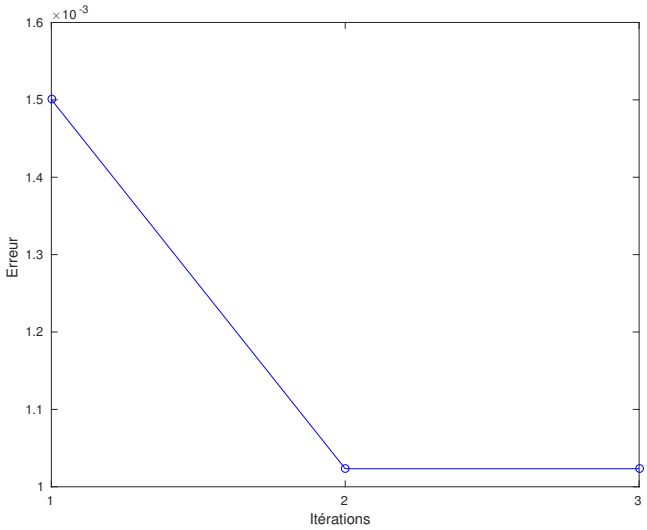


FIGURE 6.11 – Convergence de l'algorithme de reconstruction de surface pour le bulbe de chalutier

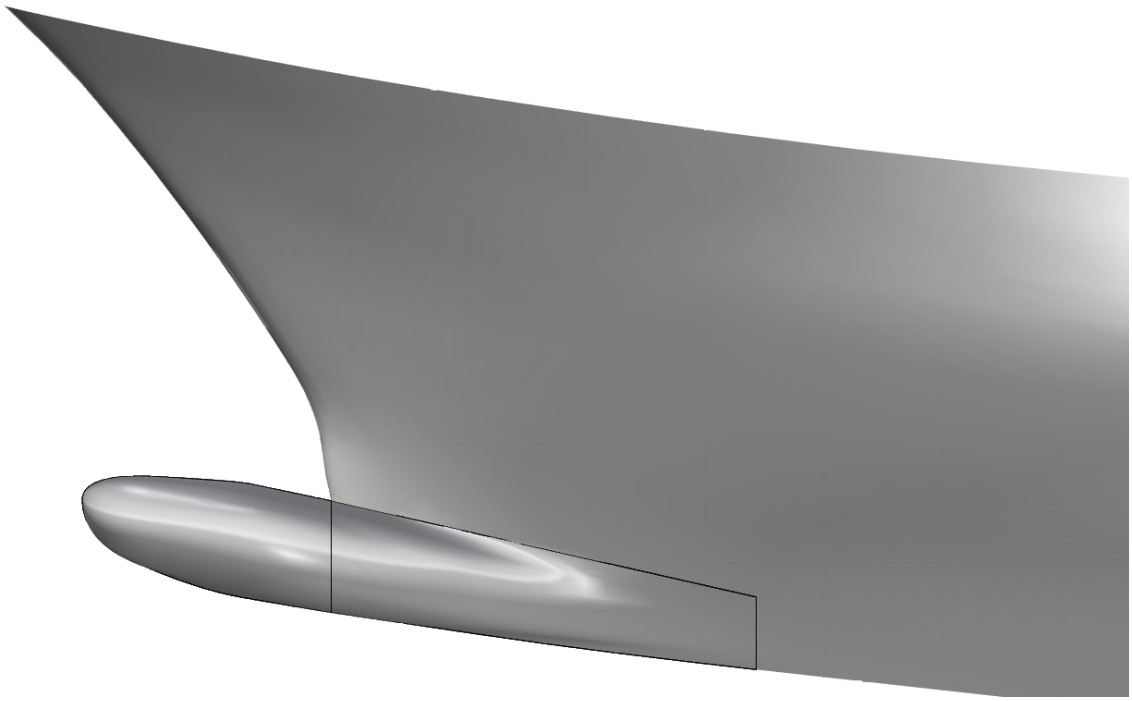


FIGURE 6.12 – Les deux surfaces du bulbe reconstruites avec la technique *form finding*

6.3.4 Résumé de l'algorithme

L'algorithme 4 expose le fonctionnement de l'algorithme de reconstruction de surface par *form Finding* de façon générale. Cette partie du modeleur a été implémentée avec MATLAB, mais pourra être développée en C++ pour améliorer la vitesse d'exécution (résolution de systèmes linéaires de grande taille).

Algorithme 4 : Algorithme général de la reconstruction de surface par *form Finding*

Data : Nuage de points distribué sur le squelette $P : \{P_l,\ l = 0, ..., N_P\}$; Normales des surfaces fixes aux points sur la frontière avec une surface reconstruite ; D_u le nombre de points de contrôle en u souhaité ; D_v le nombre de points de contrôle en v souhaité ; Degré des surfaces ; Vecteurs de nœuds des surfaces

Result : Surfaces B-Splines enveloppant le squelette

`/* Définition des tailles des matrices` H `et` r `*/`

$H = zeros(3 * D_u * D_v,\ 3 * D_u * D_v)$

$r = zeros(3 * D_u * D_v,\ 1)$

Identification des points P_l partagés par deux surfaces reconstruites adjacentes

`/* Obtention des surfaces initiales */`

Calcul des $H_{0,i}$ et $r_{0,i}$ en tenant compte uniquement des contraintes données par les équations (6.3.2) (approximation de surface), (6.3.3) (tangentes avec les surfaces fixes en u et v) et (6.3.6) (lissage)

$H_0 = \sum_{i=0}^{3}(H_{0,i}^T * H_{0,i})$

$r_0 = \sum_{i=0}^{3}(H_{0,i}^T * r_{0,i})$

Résolution du système $H_0\mathbf{x_0} = r_0$

Soit n=0

while $E_{total} > seuil\ d'erreur$ **do**

  Calcul des normales aux N_N points P_l situés sur la frontière entre deux surfaces reconstruites

  for *Pour toutes les surfaces à reconstruire* **do**

    Calcul des $H_{n,i}$ et $r_{n,i}$ en fonction des contraintes données par les équations (6.3.2) (approximation de surface), (6.3.3) (tangentes avec les surfaces fixes en u et v), (6.3.4) et (6.3.5) (tangentes avec les surfaces mobiles en u et v), et (6.3.6) (lissage)

    $H_n = \sum_{i=0}^{6}(H_{n,i}^T * H_{n,i})$

    $r_n = \sum_{i=0}^{6}(H_{n,i}^T * r_{n,i})$

    Résolution du système $H_n\mathbf{x_n} = r_n$

  end

  Calcul de l'erreur sur les normales aux frontières entre les surfaces reconstruites

  n=n+1

end

Chapitre 7

Optimisation automatique de forme

Dans ce chapitre, nous détaillons la mise en place d'une boucle d'optimisation automatique de forme. Nous avons implémenté une technique qui permet d'obtenir de façon automatique la forme optimale d'un objet, à partir d'une forme mère et d'un ou plusieurs critères de performance. Le chapitre 8 montrera des applications de cette technique.

Pour réaliser une boucle automatique d'optimisation de forme, il faut lier un modeleur paramétrique de forme, un mailleur, un solveur numérique capable d'évaluer les critères de performance choisis et un algorithme d'optimisation adéquat. Cet enchaînement est détaillé dans la section 7.1.
Ensuite, nous présentons les solveurs numériques et les algorithmes d'optimisation que nous avons utilisés et liés dans une boucle d'optimisation dans les sections 7.2 et 7.3.

Le but de ce chapitre est de donner des indications pratiques sur l'utilisation de solveurs et algorithmes d'optimisation dans le cadre du développement d'une boucle d'optimisation automatique de forme. Nous expliquerons uniquement les méthodes numériques adaptées à la réalisation de cette tâche.

7.1 Boucle automatique d'optimisation de forme

Une boucle automatique d'optimisation de forme [Brizzolara et al., 2015, Peri et al., 2001, Blanchard et al., 2013] est usuellement composée de :

1. un modeleur paramétrique, permettant de déformer l'objet en fonction d'un ensemble de paramètres de forme,
2. un mailleur surfacique ou volumique,
3. un solveur numérique, permettant d'évaluer les performances de l'objet déformé,
4. un algorithme d'optimisation qui détermine l'ensemble de paramètres de forme qui minimise ou maximise les critères de performance.

La figure 7.1 montre l'enchaînement des logiciels qui forment la boucle automatique d'optimisation de forme. La figure 7.2 détaille les étapes du modeleur.

La boucle commence avec la paramétrisation de l'objet par le modeleur, décrite dans le chapitre 4. Pour se lier avec le modeleur paramétrique développé dans cette thèse, la géométrie

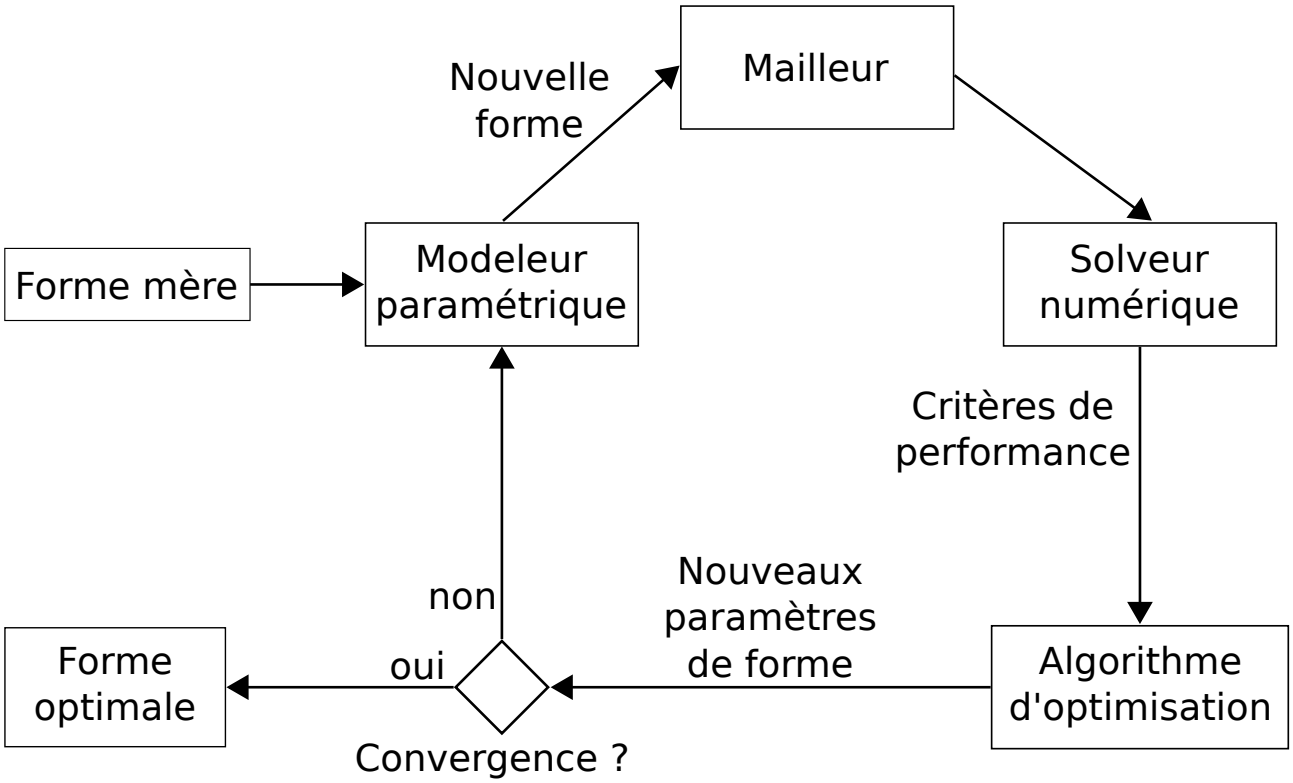


FIGURE 7.1 – Boucle d'optimisation automatique de forme

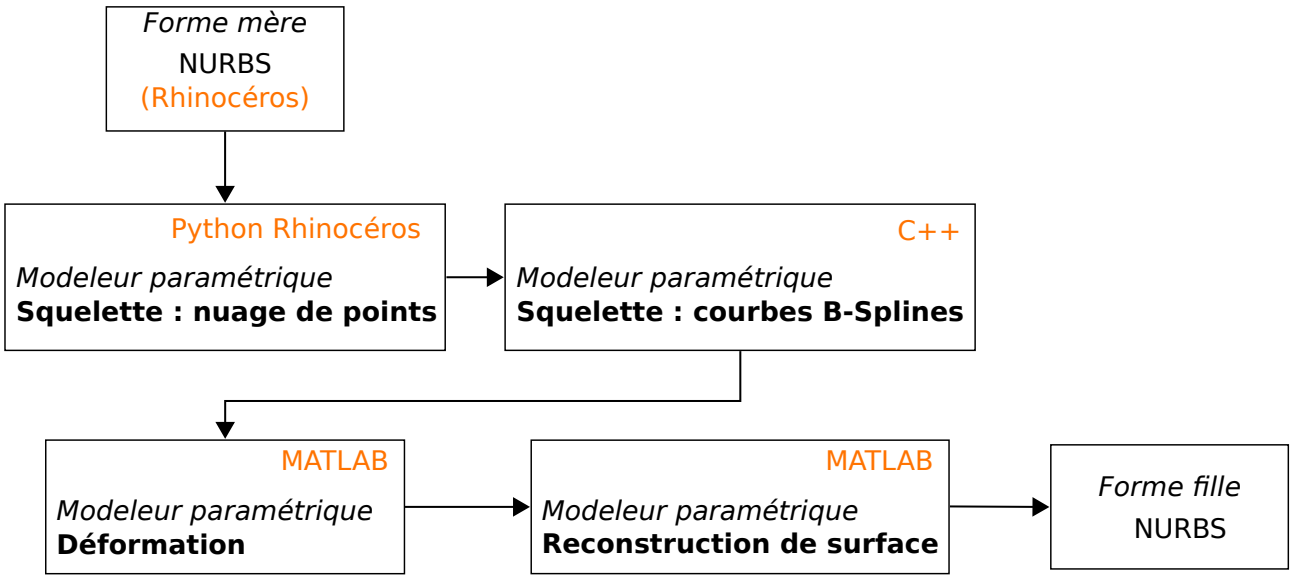


FIGURE 7.2 – Étapes du modeleur paramétrique et type de code

de la mère de l'objet doit être décrite avec des NURBS, comme expliqué dans la section 4.1.3. Notons que seules les étapes de déformation et de reconstruction de surface sont répétées à l'intérieur de la boucle d'optimisation automatique.

Ensuite, les trois étapes suivantes s'enchaînent jusqu'à la convergence de l'algorithme d'optimisation :

1. La forme est analysée avec le solveur numérique et ses performances sont transmises à l'algorithme d'optimisation.
2. En fonction des résultats, l'algorithme d'optimisation génère un nouvel ensemble de paramètres de forme.
3. Le modeleur déforme l'objet en fonction de ces paramètres et la forme est à nouveau analysée avec le solveur numérique. La méthodologie de déformation est expliquée dans le chapitre 5. Si le solveur nécessite une description de l'objet par des surfaces continues, alors nous utilisons une des méthodes de reconstruction de surfaces expliquée dans le chapitre 6. Lorsque c'est possible, le modeleur géométrique génère automatiquement un maillage adapté au solveur numérique.

Après convergence, nous obtenons une géométrie optimale de l'objet correspondant aux critères de performances choisis.

Ce type de méthode permet de tester des centaines de designs automatiquement. Un architecte naval réalisant cette recherche de forme optimale manuellement ne peut tester que quelques designs. Le champs d'exploration de formes possible est nettement élargi par une boucle d'optimisation automatique de forme.

Nous décrivons ci-dessous les solveur numériques et les algorithmes d'optimisation que nous avons utilisé et liés dans une boucle d'optimisation automatique de forme avec le modeleur paramétrique développé durant cette thèse.

7.2 Simulation numérique, introduction aux modèles de fluide

Les applications de cette thèse sont liées à des objets soumis à des écoulements fluides. Nous utilisons trois types de solveurs : fluide parfait avec VLM, fluide parfait avec couche limite et RANSE (Reynolds Averaged Navier-Stokes Equations), que nous décrivons brièvement ci-après.

Modèle fluide parfait

Les solveurs fluide parfait se basent sur l'hypothèse que le fluide est incompressible, non visqueux et irrotationnel.

La méthode *Vortex lattice method (VLM)* est un modèle fluide parfait très répandu [Katz and Plotkin, 2001].

Ce modèle est basé sur la méthode particulaire de Rehbach [Rehbach, 1977, Charvet, 1991, Charvet et al., 1996] et Huberson [Huberson, 1986] qui permet de modéliser des écoulements tridimensionnels incompressibles en régime instationnaire autour d'un obstacle de faible épaisseur.

Le sillage est décrit pas des particules tourbillonnaires, dont la cinématique est décrite à partir d'ne méthode lagrangienne. L'évolution de l'intensité des tourbillons est calculée à partir des équations d'Helmholtz.

La méthode VLM permet notamment de décrire les écoulements à faible angle d'incidence sur des profils portants minces. L'écoulement est attaché sur la totalité de la surface du profil, et le sillage se forme au bord de fuite par des lâchers tourbillonnaires en respectant la condition de Kutta-Joukowski. Cette méthode permet d'obtenir la portance du profil, et sa traînée induite.

La surface du profil est modélisée par des panneaux rectangulaires, et donc seul un maillage de la surface de l'objet est suffisant pour réaliser la simulation.

La méthode fluide parfait néglige les effets de viscosité du fluide et impose le point de décollement au bord de fuite du corps. Ce sont des simplifications conséquentes du modèle. Les résultats des simulations peuvent être très éloignés de la réalité si les conditions d'utilisation du modèle ne sont pas satisfaites.

Lorsque les conditions d'utilisation sont en accord avec le modèle, les résultats obtenus sont proches de ceux que l'on pourrait obtenir avec des simulations de type RANSE, mais pour des temps de calculs beaucoup plus courts.

Les simulations de profils portant 2D et les foils (profils en 3D) dans des écoulements stationnaires à faible incidence correspondent au domaine d'application où le modèle fluide parfait est cohérent.

Pour la simulation des profils en 2D, nous utilisons XFOIL [Drela, 1989], décrit dans la section 8.1.1. XFOIL est un code fluide parfait avec couche limite, c'est à dire que les effets visqueux et le décollement sont pris en compte. Pour la simulation des profils en 3D, par exemple pour des foils, nous utilisons le logiciel AVANTI [Roux, 1999, Durand et al., 2014, Lothodé et al., 2013], décrit dans la section 8.2.1.

Modèle Reynolds Averaged Navier-Stokes Simulation (RANSE)

Contrairement au modèle fluide parfait, les méthodes *Reynolds Averaged Navier-Stokes Equations* (RANSE) sont plus précises et permettent de simuler les effets visqueux du fluide. La turbulence peut être modélisée et prise en compte dans la simulation. En revanche la puissance et les temps de calcul nécessaires à réaliser de telles simulations sont beaucoup plus importants. En général ce type de calcul demande d'utiliser plusieurs dizaines de processeurs en parallèle sur plusieurs heures.

Les solveurs RANSE résolvent les équations de Navier Stokes en moyenne de Reynolds pour des écoulements incompressibles, turbulents et instationnaires [Duvigneau et al., 2003, Leroyer, 2004, De Nayer, 2008, Durand, 2012, Roux, 1999].

Ils sont généralement basés sur une méthode de volumes finis : le domaine de calcul est discrétisé en volumes élémentaires, ou mailles, dans chacun desquels un bilan local des flux est calculé.

Pour les calculs avec surface libre, une méthode de capture d'interface peut être utilisée [Queutey and Visonneau, 2007].

La plupart des solveurs peuvent traiter des maillages non-structurés composés de volumes de formes arbitraires, permettant de capturer des géométries très complexes [Wackers et al., 2010, Wackers et al., 2012]. Les mailleurs volumiques sont très sensibles à la qualité du modèle géométrique qui décrit l'objet. En général, ils permettent d'importer des CAO décrite avec un standard NURBS comme l'IGES ou le STEP. Si la géométrie importée n'est pas propre et entièrement étanche, des corrections manuelles doivent être faites, empêchant la réalisation d'un processus entièrement automatique.

Nous avons utilisé la suite logiciel FINE^{TM}/Marine pour réaliser ce type de simulations, avec le mailleur volumique HEXPRESS^{TM} et le solveur fluide ISIS-CFD.

7.3 Introduction au principaux algorithmes d'optimisation

Le choix de l'algorithme d'optimisation est un pilier central de la boucle d'optimisation automatique de forme. Ce choix dépend du type de problème abordé (mono/multi-objectifs), du solveur choisi (de sa rapidité, de sa précision) et de la variation dans l'espace des paramètres du(des) critère(s) de performance choisi(s).

Nous décrivons ici brièvement les principes généraux des algorithmes d'optimisation numérique mono-objectif, section 7.3.1, et multi-objectifs, section 7.3.2. Nous détaillons les algorithmes basés sur des méta-modèles, qui sont particulièrement bien adaptés pour les applications liées aux solveurs numériques.

7.3.1 Problèmes mono-objectif

Considérons le problème suivant :

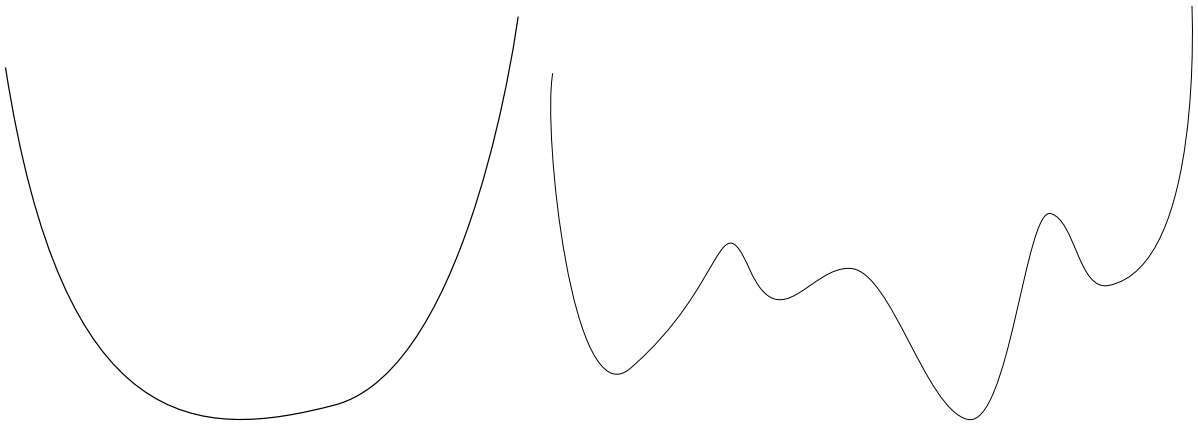
$$\begin{array}{ll} \min_{x\in\mathbb{R}^n} & f(x) \\ & g_i(x) = 0, \quad i = 1, ..., l \\ & h_j(x) \geq 0, \quad j = 1, ..., m \\ & x \in \mathcal{S} \subset \mathbb{R}^n \end{array} \tag{7.3.1}$$

où $f(x)$ est la fonction objectif, ou fonction coût du problème, $g_i(x)$ sont les contraintes d'égalité, $h_j(x)$ les contraintes d'inégalité du problème, et $\mathcal{S}$ est l'espace des variables ou espace de recherche.

Alors x_A est dit point admissible de (7.3.1) si $x_A \in \mathcal{S}$ et si les contraintes du problème sont respectées : $g_i(x_A) = 0,\ i = 1, ..., l$ et $h_j(x_A) \geq 0,\ j = 1, ..., m$.
$x^* \in \mathcal{S}$ est un minimum global de $f \iff \forall x \in \mathcal{S},\ f(x^*) \leq f(x)$.
$x^* \in \mathcal{S}$ est un minimum local de $f \iff$ il existe $\epsilon > 0$ tel que $\forall x \in \mathcal{S}$ vérifiant $\|x - x^*\| \leq \epsilon$, $f(x^*) \leq f(x)$.

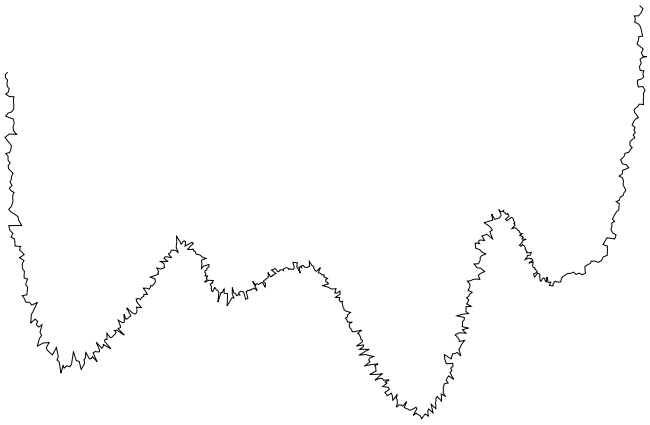
La figure 7.3(a) montre une une fonction convexe qui présente par définition un minimum unique global. La figure 7.3(b) montre une fonction présentant deux minima locaux et un minimum global. Ce type de fonction est appelé multi-modale. La figure 7.3(c) montre un graphe de données bruitées. Ce graphe présente de nombreux minima locaux.

[Nocedal and Wright, 2006] donne en détails les théorèmes d'existence et d'unicité du minimum d'une fonction.



(a) Fonction convexe

(b) Fonction multi-modale



(c) Fonction bruitée

FIGURE 7.3 – Les différents types de fonctions objectif

En pratique, des algorithmes numériques sont utilisés pour résoudre un problème d'optimisation et déterminer x^*. Cet algorithmes sont classés en deux catégories principales décrites ci-après : les méthodes de recherche locales et les méthodes de recherche globales.

Nous détaillons également les méthodes d'optimisation basées sur un méta-modèle. Ces méthodes sont particulièrement utiles pour déterminer le minimum de fonctions objectifs coûteuses à évaluer, comme peuvent l'être les critères de performance calculés avec un solveur numérique.

Méthodes locales

Les méthodes locales sont efficaces sur des fonctions convexes. Elles convergent uniquement vers des minima locaux [Vanderplaats, 1987, Fletcher, 1987, Gill et al., 1981].

Les algorithmes les plus utilisés pour ce type de méthodes sont les algorithmes de descente, basés sur le modèle suivant :

On choisi un point de départ x_0, puis à l'itération k on effectue les étapes suivantes :

- Évaluation du gradient $\nabla f(x_k)$
- Choix d'une direction de recherche $d_k(\nabla f(x_k))$
- Choix d'un pas de recherche ρ_k
- Mise à jour $x_{k+1} = x_k + \rho_k d_k$

L'algorithme s'arrête lorsque $\|x_{k+1} - x_k\| \leq \epsilon$, ou que le pas ρ_k devient trop petit. ϵ est le seuil de tolérance de l'algorithme.

Les algorithmes de *plus grande descente*, de *gradient conjugué*, de *Newton* ou *quasi-Newton* sont les plus connus parmi les algorithmes de descente.

Chaque algorithme se caractérise par le choix de la direction et du pas de recherche, et en conséquence par la rapidité de convergence.

Méthodes globales

Les méthodes globales permettent d'explorer plus largement la fonction coût et donc de détecter plusieurs minima. Ces méthodes peuvent trouver le meilleur minimum local entre plusieurs minima locaux de fonctions multi-modales [Michalewicz, 1996, Goldberg, 1989].

Il existe plusieurs catégories d'algorithme d'optimisation globale. Nous allons détailler succinctement les deux catégories suivantes :

- Algorithme évolutionnaires
- *Pattern search* (algorithme de Nelder-Mead simplex ou de Torczon)

D'autres algorithmes que nous ne décrivons pas ici sont très utilisés comme la recherche aléatoire, l'essaim de particules, le recuis simulé ou la méthode de monte-carlo.

Les algorithmes évolutionnaires sont basés sur une recherche aléatoire, tout comme les algorithmes de recherche aléatoire, d'essaim de particules, de recuis simulé, alors que les méthodes *Pattern search* ne le sont pas.

Il est important de noter qu'aucun de ces algorithmes ne nécessite de calculer le gradient de la fonction objectif.

La famille des *algorithmes évolutionnaires* s'inspirent de la théorie de l'évolution et de la sélection naturelle de Darwin. Le plus connu est l'algorithme génétique [Holland, 1992], que nous décrivons ici. Ce type d'algorithmes ont un vocabulaire particulier : un point de l'espace de recherche $\mathcal{S}$ est appelé *individu* et la *population* désigne un ensemble de point.

L'algorithme génétique se compose des étapes suivantes :

1. Une population initiale $P(t=0)$ est distribuée dans $\mathcal{S}$ aléatoirement.
2. Les individus de la population courante $P(t)$ sont évalués, c'est-à-dire que l'on calcule la valeur de la fonction objectif à tous les points de la population.
3. On sélectionne certains individus de $P(t)$ en fonction de la valeur de la fonction objectif, en général on choisi ceux qui la minimisent le mieux.
4. Les individus sélectionnés sont *croisés* et *mutés*, c'est à dire que de nouveaux points sont crées à partir des croisements et modifiés aléatoirement pour obtenir une nouvelle population $P(t+1)$.
5. Nouvelle itération : $t = t+1$, les étapes 2 à 5 sont répétées jusqu'à convergence.
6. La convergence est atteinte lorsque le niveau de performance souhaité est atteint, c'est-à-dire lorsque les individus ont identifié un minimum suffisamment petit.

Les algorithmes type *Pattern search* sont basés sur le concept de simplex, un polytope composé de $N+1$ sommet pour un espace de N dimensions. A chaque itération de la méthode le simplexe se déplace et se transforme (réduction ou augmentation de taille) jusqu'à converger vers le minimum de la fonction objectif [Nelder and Mead, 1965, Torczon, 1989].

Nous décrivons ici le modèle de ce type d'algorithme :

1. L'algorithme est initialisé avec $N+1$ points $\{x_1, ..., x_{N+1}\}$, qui forment le simplexe initial.
2. Chacun des points est évalué et correspond à une valeur de la fonction objectif . Les valeurs de la fonction objectif sont triées de la plus petite à la plus grande : $f(x_1) \leq ... \leq f(x_N) \leq f(x_{N+1})$.
3. On calcule le centre de gravité x_G des N meilleurs points $\{x_1, ..., x_N\}$, puis on calcule la réflexion du point le moins bon x_{N+1} par rapport à x_G : $x_r = x_G + \alpha\,(x_G - x_{N+1})$, où α est un coefficient de réflexion.
4. Ensuite si $f(x_r) \leq f(x_N)$, c'est-à-dire que la recherche va dans une direction intéressante, le simplexe est étiré selon un facteur donné β. Le nouveau point est calculé comme $x_e = x_G + \beta\,(x_G - x_{N+1})$. Si $f(x_e) \leq f(x_r)$, alors on remplace x_{N+1} par x_e, sinon on remplace x_{N+1} par x_r
5. En revanche si $f(x_r) > f(x_N)$, la recherche va dans une mauvaise direction, le simplexe est alors contracté selon un facteur donné γ. Le nouveau point est calculé comme $x_c = x_{N+1} + \gamma\,(x_G - x_{N+1})$. Si $f(x_c) \leq f(x_N)$, alors on remplace x_{N+1} par x_c et on retourne directement à l'étape 2, sinon on passe par l'étape 6.
6. Enfin, une homothétie de rapport δ et de centre le meilleur point x_1 est utilisée pour calculer les nouvelles coordonnées du simplexe : $x_i = x_1 + \delta(x_i - x_1)$, pour $i = 2, ..., N+1$. Puis retour à l'étape 2.
7. La convergence est atteinte lorsque le simplexe est suffisamment petit et que ses $N+1$ sommets convergent vers le même point.

Les coefficients α, β, γ et δ doivent satisfaire les contraintes suivantes :

$$\alpha > 0, \quad 0 < \beta < 1, \quad \gamma > 1, \quad 0 < \delta < 1$$

Pour l'algorithme de Nelder-Mead, en général on choisi $\alpha = 1$, $\beta = 1/2$, $\gamma = 2$, $\delta = 1/2$.

Pour améliorer la convergence vers un minimum global, l'algorithme est généralement réinitialisé autour de la dernière solution obtenue et ce jusqu'à ne plus améliorer la solution finale.

Méthodes avec méta-modèles

Pour détecter des minima globaux, les algorithmes évolutionnaires sont efficaces mais très coûteux en nombre d'évaluations de la fonction objectif. Les applications liées à l'utilisation d'un solveur numérique pour l'évaluation de la performance d'un objet ont une fonction objectif est très coûteuse à évaluer, en termes de temps et de ressources de calcul.

Les algorithmes basés sur des méta-modèles permettent de contourner cette difficulté en remplaçant la fonction objectif coûteuse par un modèle approché plus simple [Jones et al., 1998, Jones, 2001]. L'idée générale de cette famille d'algorithmes est la suivante :

1. Construction d'une base de données de valeurs de la fonction objectif $(x_i, f_i = f(x_i))_{i=1,\dots,N}$, appelée *Plan d'Expérience*.
2. Construction d'un méta-modèle $\hat{f}$ à partir de cette base de données (interpolation polynomiale, surface de réponse, fonctions de bases radiales, Krigeage, ect.).
3. Résolution du problème d'optimisation choisi sur la fonction $\hat{f}$ pour obtenir un nouveau point x^*, déterminé par exemple avec un algorithme génétique.
4. Évaluation de la fonction objectif en x^*, et enrichissement de la base de données de l'étape 1 avec le point $(x^*, f(x^*))$. Retour à l'étape 2 avec la nouvelle base de données.

Cette méthode converge vers le minium de f en utilisant principalement des évaluations de $\hat{f}$.

L'une des méthodes les plus connue pour construire le méta-modèle est le *Krigeage* (ou *Kriging* en anglais), appelé aussi *processus Gaussien* (*Gaussian process regression* en anglais). Cette méthode est basée sur une approche statistique et permet d'obtenir une estimation de la fonction f à interpoler et un intervalle de confiance entre deux point successifs du plan d'expérience. Elle est particulièrement bien adaptée pour les données bruitées, comme peuvent l'être les résultats de simulations numériques [Duvigneau and Chandrashekar, 2012].

Le Krigeage consiste à considérer les valeurs du plan d'expérience comme des résultats de la réalisation d'un processus Gaussien [Jones, 2001]. Une prédiction de la valeur moyenne du processus Gaussien complet et de sa variance peuvent être calculées, donnant les valeurs de la fonction d'interpolation et l'erreur d'incertitude associée à chaque point [Rasmussen and Williams, 2005].

La figure 7.4 illustre un méta-modèle de Krigeage sur une fonction objectif analytique connue tracée en noir : $f(x) = -0.4\,sin(8x) + x\,sin(6x)$, avec 6 points d'échantillonnages dans le plan d'expérience, tracé en rouge. La prédiction de la moyenne processus Gaussien, et donc la fonction $\hat{f}$ donnée par le méta-modèle du Krigeage, est tracée en blue. La variance $\hat{\sigma}(x)$ de ce processus Gaussien est prédite et nous traçons les valeurs de la moyenne $+/- \; 3\,\hat{\sigma}$, qui est la valeur standard donnée par le Krigeage pour atteindre un intervalle de confiance de 95%.

L'algorithme d'optimisation *Efficient Global Optimization (EGO)* [Jones et al., 1998] est un algorithme basé sur un méta-modèle. *EGO* sélectionne le point x^* ajouté à l'étape 3 comme le

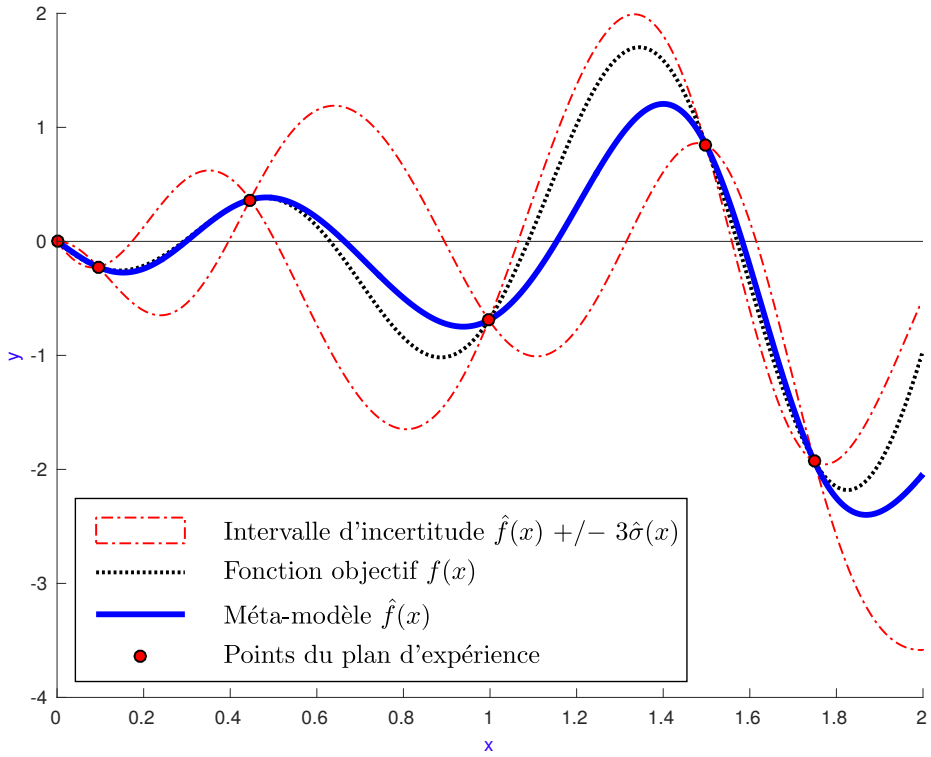


FIGURE 7.4 – Méta-modèle de Krigeage sur une fonction 1D

point qui minimise (ou maximise) une *fonction de mérite*. Utiliser ces fonctions plutôt que le minimum du méta-modèle $\hat{f}$ permet par exemple d'explorer les zones avec une grande incertitude et ainsi améliorer la qualité du méta-modèle. Cette technique évite de converger vers un minimum du méta-modèle qui ne serait pas un minimum du modèle original.

Parmi les fonction de mérite les plus connues, citons *l'amélioration espérée* (ou *Expected Improvement*, *EI*, en anglais) [Jones et al., 1998]. Notons x_{min} le meilleur point du plan d'expérience. Suivant la définition du méta-modèle de Krigeage, les valeurs possibles de la fonction objectif f au point x^* sont distribuées suivant une loi normale. Les valeurs de la fonction objectif au point x^* qui améliorent le modèle vérifient : $f(x^*) < f(x_{min})$. Les autres valeurs possibles du point x^* qui ne vérifie pas l'amélioration sont mises à zéro. L'amélioration en x^* est donc définie comme $max(f(x_{min}) - f(x^*), 0)$.

L'*EI* consiste donc à déterminer x^* qui maximise $f(x_{min}) - f(x^*)$.

Pour construire le plan d'expérience, il est possible d'échantillonner de façon régulière le domaine de variation de x. D'autres techniques ont été introduites par [McKay et al., 1979, Iman et al., 1981], notamment l'*échantillonnage par hypercube latin*, qui est une méthode statistique permettant de mieux répartir les échantillonnages dans l'espace des paramètres. L'échantillonnage par hypercube latin découpe chaque dimension de l'espace des paramètres en N intervalles, puis tire au hasard un point dans chacun de ces intervalles. Cette approche assure une bonne couverture du domaine de variation de x.

7.3.2 Problèmes multi-objectifs

Un problème multi-objectifs est formulé comme [Collette and Siarry, 2002] :

$$\begin{array}{lll} \min_{x\in\mathbb{R}^n} & (f_1(x), f_2(x), ..., f_k(x)), & k \geq 2 \\ & g_i(x) = 0, & i = 1, ..., l \\ & h_j(x) \geq 0, & j = 1, ..., m \\ & x \in \mathcal{S} \subset \mathbb{R}^n & \end{array} \tag{7.3.2}$$

Ce type de problèmes n'a pas solution unique mais une multitude de solutions situées sur un *front de Pareto,* qui correspond aux meilleurs compromis entre les k fonctions objectifs.

La notation *min* du problème (7.3.2) correspond à l'identification de points minimaux au sens de Pareto, C'est-à-dire en utilisant un critère d'ordre partiel au sens de la dominance de Pareto, définie dans la définition 7.3.1.

Le but des algorithmes d'optimisation multi-objectifs est de déterminer le front de Pareto du problème [Collette and Siarry, 2002].

Définition 7.3.1. *Un point $x \in \mathcal{S}$ solution de* (7.3.2) *domine au sens de Pareto une solution $y \in \mathcal{S}$ si et seulement si $\forall k, f_k(x) \leq f_k(y)$ et $\exists k$ tel que $f_k(x) < f_k(y)$.*
Le front de Pareto est l'ensemble des points non dominés de $\mathcal{S}$.

La figure 7.5 montre un exemple de front de Pareto. Les solutions possibles, en bleu, sont dominées par les points du front de Pareto, en rouge.

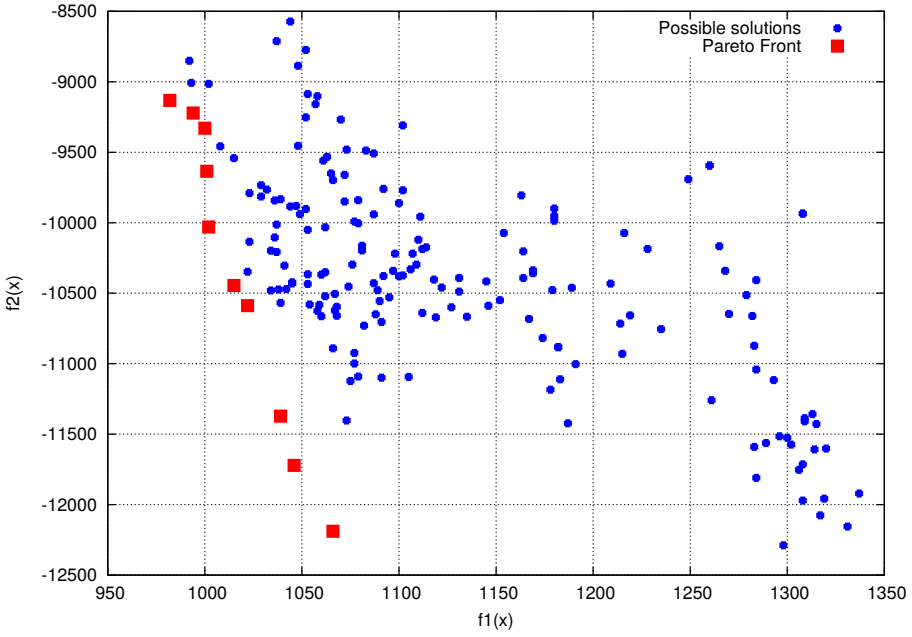


FIGURE 7.5 – Exemple de front de Pareto

Pour déterminer le front de Pareto, des méthodes basées sur les algorithmes génétiques sont souvent utilisées.

Les individus de la population échantillonnées sur les fonctions objectifs sont classés suivant le critère de dominance décrit dans la définition 7.3.1. Ce sont les points qui dominent le plus

d'autres points qui sont choisis comme les meilleurs individus. Ensuite, les étapes habituelles de l'algorithme génétique sont réalisées.

Un autre approche basée sur les algorithmes évolutionnaire est couramment utilisée pour déterminer le front de Pareto. Cette fois, l'algorithme multi-objectifs est basé sur un autre type d'algorithmes évolutionnaire : le (λ, μ)-ES.

La population initiale est modifiée par des perturbations aléatoires de ses individus, suivant une loi de probabilité normale.

L'algorithme *Pareto Archived Evolution Strategy (PAES)* est un des algorithmes basés sur (λ, μ)-ES les plus utilisés [Knowles and Corne, 1999, Abraham et al., 2005, Deb and Kalyanmoy, 2001].

PAES démarre d'une population initiale aléatoire, générée autour d'un point initial choisi par l'utilisateur. Les individus sont classés suivant le critère de dominance et sont sauvegardés. Ensuite, chaque individu i de la population génère un descendant m par perturbation aléatoire. Si m est dominé par i, m n'est pas conservé et i génère un nouveau descendant. Si m domine i, alors m est comparé aux autres points sauvegardés. Si m est dominé par un des points de la sauvegarde, m n'est pas conservé. Sinon, m est ajouté à la sauvegarde et tous les points dominés par m sont supprimés. Ces étapes se succèdent jusqu'à convergence vers le front de Pareto.

Conclusion

Nous utiliserons les algorithmes et méthodes numériques détaillées dans ce chapitre pour les applications pratiques proposées dans le chapitre suivant. Dans ces applications, nous utilisons en "boite noire" les outils de simulation, c'est pour cela qu'il est important de connaître le domaine de validité des méthodes. Nous utiliserons une toolbox d'optimisation, FAMOSA, développée à l'Inria par l'équipe de recherche *Opale*, qui dispose de la plupart des algorithmes décrits ici.

Chapitre 8

Applications

Ce chapitre est consacré aux applications pratiques du modeleur paramétrique. Nous présentons trois cas d'applications en utilisant différents solveurs numériques.

Le premier exemple porte sur l'optimisation de la forme d'un profil d'aile d'avion pour maximiser la finesse tout en imposant une valeur minimal pour la portance. Dans ce cas, le solveur fluide parfait avec couche limite *XFOIL* est utilisé.

Le deuxième exemple porte sur l'optimisation multi-objectifs de la forme d'un foil AC45. Nous cherchons à minimiser la traînée en maximisant la stabilité du foil. Le solveur fluide parfait *AVANTI*, couplé à *XFOIL* est utilisé.
Une boucle d'optimisation automatique est proposée pour chacun de ces deux exemples.

La dernière application porte sur l'optimisation de la forme d'un bulbe de chalutier de pêche, pour minimiser la traînée totale de la coque. Pour ces simulations, nous utilisons le solveur RANS *ISIS-CFD* de la suite FINE^{TM}/Marine.

Les temps d'exécution donnés ont été mesurés sur un PC portable quatre cœurs HP Probook-450 avec un Intel® Core^{TM} i7-4702MQ CPU 2.20GHZ, RAM 8.00 Go.
Les temps d'exécution de ISIS-CFD sont donnés pour 32CPU utilisés sur un cluster de calcul, doté au total de 144 processeurs Intel® E5-2670 - 2.60 GHz - 8 cœurs, 4608 Go de mémoire vive.

8.1 Optimisation de forme d'un profil

Nous proposons une application du modeleur paramétrique à l'optimisation automatique de forme d'un profil d'aile d'avion.

Beaucoup d'études ont été menées sur les performances des profils d'ailes, notamment par le *National Advisory Committee for Aeronautics* (NACA), et donc il existe de nombreux profils de références. Les profils sont des formes utilisées dans des domaine d'applications très variés : ailes d'avions (avions de lignes, militaires, planeurs), éoliennes, hélices, turbines, foils, tuyères, ailerons automobiles, ect.

En fonction du domaine d'application, la forme d'un profil varie énormément. Il est intéressant de pouvoir proposer des outils d'optimisation de forme afin d'ajuster au mieux les performances d'un profil à l'application visée.

Dans cet exemple, nous avons réalisé une boucle d'optimisation automatique de forme complète, telle que décrite dans le chapitre 7. Le modeleur paramétrique a été couplé au solveur fluide parfait XFOIL ; décrit ci-dessous, et à la toolbox d'optimisation FAMOSA, développée à l'Inria par l'équipe de recherche *Opale* [El Majd et al., 2008, Duvigneau and Chandrashekar, 2012]. Le maillage d'XFOIL est généré automatiquement par le logiciel. Comme donnée d'entrée, il faut fournir à XFOIL un nuage de point ordonné distribué sur la surface du profil. Ce type de format est généré automatiquement par le modeleur paramétrique.

Nous avons ainsi pu réaliser des centaines de calculs de façon automatique pour converger vers des formes de profil optimales.

8.1.1 Simulation avec XFOIL

XFOIL est un solveur très répandu pour l'analyse aérodynamique de profils, développé par[Drela, 1989]. Il permet de calculer la portance et la traînée de profils 2D.

XFOIL permet de simuler des écoulements fluides bi-dimensionnel et incompressibles sur des profils avec une *méthode à panneau* (*panel code method* en anglais). Les effets visqueux sont modélisés avec un modèle de couche limite permettant de déterminer la position de la transition laminaire - turbulent de l'écoulement par une méthode e^n [Van Ingen and Technische Hogeschool Delft, 1956, Smith et al., 1956].

XFOIL contient une base de donnée des profils NACA. Nous utilisons ici le *NACA0012*, illustré dans la figure 8.1, comme forme de départ.

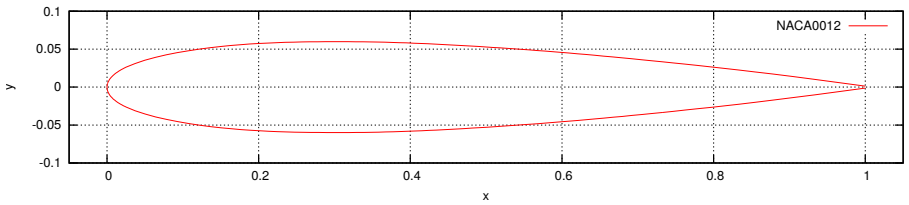


FIGURE 8.1 – Illustration du NACA0012

Nous imposons un nombre de Reynolds de 10^6 pour toutes les simulations, la longueur de référence considérée ici est la corde des profils qui est fixée à 1m.

8.1.2 Critères de performance

La finesse d'un profil est calculée comme le rapport entre le coefficient de portance C_l et le coefficient de traînée C_d :

$$finesse = \frac{C_l}{C_d}$$

L'amélioration de la finesse est un objectif important lors de la conception d'ailes d'avion. En effet, le designer cherche à obtenir un profil qui permet de créer une force de portance au moins opposée au poids de l'avion, tout en diminuant le plus possible la traînée, afin de réduire

la consommation de carburant.

Nous nous fixons comme objectif d'optimiser la finesse, tout en imposant que la portance soit supérieure à un seuil critique fixé s.

Nous cherchons donc à résoudre le problème d'optimisation suivant :

$$\max_{C_l, C_d} F_{obj} = \frac{C_l}{C_d} + \lambda \, 1_{](C_l - s < 0)[} \, (C_l - s) \tag{8.1.1}$$

$1_{](C_l - s < 0)[}$ est une fonction indicatrice permettant d'activer la contrainte C_l, qui doit être supérieur au seuil s. Ce terme est nul quand $C_l > s$. En revanche, si $C_l < s$, alors ce terme vaut 1 et une pénalisation de facteur $\lambda \, (C_l - s)$ est appliquée. Dans notre cas, nous prennons λ très grand, soit 10^{10}.

Lorsque $(C_l - s < 0)$ est nul, seul le terme de finesse est actif.

Notons que FAMOSA est paramétré pour minimiser les fonctions objectif. Nous résolvons donc le problème équivalant à (8.1.1) :

$$\min_{C_l, C_d} -F_{obj}$$

8.1.3 Déformations

Dans ce cas le squelette est simplifié en une seule section, et ne contient pas de génératrice. La courbe B-Spline décrivant le *NACA0012* a été obtenue en utilisant l'algorithme d'approximation de courbe, présenté dans la section 3.2. 20 points de contrôles ont été utilisés, 10 pour l'extrados et 10 pour l'intrados.

Les paramètres de formes utilisés sont illustrés dans la figure 8.2. La corde est fixée à 1.

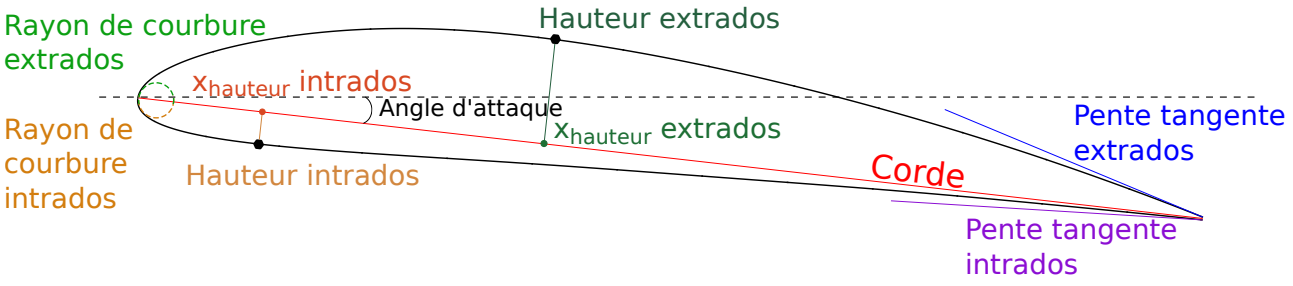


FIGURE 8.2 – Paramètres du profil

Comme indiqué dans la section 5.3, des contraintes métier sont imposées au profil. Les coordonnées du point de contrôle au bord d'attaque pour l'extrados et l'intrados doivent être identiques, et les tangentes à ces deux points doivent avoir une direction opposée, perpendiculaire à la corde.

Les limites de variations des neuf paramètres sont définies dans le tableau 8.1.

	Valeur initiale	**Variation min**	**Variation max**
Angle d'attaque	$0°$	$-5°$	$+20°$
Rayon de courbure extrados et intrados	$0,0164$	-100% (=0)	$+400\%$ (=$0,0821$)
Hauteur extrados	$0,06$m	-90% (=$0,006$m)	$+200\%$ (=$0,18$m)
Hauteur intrados	$-0,06$m	-350% (=$0,15$m)	$+200\%$ (=$-0,18$m)
$X_{Hauteur}$ extrados et intrados	$0,3$m	-40% (=$0,18$m)	$+50\%$ (=$0,45$m)
Pente tangente extrados	$7,892°$	-90% (=$0,789°$)	$+200\%$ (=$23.676°$)
Pente tangente intrados	$-7,892°$	-350% (=$19,731°$)	$+200\%$ (=$-23,676°$)

TABLE 8.1 – Bornes de l'espace des paramètres de formes du profil

Pour générer une nouvelle géométrie depuis la CAO originale, le modeleur paramétrique prend en moyenne 50 secondes pour la déformation du profil. Une méthode pas à pas (voir section 5.2.2) avec 5 itérations est utilisée.

En moyenne, une simulation XFOIL prend 2.1 secondes pour calculer les coefficients de portance et de traînée, y compris le post-traitement des résultats.

8.1.4 Résultats

Le but de cette étude est de déterminer les paramètres de formes du profil minimisant la fonction objectif (8.1.1). Le profil initial est le *NACA0012*. Le seuil fixé pour C_l est de $0,3$.

Pour déterminer le minimum de la fonction objectif nous avons utilisé un algorithme de type *Pattern search*, détaillé dans la section 7.3.

Les bornes de variations de l'espace des paramètres décrites dans le tableau 8.1 ont toutes été adimentionnées sur $[0,1]$. Le critère de converge à été fixé à une variation des paramètres inférieure à 10^{-3}. L'algorithme a été lancé une première fois, et a pris 31 itérations pour atteindre ce critère. Puis, l'algorithme a été lancé une deuxième fois à partir de point déterminé comme étant l'optimum. Cette fois, 20 itérations ont été nécessaires pour atteindre le critère de convergence.

Le tableau 8.2 présente les résultats du profil initial et du meilleur profil obtenu.

	$\mathbf{C_l}$	$\mathbf{C_d}$	**Finesse** (=$\mathbf{F_{obj}}$)
NACA0012	$0,0$	$0,00531$	$0,0$
Profil optimisé	$1,2838$	$0,00858$	$149,627$

TABLE 8.2 – Comparaison du profil initial et du profil optimisé

Le profil optimisé est illustré dans la figure 8.3. Les valeurs des paramètres correspondants sont détaillés dans le tableau 8.3.

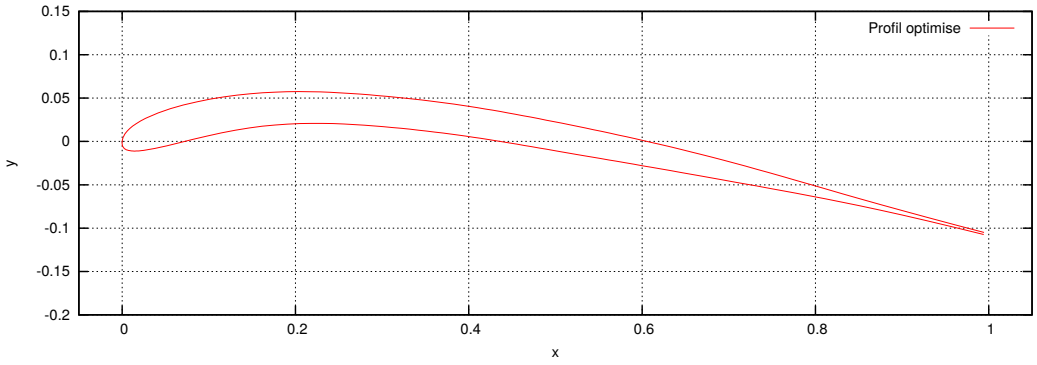


FIGURE 8.3 – Profil optimisé à l'angle d'attaque optimisé $(6,0938°)$

	Profil optimisé
Angle d'attaque	$+6,0938°$
Rayon de courbure extrados	$-4,7\%$ $(=-0,0157)$
Rayon de courbure intrados	0% $(=0,0164)$
Hauteur extrados	$+40,7\%$ $(=0,0844\text{m})$
Hauteur intrados	-182% $(=0,0492\text{m})$
$X_{Hauteur}$ extrados	$7,4\%$ $(=0,3472\text{m})$
$X_{Hauteur}$ intrados	$+10,2\%$ $(=0,4376\text{m})$
Pente tangente extrados	$+7.5\%$ $(=8,486°)$
Pente tangente intrados	-200% $(=7,911°)$

TABLE 8.3 – Paramètres de formes du profil optimisé

Conclusion

Cette étude a permis de mettre en place une boucle d'optimisation automatique de forme. A partir du modèle initial du *NACA0012*, les déformations suivant des paramètres donnés et l'analyse ses performances avec XFOIL ont été automatisées.

La toolbox FAMOSA a été intégrée dans cette boucle.

Cette application constitue un premier exemple simple de couplage du modeleur géométrique avec des outils de simulation numérique et d'optimisation. Elle a permis de valider la capacité d'exploration de l'espace des formes par le modeleur en ne considérant qu'un nombre réduit de paramètres métiers (9 paramètres), moindre par rapport au nombre de paramètres géométriques $(20 * 2)$. Les formes obtenues dans cette étude démontrent la capacité du modeleur à générer des formes variées et valides. La figure 8.4 illustre quelques profils générés par le modeleur durant l'optimisation, à angle d'attaque nul.

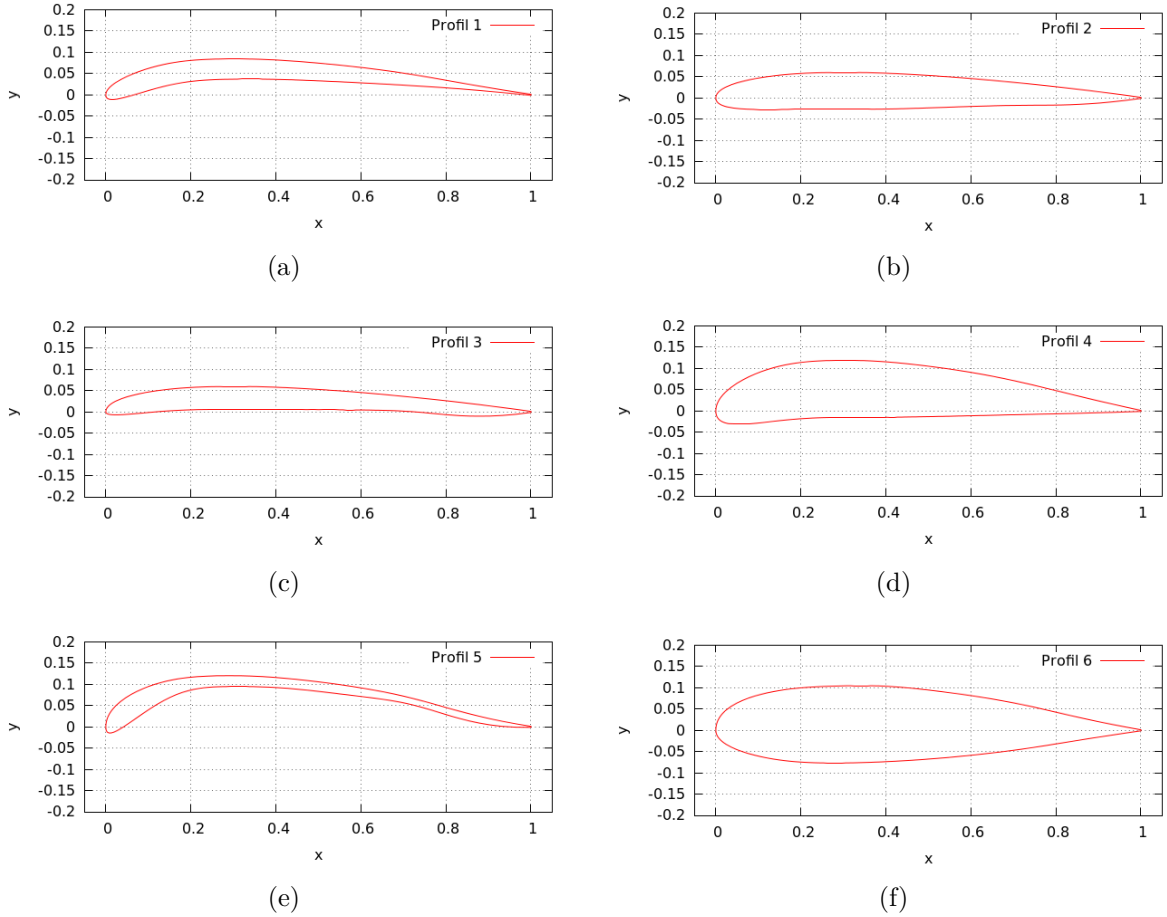


FIGURE 8.4 – Exemples de déformations du profil, à angle d'attaque nul

Cette approche permet aussi d'explorer rapidement cet espace de formes. Les résultats obtenus peuvent servir de point de départ à des méthodes plus fines mais plus coûteuses, utilisant un solveur numérique plus précis, et des modèles géométriques ayant plus de points de contrôle.

8.2 Optimisation de forme d'un foil AC45

Ces dernières années, de nouveaux bateaux ultra rapides ont été conçus en utilisant des foils. Le but d'un foil est de soulever la coque au dessus de l'eau. Ainsi, la résistance totale de la coque (traînée de friction et de vague) diminue et le bateau peut atteindre des vitesses très élevées.

Pour les bateaux de course le foil a une forme de "L", avec une partie verticale qui contre les forces générées par les voiles et une partie horizontale qui supporte le poids de la coque.

En navigation le foil permet au bateau de "voler" comme le montre la figure 8.5. Cependant, pour maintenir cet état de vol, la stabilité du foil est un aspect critique, tant pour la sécurité que pour la performance.

Les designers doivent jouer sur un grand nombre de paramètres pour produire un foil avec une traînée faible et une grande stabilité.

Nous considérons ici l'exemple du foil AC45. Ce type de foil est dit "one-desing", c'est à dire que tous les bateaux AC45 ont le même design unique du foil.

Nous cherchons à optimiser la forme de l'AC45 avec pour objectif de diminuer sa traînée totale tout en augmentant sa stabilité et sa facilité d'utilisation en navigation. Les performances du foil sont calculées en utilisant le solveur fluide parfait *AVANTI* couplé à *XFOIL*.

Le foil AC45 est actuellement utilisé par l'équipe Groupama Team France dans la 35^{ieme} Coupe de l'America. La figure 8.5 montre le bateau en train de voler grâce au foil. Un des foils est dans l'eau (à droite) et l'autre est en position rétractée (à gauche) sous la coque.

FIGURE 8.5 – Illustration de l'utilisation du foil AC45 sur le bateau de course de Groupama Team France, *Credit : ® Eloi Stichelbaut / Groupama Team France*

Dans cet exemple, nous avons réalisé une boucle d'optimisation automatique de forme complète, telle que décrite dans le chapitre 7. Le modeleur paramétrique a été couplé au solveur fluide parfait AVANTI, décrit ci-dessous, et à la toolbox d'optimisation FAMOSA, développée à l'Inria par l'équipe de recherche *Opale* [Zerbinati et al., 2011, Duvigneau and Chandrashekar,

2012]. Nous avons ainsi pu réaliser des milliers de calculs de façon automatique pour converger vers des formes de foil optimales.

8.2.1 Simulations avec AVANTI

AVANTI [Durand et al., 2014, Lothodé et al., 2013], le solveur fluide utilisé dans l'étude est développé et commercialisé par la société K-Epsilon.

La méthode utilisée ici est de type *Vortex Lattice Method (VLM)* avec une résolution du sillage .

Le foil est représenté avec un nombre fini d'éléments, correspondant aux sections définies dans le squelette. Chaque section a une forme de profil. On calcule pour chaque élément une vitesse locale, un nombre de Reynolds local et un angle d'attaque. Chaque élément a une base de données associée calculée par XFOIL contenant la valeur des coefficients de portance et de traînée pour une plage d'angles d'attaque, en général entre $-5°$ et $20°$.

AVANTI utilise cette base de données pour estimer les coefficients de portance et de traînée de chaque élément en fonction de l'angle d'attaque local calculé. La portance est ensuite convertie en une vorticité locale. Le sillage est initialisé avec le gradient de la vorticité, puis il est résolu. Ces étapes sont répétées jusqu'à convergence grâce à une méthode itérative directe, qui est en mesure de trouver une solution stationnaire.

Comme fichiers d'entrée, AVANTI se base sur :

- une description des objets par des séries de nuages de points pour les sections (même fichiers qu'XFOIL en 2D)
- un fichier qui décrit le positionnement 3D des sections avec un point d'attache et un quaternion.

Ces fichiers sont générés automatiquement par le modeleur paramétrique. L'utilisation d'AVANTI ne pose donc pas de problème de compatibilité dûe au maillage, c'est pourquoi il est possible de l'automatiser complètement.

Dans notre cas pour l'étude de l'AC45, seule la partie du foil sous l'eau est simulée. L'influence de la surface libre est prise en compte avec un modèle d'anti-symétrie. Ce modèle est une approximation satisfaisante pour de grandes vitesses. Comme [Faltinsen, 2006] le suggère, avec un nombre de Froude supérieur à 1, une condition de surface libre correspondant à un nombre de Froude infini peut être utilisée. Dans notre cas, le nombre de Froude est de l'ordre de 5 et il est calculé par la formule suivante :

$$F_r = \frac{v}{\sqrt{g\, corde}}$$

où v est la vitesse ($11,32\ m/s$), g est accélération de la pesanteur ($9,81\ m/s^2$) et la corde est la valeur moyenne de la corde le long du foil ($0.44\ m$).

La figure 8.6 illustre le sillage de l'AC45 calculé par AVANTI et les lignes de courant. La ligne de portance est située à 25% de la corde depuis le bord d'attaque le long du foil. Avec la légende de coloration de la vorticité, il est possible d'identifier les zones de l'AC45 qui génèrent le plus de force de poussée permettant de soulever le bateau : le coude et le tip.

Le repère de référence est défini comme : X est orienté dans le sens opposé au flux, Z est dans la direction verticale orientée vers le haut du foil et Y est horizontal au foil et perpendiculaire à X.

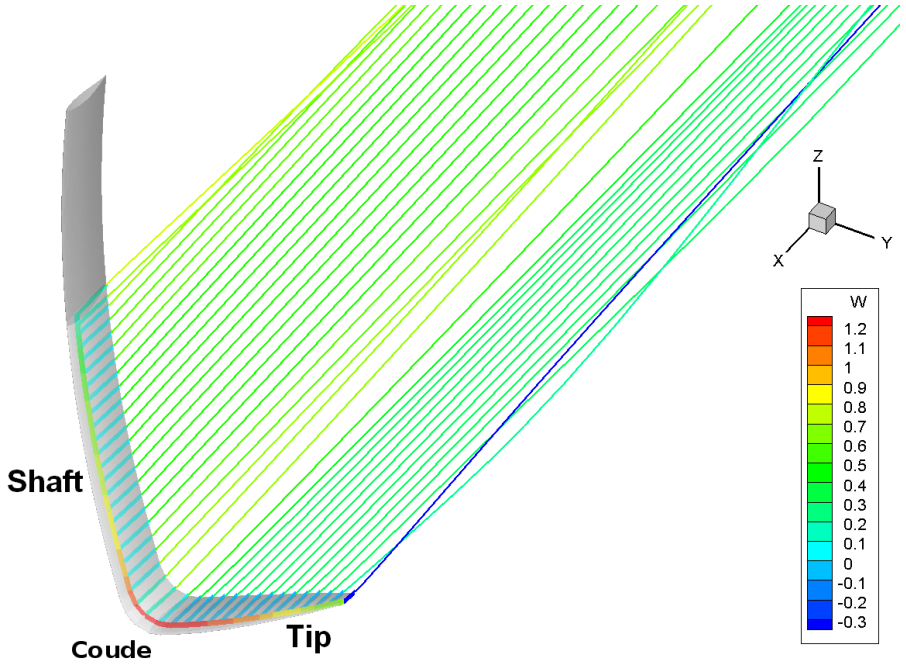


FIGURE 8.6 – Illustration du sillage et de la ligne de portance de l'AC45 calculés avec AVANTI

8.2.2 Critères de performance

Nous avons choisi trois critères calculés par AVANTI pour définir les performances du foil :

1. Un critère de performance global : la traînée totale du foil F_x dans le repère de référence. Une traînée faible permet d'augmenter la vitesse finale du bateau.
2. Un critère de stabilité : représenté par $\frac{\partial F_z}{\partial z}$, où F_z est la force totale en z du foil. Le but de ce critère est d'assurer que le bateau restera à une hauteur z fixée grâce au foil capable de s'auto-ajuster en équilibrant la force F_z qu'il produit en fonction des mouvements en z du bateau.
3. Un critère de stabilité et d'utilisation : représenté par $\frac{\partial rake}{\partial V}$, où le *rake* est l'angle d'incidence du foil suivant l'axe de rotation Y, et V est la vitesse du bateau. Le rake est un réglage que l'équipage doit ajuster pendant la navigation, afin de modifier la force verticale F_z. Donc il est important que la forme du foil soit capable de garder la valeur du rake inchangée quand la vitesse varie.

Les simulations sont paramétrées avec une force en Y, F_y, fixée qui correspond à l'opposé des forces appliquées par les voiles sur le bateau. La force en Z, F_z, est aussi fixée et correspond à l'opposé du poids du bateau. La vitesse du bateau est fixée à 22 noeuds. AVANTI résout les angles de dérive et de rake, jusqu'à ce que les forces calculées convergent vers les forces

imposées.

F_x est calculée durant la simulation, l'objectif est de le diminuer autant que possible. Dans le repère de référence, F_x est orientée selon l'axe x négatif. Donc le signe de F_x sera négatif, mais nous allons considérer sa valeur absolue pour mesurer la performance du foil.

Le second critère $\frac{\partial F_z}{\partial z}$ est estimé avec des différences finies. C'est à dire que pour un petit déplacement Δz la variation de force F_z est calculée. Pour être stable, le foil doit générer une force F_z opposée à la direction de la variation en z du bateau. Le ratio $\frac{\partial F_z}{\partial z}$ doit donc être négatif et le plus grand possible.
Par exemple, si la bateau se soulève trop au dessus de l'eau alors la force F_z générée par le foil doit diminuer, et inversement. De cette façon le foil permet au bateau de se maintenir dans une position stable.

Le troisième critère, $\frac{\partial rake}{\partial V}$, est aussi évalué avec des différences finies en calculant l'angle de rake obtenu pour des petites variations de vitesse ΔV. Dans ce cas, l'angle de rake doit varier le moins possible quand la vitesse augmente. Le ratio $\frac{\partial rake}{\partial V}$ doit donc être positif et le plus petit possible.

Dans les deux cas, les pas Δz et ΔV utilisés pour les différences finies ont été validés afin que l'approximation soit satisfaisante.

Le but de notre étude est de diminuer la traînée totale du foil AC45 le plus possible tout en conservant les critères de stabilité aussi grands que possible.

8.2.3 Déformations

Nous distinguons deux catégories de paramètres de variation de forme : les paramètres globaux et les paramètres locaux. Les paramètres globaux correspondent aux paramètres de la génératrice, et ceux locaux aux paramètres des sections du squelette.

Les paramètres globaux choisis sont : longueur du *tip*, l'angle entre le *tip* et le *shaft* (angle d'ouverture) et l'angle de *cant*, illustrés sur la figure 8.7. Nous considérons l'angle de *cant* comme un paramètre de design et non comme un réglage de navigation.
Les sections du squelette du foil définissent des profils. Les paramètres locaux que nous avons choisis sont la longueur de corde et l'angle d'attaque, appelé *twist* pour un foil, illustrés par la figure 8.7.

Pour générer une nouvelle géométrie depuis la CAO originale, le modeleur paramétrique prend en moyenne 12 secondes pour construire le squelette, puis 5.1 pour la déformation de la génératrice et 5 secondes pour la déformation de toutes les sections.
AVANTI nécessitant en entrée un nuage de points distribués sur les sections du squelette, il n'est pas nécessaire de reconstruire de surface.
En moyenne, une simulation AVANTI prend 20 secondes pour calculer les trois critères de performance et post-traiter ces résultats.

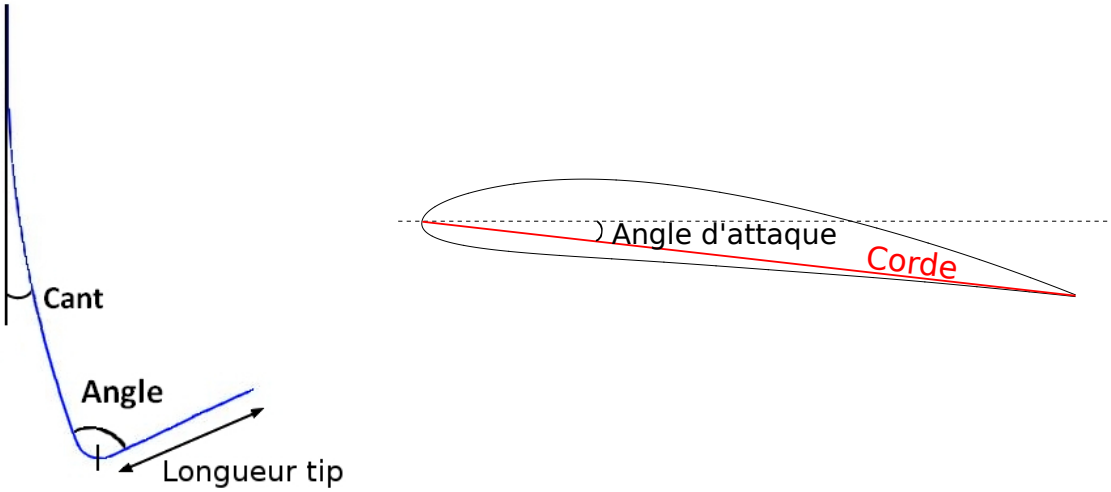


FIGURE 8.7 – Paramètres de forme globaux (gauche) et locaux (droite) du foil

Le squelette du foil AC45 est illustré dans la figure 8.8.

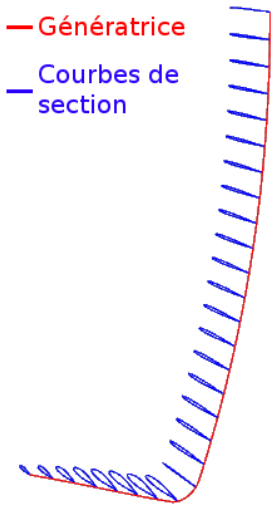


FIGURE 8.8 – Squelette du foil

Les paramètres locaux, corde et twist, sont contrôlés par des *courbes de répartition*. Les sections 4.2.1 et 5.2.4 introduisent la définition de ces fonctions et la stratégie de déformation. La figure 8.9 montre la courbe de répartition de la corde du foil avec 5 points de contrôle avec un exemple de déformation. Nous imposons de décrire ces courbes de répartition avec 5 points de contrôle chacune. Le nombre de degré de liberté total pour les deux paramètres est donc de 10, seule la position en y de chaque point de contrôle pouvant être modifiée. Le modèle initial comprenait 56 degrés de liberté, étant donné que le squelette du foil est composé de 28 sections.

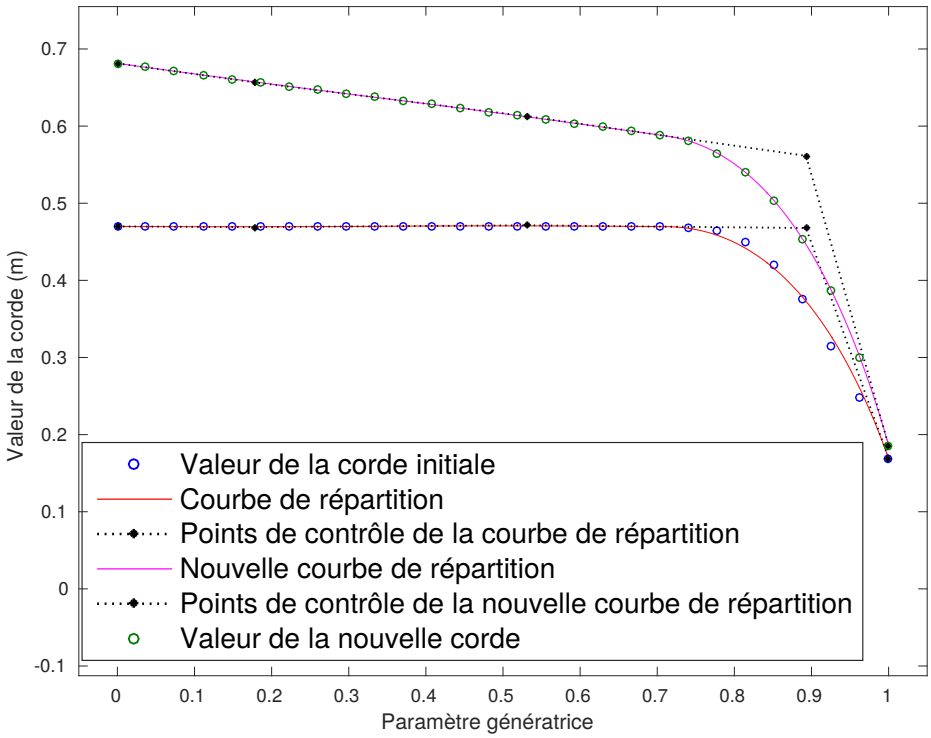


FIGURE 8.9 – Modification de la courbe de répartition de la corde le long du foil, nouvelles valeurs de la corde

Les variations initiales des paramètres sont distribuées dans un espace des paramètres défini dans le tableau 8.4, et les variations extrêmes des paramètres globaux sont illustrées dans la figure 8.10.

	Longueur du Tip	**Angle**	**Cant**	**Corde**	**Twist**
Valeur initiale	1.37m	77.24°	2.42°	0.44m (moyenne)	0°
Variation min	−30% (= 0.96m)	−30% (= 54.1°)	−313.7% (= −5.2°)	−50% (0.22m)	−10°
Variation max	+40% (= 1.92m)	+20% (= 92.65°)	+727.2% (= 20°)	+50% (0.66m)	10°

TABLE 8.4 – Bornes de l'espace des paramètres de forme du foil

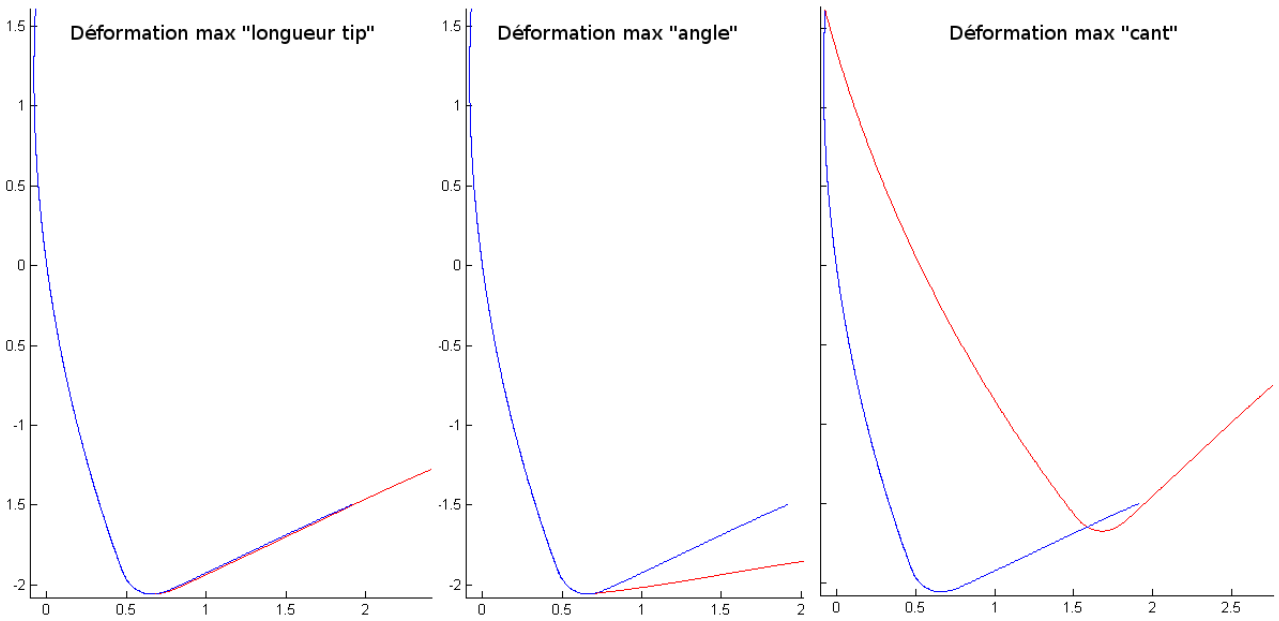


FIGURE 8.10 – Illustrations des variations de forme globales du foil dans l'espace des paramètres choisis

8.2.4 Résultats

Il n'existe pas de solution unique d'un problème multi-objectifs, mais une multitude de solutions situées sur un *front de Pareto,* qui correspondent aux meilleurs compromis entre toutes les fonctions objectif (voir section 7.3.2 pour une explication détaillée des algorithmes multi-objectifs).

Le but de cette étude est de déterminer deux fronts Pareto : l'un pour le couple de fonctions objectif (F_x, $\frac{\partial F_z}{\partial z}$) et l'autre pour le couple (F_x, $\frac{\partial rake}{\partial V}$). L'algorithme actuellement implémenté dans FAMOSA n'inclut que deux fonctions objectif à la fois, c'est pourquoi nous divisons notre recherche de forme optimale en deux fronts de Pareto.

Nous nous basons sur les bornes initiales de l'espace des paramètres pour réaliser une exploration du domaine de variation des fonctions objectif. Pour cela, nous distribuons 20 points suivant un Hypercube Latin [McKay et al., 1979, Iman et al., 1981].

Cette première distribution, que nous pouvons appeler plan d'expérience, nous aide à situer l'espace de variations des paramètres, afin d'initialiser l'algorithme de recherche du front de Pareto avec des points intéressants.

Ensuite, nous utilisons un algorithme PAES, expliqué dans la section 7.3.2, pour identifier le front de Pareto pour les couples de fonctions objectif (F_x, $\frac{\partial F_z}{\partial z}$) et ($F_x$, $\frac{\partial rake}{\partial V}$). Pour obtenir la totalité du front de Parto, nous avons relancé plusieurs fois l'algorithme avec des points de départ différents.

L'algorithme PAES n'est pas borné, donc les limites proposées dans le tableau 8.4 ne sont plus respectées. Relancer l'algorithme PAES près d'un point précédemment identifié comme étant sur le front de Pareto éloigne de plus en plus le front du domaine de variation initial. Il s'agit donc de trouver un compromis entre la convergence vers le front et l'éloignement de l'espace des paramètres fixé.

Résultats pour $(F_x, \frac{\partial F_z}{\partial z})$

Le front de Pareto du couple de fonctions objectif $(F_x, \frac{\partial F_z}{\partial z})$ est présenté dans la figure 8.11. Le point orange triangulaire est la référence des valeurs de F_x et $\frac{\partial F_z}{\partial z}$ de l'AC45. Les points bleus sont les points obtenus avec l'Hypercube Latin lors de l'exploration du domaine. Les points verts représentent les points testés par l'algorithme PAES pour converger vers le front de Pareto. Enfin, les points rouges sont les points situés sur le front.

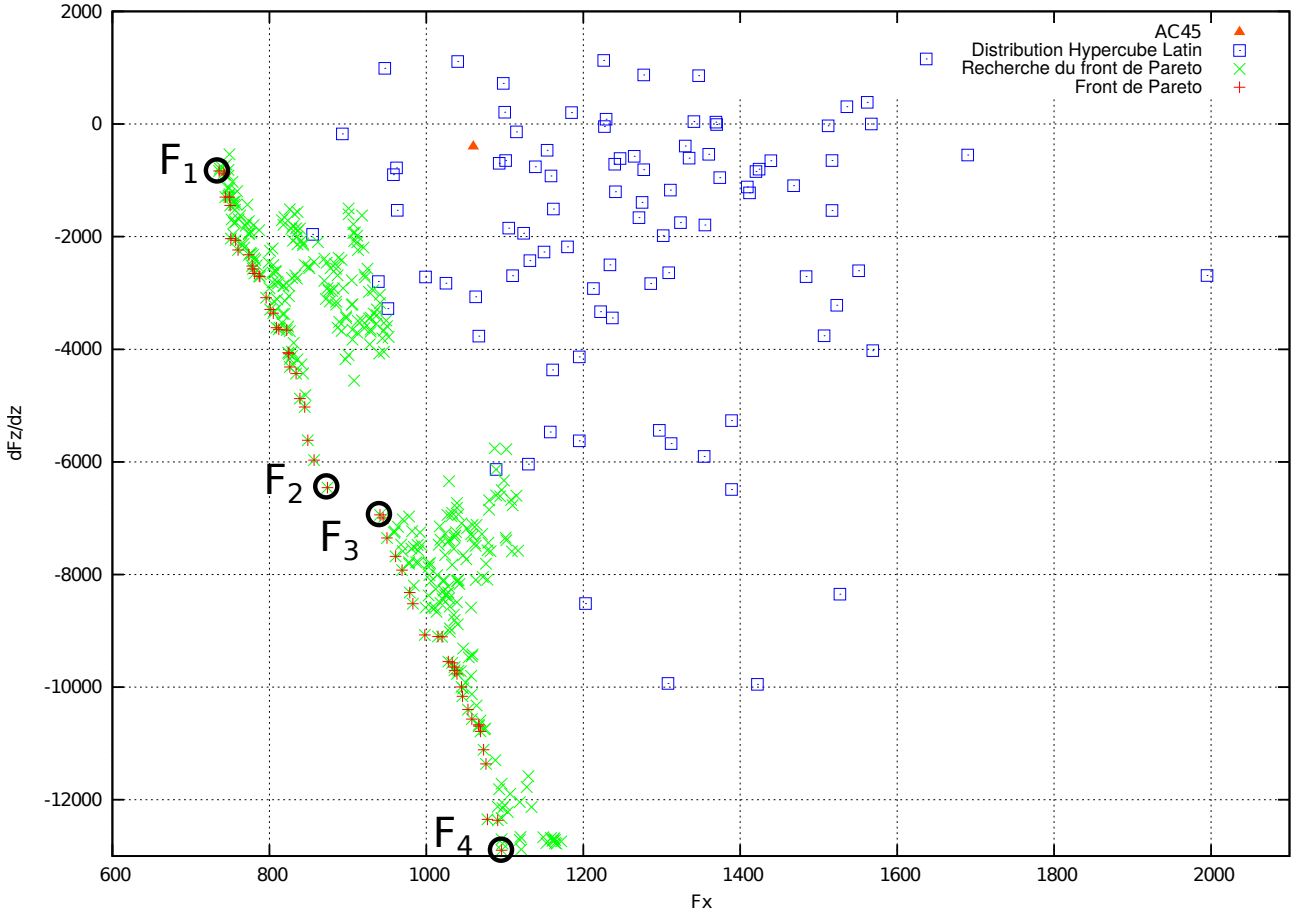


FIGURE 8.11 – Front de Pareto $(F_x, \frac{\partial F_z}{\partial z})$

Nous illustrons 4 formes de foils sur le front de Pareto, F_1, F_2, F_3, F_4 dans la figure 8.13 à la fin de la section, et détaillons les valeurs des paramètres dans le tableau 8.5.

On observe que les formes obtenues sur le front sont plus minces que le foil original, le tip est plus long et l'angle de cant est nettement plus élevé. L'angle d'ouverture est aussi plus grand que celui original. Les tendances sont nettes dans l'évolution des fonctions objectif selon la variation des paramètres :

- la longueur du tip est systématiquement plus grande pour les points sur le front de Pareto,
- l'angle entre le tip et le shaft a tendance à augmenter pour diminuer la traînée,

- la stabilité augmente avec l'angle de cant,
- la traînée diminue avec la diminution de la corde le long du foil,
- la stabilité augmente avec l'augmentation du twist le long du foil

Les foils obtenus sont tous géométriquement valides, mais leurs propriétés structurelles sont discutables. Des foils aussi fins peuvent conduire à des défaillances structurelles.

#	% variation Longueur Tip	% variation d'angle	% variation de Cant	% variation moyenne de corde	Valeur moyenne du twist	Traînée totale ($\|F_x\|$) en N	$\frac{\partial \mathbf{F_z}}{\partial \mathbf{z}}$
AC45	0%	0%	0%	0%	0°	1077	-423
F_1	+29%	+35%	+596%	$-13,8\%$	$-1,15°$	736	-833
F_2	+38%	+18%	+684%	-14%	$-0,06°$	874	-6457
F_3	+24%	+24%	+885%	+8.4%	$7,5°$	942	-6941
F_4	+30%	+3.5%	+985%	+11.6%	$7,62°$	1188	-15071

TABLE 8.5 – Paramètres de forme et valeurs des critères de performance pour les foils sur le front de Pareto (F_x, $\frac{\partial F_z}{\partial z}$).

Résultats pour (F_x, $\frac{\partial rake}{\partial V}$)

Le front de Pareto du couple de fonctions objectif (F_x, $\frac{\partial rake}{\partial V}$) est présenté dans la figure 8.12. Les codes couleurs sont identiques à ceux de la figure 8.11.

Nous présentons 4 formes de foils sur le front de Pareto, F_5, F_6, F_7, F_8 dans la figure 8.14 à la fin de la section, qui illustrent la tendance du front de Pareto. Les valeurs des paramètres sont détaillées dans le tableau 8.6.

#	% variation Longueur Tip	% variation d'angle	% variation de Cant	% variation moyenne de corde	Valeur moyenne du twist	Traînée totale ($\|F_x\|$) en N	$\frac{\partial rake}{\partial V}$
AC45	0%	0%	0%	0%	0°	1077	1,033
F_5	+39%	+42%	+576%	-38%	$6,17\circ$	595	0,929
F_6	+31%	+48%	+587%	$+0,023\%$	$3,77°$	755	0,631
F_7	+43%	+48%	+856%	$0,09\%$	$1,78°$	890	0,444
F_8	+75%	+30%	+380%	+13%	$3,46°$	1005	0,393

TABLE 8.6 – Paramètres de forme et valeurs des critères de performance pour les foils sur le front de Pareto (F_x, $\frac{\partial rake}{\partial V}$).

Comme pour le couple de fonctions objectif (F_x, $\frac{\partial F_z}{\partial z}$), la longueur du tip et l'angle de cant des foils sur le front de Pareto sont nettement plus élevés que pour l'AC45. Dans le cas du couple de fonctions objectif (F_x, $\frac{\partial rake}{\partial V}$), l'angle d'ouverture est aussi nettement plus élevé.

On observe que les formes qui minimisent le plus la traînée sont très minces, et celle qui minimisent $\frac{\partial rake}{\partial V}$ sont épaisses.

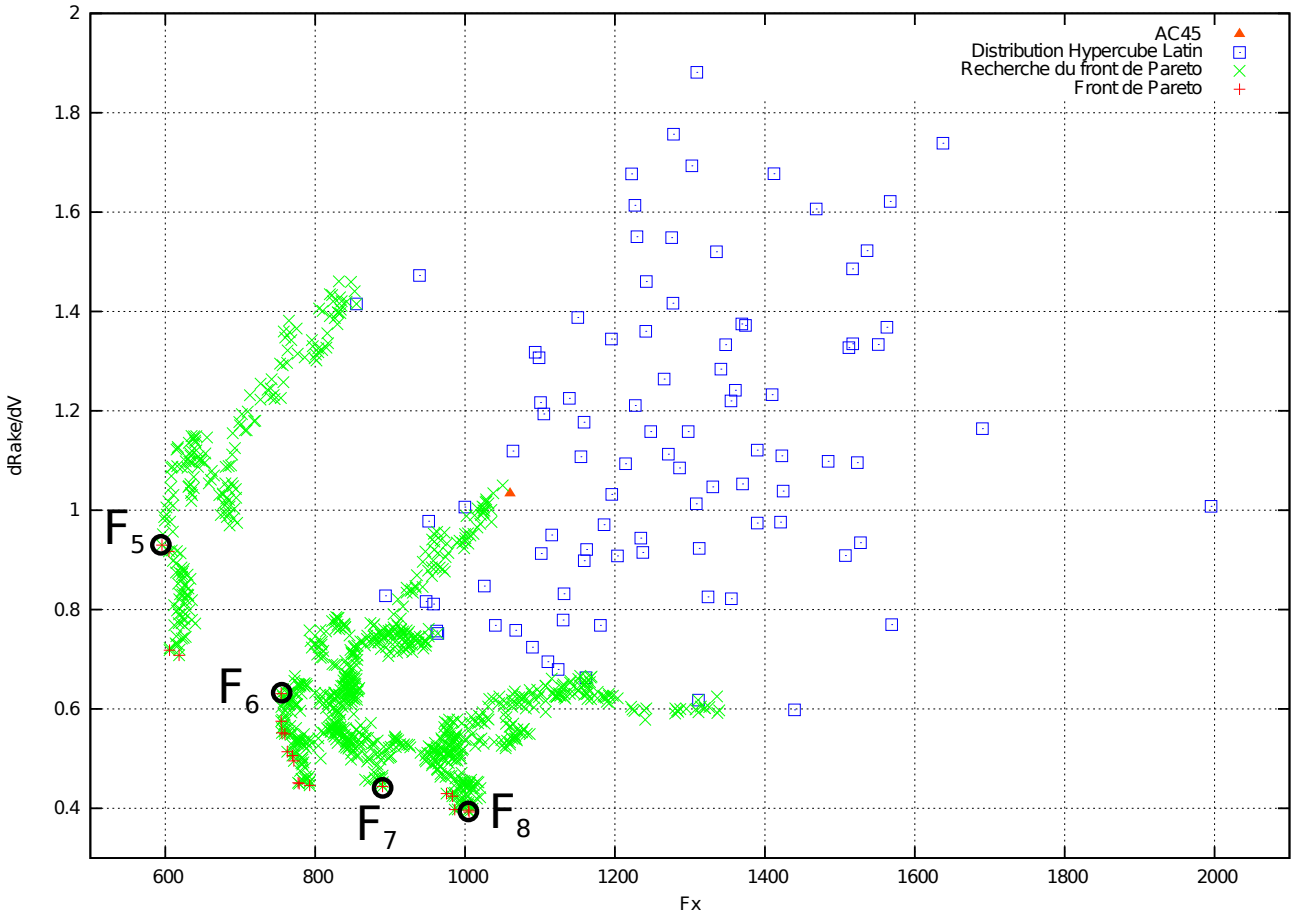


FIGURE 8.12 – Front de Pareto F_x, $\frac{\partial rake}{\partial V}$

Conclusion

Cette étude a permis de mettre en place une boucle d'optimisation automatique de forme. A partir du modèle initial du foil AC45 importé en IGES sur Rhinocéros 3D^{TM}, toutes les taches pour obtenir le squelette, déformer le foil suivant des paramètres donnés, analyser ses performances avec AVANTI et XFOIL ont été automatisées.

La toolbox FAMOSA a été intégrée dans cette boucle, permettant d'obtenir les fronts de Pareto associés aux critères de performance choisis.

Les comportements observés à travers cette étude sont cohérents par rapport aux résultats attendus, certaines tendances étant bien connues par les designers (par exemple, augmenter l'angle de cant permet d'augmenter la stabilité).

D'autres tendances ont été observées et analysées, permettant de mettre en évidence de nouveaux designs intéressants et potentiellement innovants.

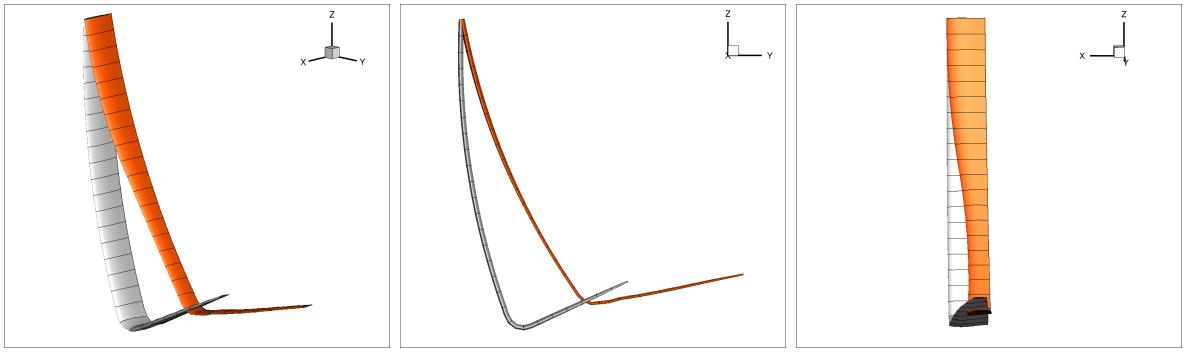
Cependant, nous avons observé que certains foils générés, bien que valides d'un point de vue géométrique et hydrodynamique, sont impossibles à construire car ils sont trop minces. La création de telles formes est tout d'abord dûe au fait que l'algorithme d'optimisation utilisé ne tient pas compte des bornes de variations des paramètres, et au fait que nous ne pénalisons pas les formes qui s'écartent trop de ces limites.

Pour obtenir des formes réalistes, il est préférable d'inclure des critères physiques

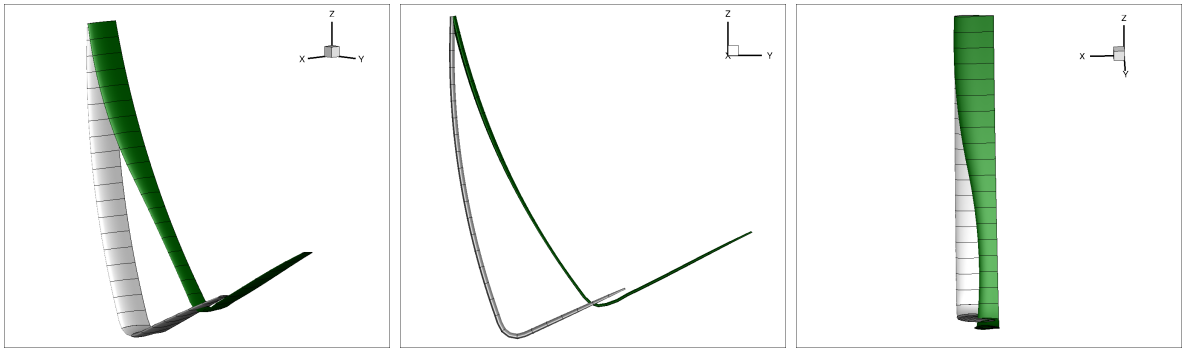
supplémentaires plutôt que d'essayer de limiter les formes non valides par des contraintes géométriques. Un solveur structure qui calcule les contraintes dans le foil généré pourrait être intégré dans la boucle. D'un point de vue optimisation, cela pourrait revenir à ajouter des contraintes pour restreindre les formes sur un espace où les performances structurelles des foils sont viables.

L'étude que nous avons menée sur l'optimisation de forme d'un foil peut être encore approfondie en incluant d'autres fonctions objectif. Il serait intéressant d'inclure l'enfoncement comme paramètre de forme, ainsi que la forme des sections (corde, twist, etc.). Pour les fonctions objectif, il serait intéressant d'inclure M_x, le moment en x du bateau. Le moment a une influence sur les performances générales du foil, et l'angle de cant peut être significativement modifié afin de trouver une position qui permet de contrer M_x.

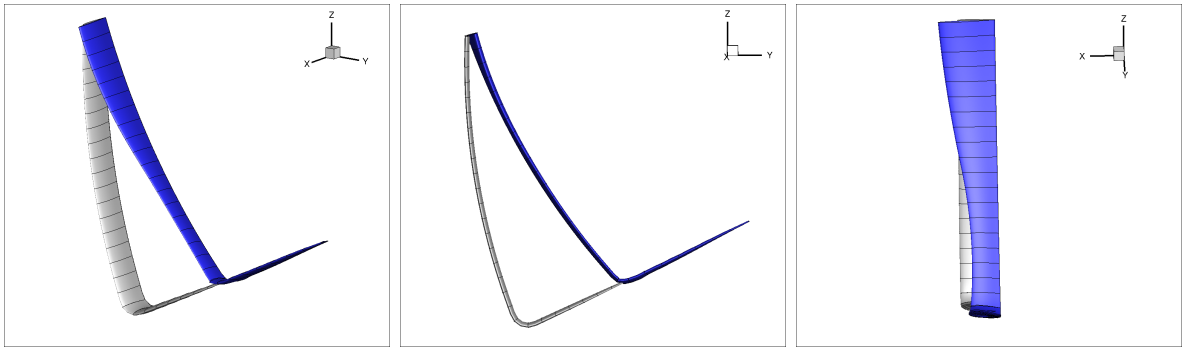
L'utilisation des foils est limitée par l'apparition du phénomène de *cavitation*. La cavitation est l'apparition de bulles de vapeur causée par la baisse de pression importante dans l'eau induite par l'action du foil. La cavitation endommage le foil, mais surtout elle cause des pertes de performances : le foil peut perdre sa force de portance et faire plonger le navire. Il est possible de prédire les conditions d'apparition de la cavitation, il est donc aussi possible d'inclure des contraintes pour éviter de générer des formes de foil propices à ce phénomène.



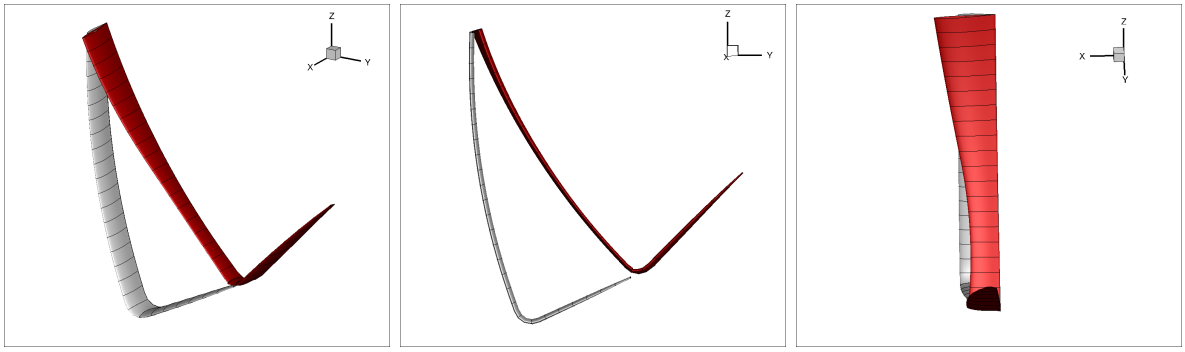
(a) Vues du foil F_1



(b) Vues du foil F_2

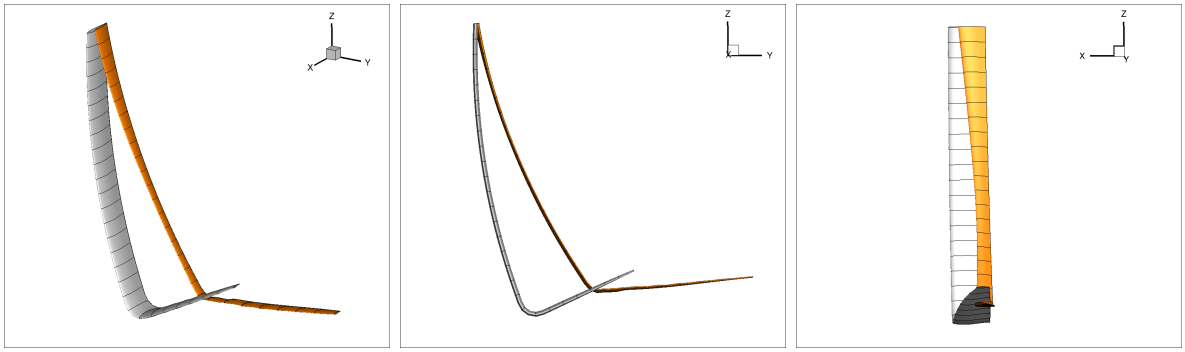


(c) Vues du foil F_3

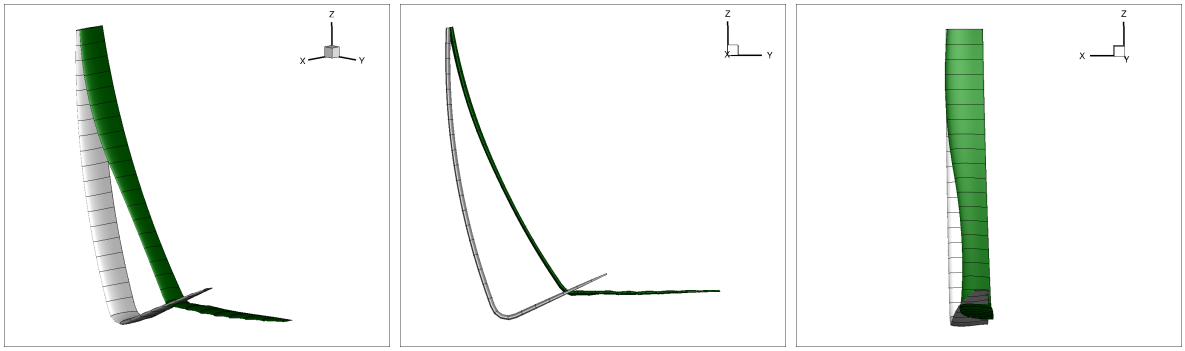


(d) Vues du foil F_4

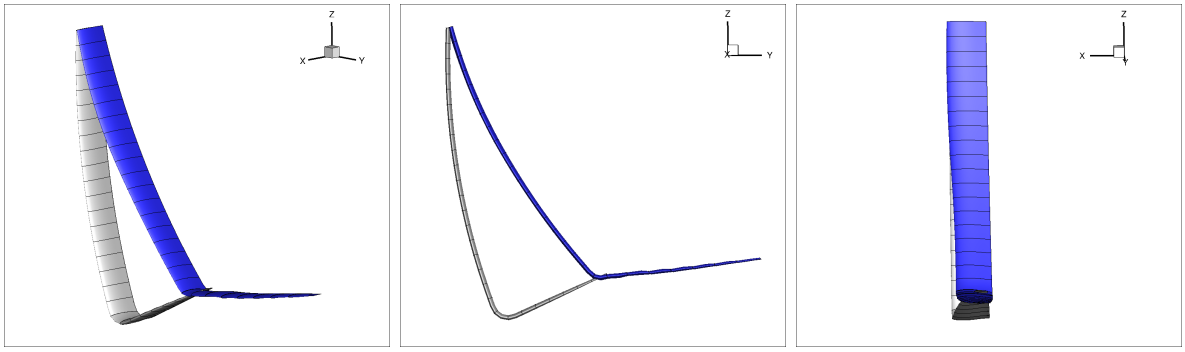
FIGURE 8.13 – Vues des foils sur le front de Pareto (F_x, $\frac{\partial F_z}{\partial z}$)



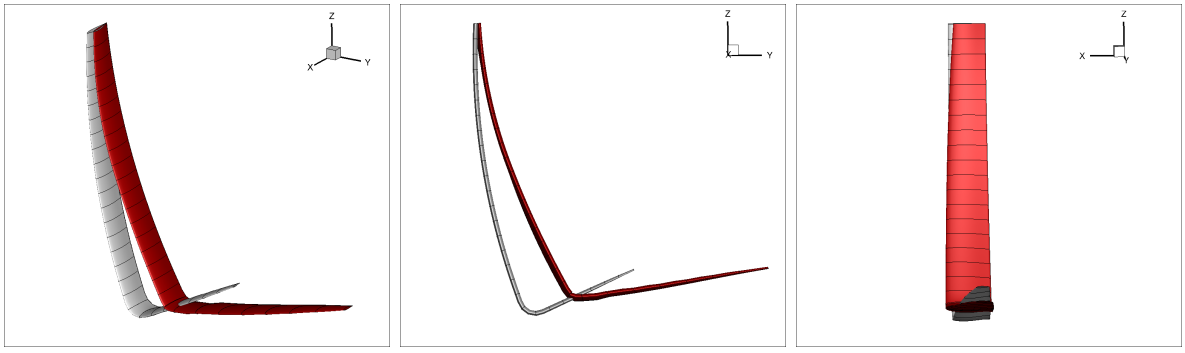
(a) Vues du foil F_5



(b) Vues du foil F_6



(c) Vues du foil F_7



(d) Vues du foil F_8

FIGURE 8.14 – Vues des foils sur le front de Pareto (F_x, $\frac{\partial rake}{\partial V}$)

8.3 Optimisation de forme d'un bulbe

Nous présentons ici une application du modeleur paramétrique sur la déformation du bulbe d'un chalutier de pêche.

Le chalutier original, illustré par la figure 8.15 ne présente pas de bulbe. En ajoutant un bulbe, l'objectif est de réduire la traînée du navire lorsqu'il navigue à sa vitesse de croisière pour aller sur le site de pêche et pour en revenir.

FIGURE 8.15 – *L'oiseau des îles*, architecte : Profils (Eric Jean)

Un bulbe initial a été dessiné par un architecte naval, puis nous proposons de faire varier sa forme suivant trois paramètres : la longueur, la largeur du bulbe et l'angle par rapport à la ligne de quille. La largeur est mesurée sur la section centrale du bulbe.

Des simulations RANSE (Reynolds-Averaged Navier–Stokes equations) ont été utilisées pour évaluer la traînée du navire.

Ces simulations étant plus complexes à mettre en place, le processus d'optimisation n'a pas été complètement automatisé pour cet exemple. La phase de maillage est difficilement automatisable et nécessite une intervention humaine pour être complétée.

De plus, le logiciel de maillage utilisé HEXPRESS^{TM} (décrit ci-dessous), est basé sur un format d'entrée propriétaire : Parasolid. Parasolid est un format de description de géométries détenu par *Siemens PLM Software*. D'autres logiciels de CAO utilisent Parasolid grâce à un système de licences mises en place par *Siemens PLM Software*, mais les exports depuis ces logiciels ne sont pas de la même qualité. En utilisant l'export de Rhinocéros 3D^{TM}, il est obligatoire de passer par un logiciel de conversion de géométrie appelé CADfix^{TM}, développé et commercialisé par *International TechneGroup Incorporated*. CADfix^{TM} n'est pas automatisable.

8.3.1 Simulations avec FINE^{TM}/Marine

Le logiciel HEXPRESS^{TM} est utilisé pour générer des maillages non conformes, entièrement hexaédriques et non structurés. Ce logiciel est développé et commercialisé par *Numeca International*, il fait partie de la suite logiciel FINE^{TM}/Marine. HEXPRESS^{TM} commence par générer une triangulation fermée des surfaces de la coque importées depuis un format *Parasolid*, puis un maillage volumique est construit tout autour en utilisant une méthode *octree*. Les techniques de lissage implémentées permettent de fournir un maillage de la couche limite de

haute qualité [Wackers et al., 2012].

Le maillage utilisé pour le chalutier est illustré par les figures 8.16 et 8.17.

La génération du maillage nécessite une CAO propre et fermée. Grâce au contrôle de forme et au lissage de surface utilisés dans le modeleur, les géométries que nous produisons sont bien adaptées pour ces contraintes et nous permettent de générer des maillages de qualité pour les simulations.

Pendant la simulation, un raffinement automatique du maillage a été utilisé. Le raffinement automatique de maillage est une méthode pour optimiser les mailles au niveau de la surface libre, en adaptant le raffinement en fonction de la déformation du champ de vague durant la simulation. Les cellules sont localement divisées en de plus petites mailles, ou elles sont fusionnées entre elles en de plus grosses mailles. Dans le cas du chalutier, le demi-maillage initial est d'environ 1.9 million de cellules et le demi-maillage final d'environ 2.2 millions de cellules.

La figure 8.16 montre une vue générale du maillage final et la figure 8.17 montre le maillage final autour de la coque, avec le raffinement automatique autour de la surface libre.

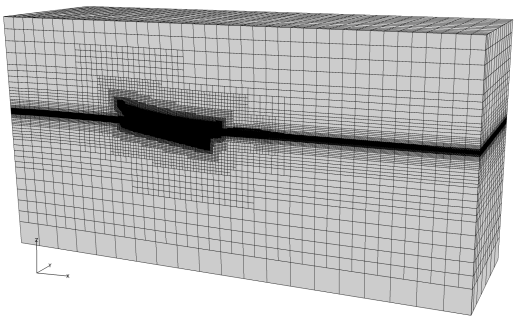


FIGURE 8.16 – Vue générale du maillage final et du domaine de calcul

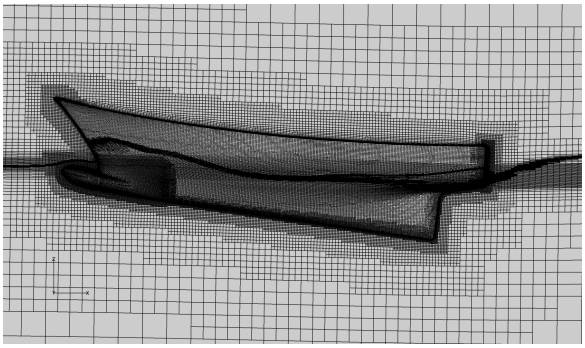


FIGURE 8.17 – Vue détaillée du navire et du raffinement du maillage final

Le solveur ISIS-CFD de la suite logiciel FINE^{TM}/Marine a été utilisé pour modéliser l'écoulement autour du chalutier.

ISIS-CFD un solveur RANSE (Reynolds-averaged Navier-Stokes Equations) incompressible et instationnaire [Duvigneau et al., 2003, Queutey and Visonneau, 2007]. ISIS-CFD utilise une méthode V.O.F. (Volume Of Fluid) pour modéliser les déformations de la surface libre [Queutey and Visonneau, 2007].

Pour modéliser la turbulence, le modèle de double équations k-ω SST est utilisé.

Pour la simulation du chalutier, l'assiette et l'enfoncement sont libres et donc résolus durant le calcul.

Le chalutier étudié a une longueur de flottaison de $22,35$ mètres et un déplacement de 150 tonnes. La vitesse pour les simulations est de 13 nœuds ($6,688$ m/s), correspondant à la vitesse du chalutier pour aller sur le site de pêche. L'assiette et l'enfoncement sont résolus, tandis que le bateau est tracté vers sa vitesse finale avec une phase d'accélération où la vitesse suit un

quart de sinusoïde. Les caractéristiques des fluides sont détaillées dans le tableau 8.7. Chaque simulation a nécessité plus de 12 heures de calculs sur 32 CPUs pour converger.

	$\rho\ (kg/m^3)$	$\mu\ (Pa.s)$
Eau	$1026,02$	$0,00122$
Air	$1,2$	$1,85 * 10^{-5}$

TABLE 8.7 – Caractéristiques des fluides utilisées pour la simulation du chalutier

8.3.2 Déformations proposées

Pour générer une nouvelle CAO depuis la géométrie originale du bulbe, le modeleur paramétrique prend en moyenne $27,6$ secondes pour construire le squelette, $14,1$ secondes pour effectuer les déformations et 20 secondes pour reconstruire la surface.

Le squelette généré pour le bulbe est illustré par la figure 8.18.

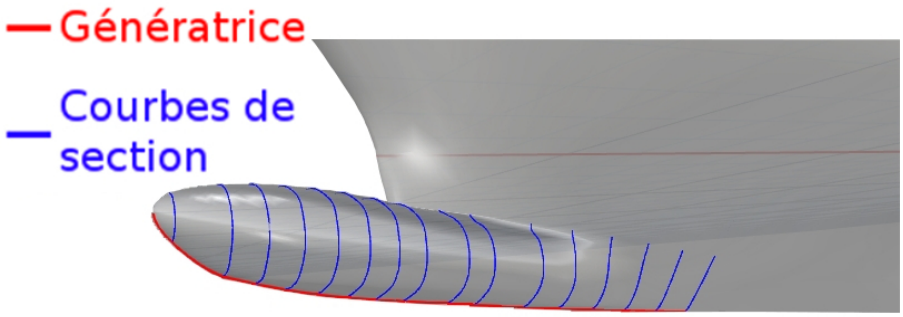


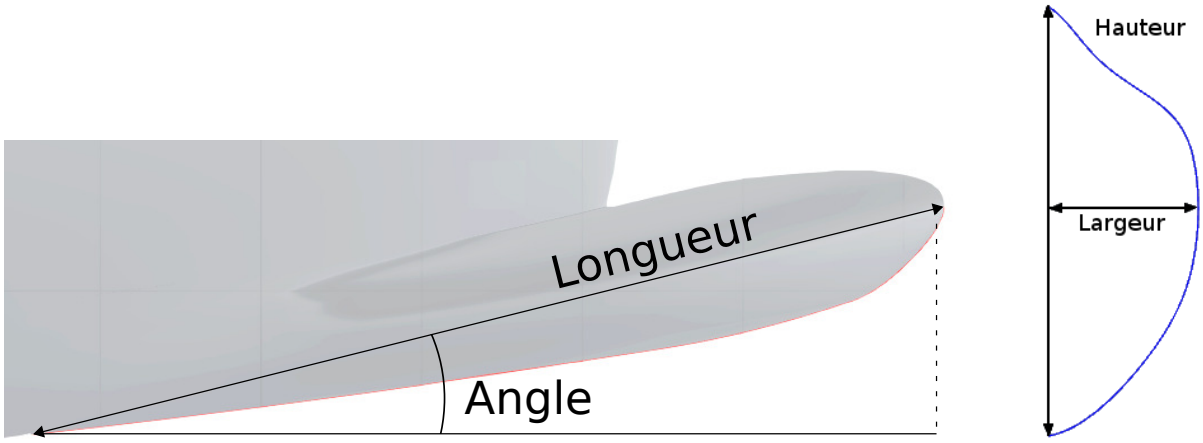
FIGURE 8.18 – Squelette du bulbe

Les paramètres de forme proposés sont la longueur, la largeur du bulbe et l'angle entre le bulbe et la ligne de quille. Les paramètres sont illustrés par les figures 8.19.
La largeur est mesurée sur la section centrale du squelette du bulbe.

Les limites de variations des paramètres sont définies dans le tableau 8.8, et les variations extrêmes sont illustrées dans la figure 8.20.

Le bulbe initial étant très court, nous avons défini la limite inférieure de la longueur du bulbe à $1,86$m (soit +40% de la longueur initiale) comme étant pertinente pour influencer positivement la traînée. La limite supérieure est fixée par la longueur maximale de l'étrave. Pour l'angle, la limite de variation est atteinte quand le bulbe sort de l'eau.

FINE^{TM}/Marine nécessite en entrée un ensemble de surfaces B-Splines qui décrivent l'objet. Ces surfaces doivent être bien raccordées (l'objet doit être complètement fermé), continues, et lisses. Pour respecter ces contraintes, nous avons développé un algorithme de reconstruction de surfaces, exposé dans le chapitre 6.
Cette étude a montré l'importance d'obtenir des surfaces de qualité, pour qu'elles puissent être utilisées facilement dans HEXPRESS^{TM} pour être maillées.



(a) Paramètres de la génératrice

(b) Paramètres des sections

FIGURE 8.19 – Paramètres du bulbe

	Longueur	**Angle**	**Largeur**
Valeur initiale	$1,61$m	$31,52°$	$0,83$m
Variation min	$+15\%$ $(= 1,86$m$)$	-25% $(= 23,64°)$	-20% $(= 0,66$m$)$
Variation max	$+90\%$ $(= 3,07$m$)$	0% $(= 31,52°)$	$+20\%$ $(0,99$m$)$

TABLE 8.8 – Bornes de l'espace des paramètres de formes du bulbe

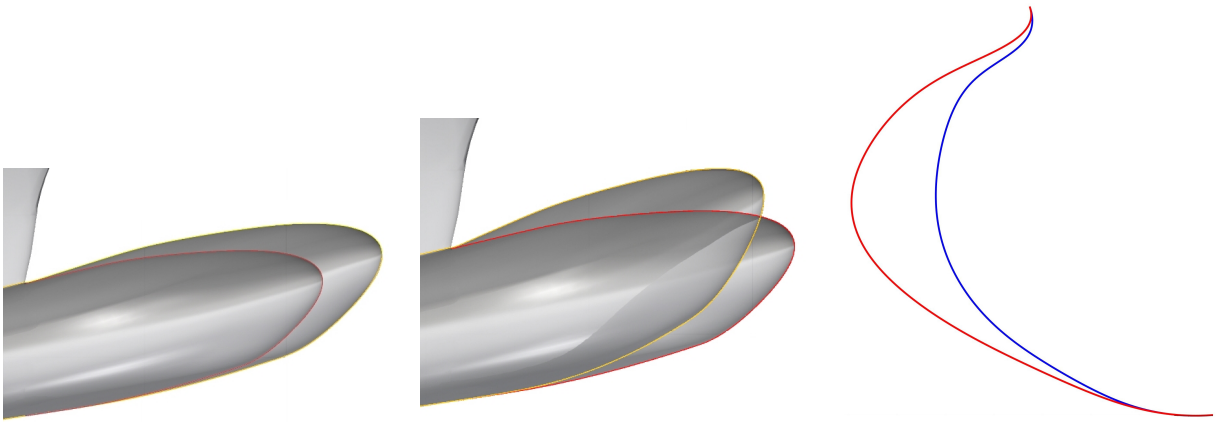
8.3.3 Résultats

Le but de cet exemple est de réaliser la première étape d'un algorithme d'optimisation avec méta-modèle : construire un *plan d'expérience*, c'est-à-dire une base de données de valeurs de la fonction objectif.

Une distribution Hypercube Latin a été utilisée pour échantillonner l'espace des paramètres, étant donné que ce type de distribution est particulièrement bien adapté pour être utilisé par des algorithmes d'optimisation type *Efficient Global Optimization* (EGO) (voir section 7.3.1).
Nous avons échantillonnés 20 points, c'est-à-dire 20 triplets de paramètres (Longueur, Angle, Largeur) dans l'espace défini dans le tableau 8.8.

Tous les points échantillonnés ont amélioré la traînée de la carène par rapport à la carène originale sans bulbe et par rapport à la carène avec le bulbe initial.
Le tableau 8.9 présente les résultats de traînée pour la carène originale sans bulbe, la carène avec le bulbe initial et la carène avec la meilleure variation obtenue dans le plan d'expérience.

La meilleure réduction de traînée est obtenue pour les valeurs de paramètres suivantes : Longueur : $+58,70\%$ $(= 2,56$m$)$; Angle : $-19,81\%$ $(= 25,28°)$; Largeur : $+9,99\%$ $(= 0,66$m$)$. La traînée F_x est décomposée suivant la force de traînée visqueuse $FVisc_x$ et celle de pression $FPres_x$.



(a) Variation de longueur (b) Variation d'angle (c) Variation de largeur

FIGURE 8.20 – Illustrations de variations de forme du bulbe dans l'espace des paramètres choisis

La meilleure variation de bulbe du plan d'expérience représente un gain de $3,64\%$ par rapport au bulbe initialement dessiné par l'architecte.

	Traînée (F_x) en N	$FVisc_x$ en N	$FPres_x$ en N	% différence
Carène originale (sans bulbe)	79910	9311	70599	-
Bulbe initial	73740	9887	63852	$7,72\%$
Meilleur bulbe	71054	10083	60970	$11,08\%$

TABLE 8.9 – Gains de traînée par rapport à la carène originale

Le bulbe impacte principalement la traînée de pression, car il réduit considérablement la vague d'étrave comme le montrent les figures 8.21.

La traînée visqueuse croît avec l'augmentation de la surface mouillée, mais cette augmentation est compensée par la diminution de la traînée de pression.

L'assiette est un facteur important pour l'évolution de la traînée car elle modifie l'écoulement autour de la coque. Lors des déformations, la distribution de volume du bulbe n'est pas conservée, mais les simulations ont été paramétrées avec le même déplacement et le même centre de gravité pour tous les designs.

Le tableau 8.10 montre l'évolution de l'assiette et de l'enfoncement des trois designs que nous analysons. L'assiette totale, respectivement l'enfoncement total, est la somme des assiettes, respectivement enfoncements, calculées à l'hydrostatique et la fin du calcul dynamique.

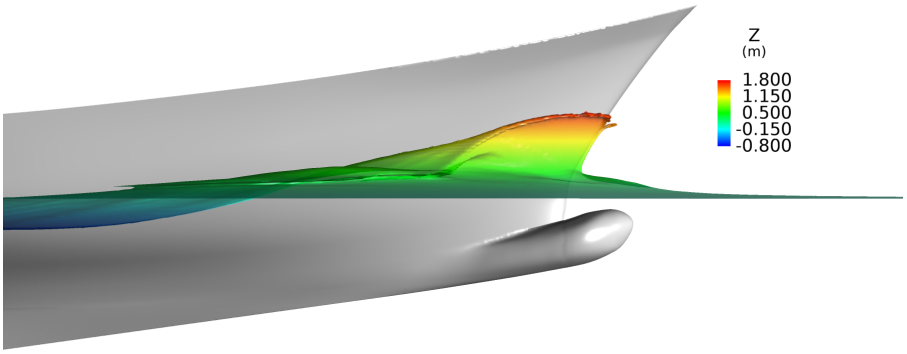
La variation totale d'assiette est relativement petite entre les différents designs. Cela montre que les résultats d'améliorations de la traînée sont réellement dus à la forme du bulbe, et non à son influence sur l'assiette.

Les figures 8.21 illustrent l'élévation de la surface libre des designs "Bulbe initial" et "Meilleur bulbe" au niveau de la vague d'étrave.

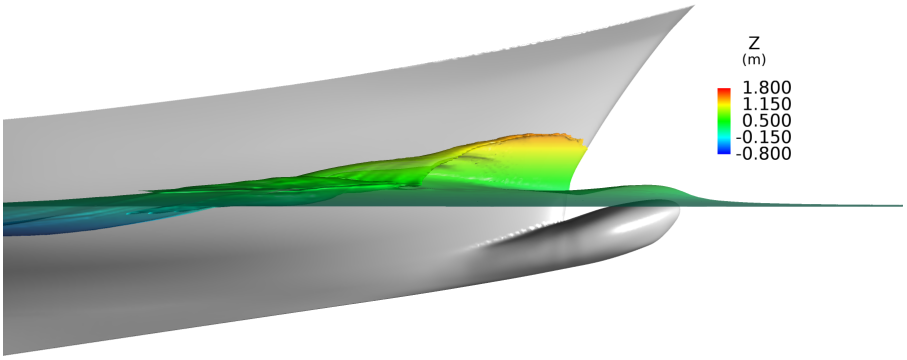
La figure 8.22 montre les champs de vagues créés par les deux mêmes carènes. On observe une diminution de la hauteur des vagues.

	Assiette totale en deg	Enfoncement total en m
Carène originale (sans bulbe)	$1,656$	$-0,257$
Bulbe initial	$1,495$	$-0,264$
Meilleur bulbe	$1,728$	$-0,225$

TABLE 8.10 – Variations d'assiette et d'enfoncement selon les designs de bulbes



(a) Élévation de la surface libre pour le bulbe initial



(b) Élévation de la surface libre pour le meilleur bulbe

FIGURE 8.21 – Élévation de la surface libre au niveau de l'étrave

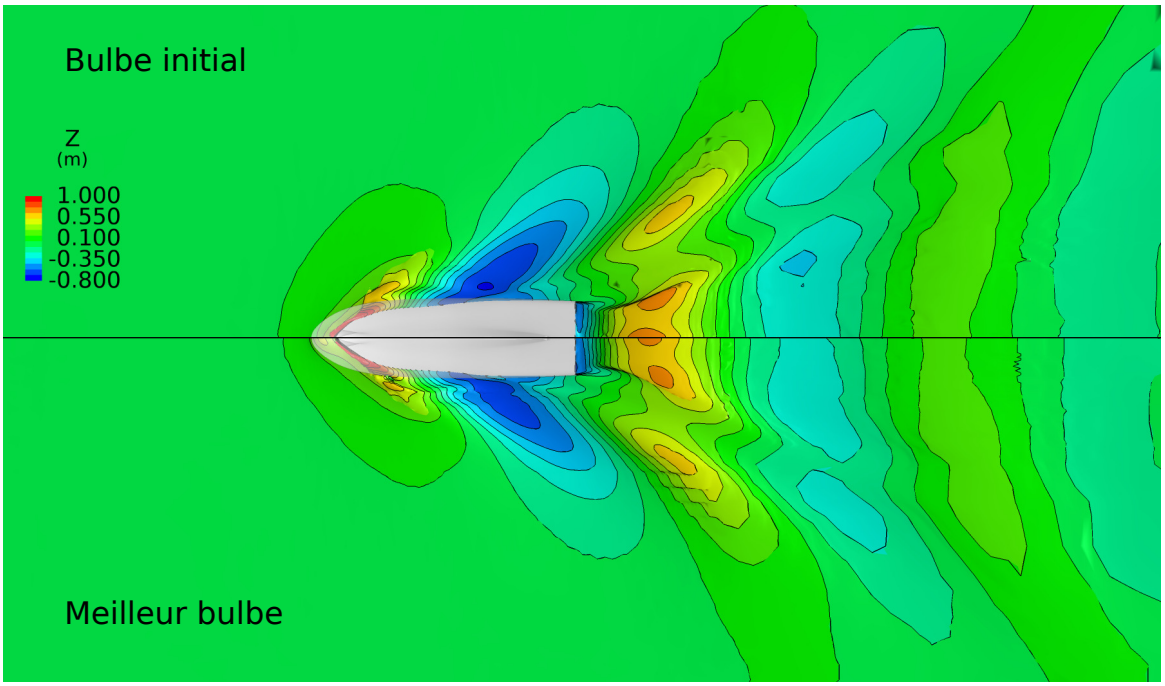


FIGURE 8.22 – Champs de vagues créés par la carène "Bulbe initial" (haut) et la carène "Meilleur bulbe" (en bas)

L'échantillonnage réalisé par l'Hypercube Latin peut être représenté graphiquement par une méthode de surface de réponse, illustrée dans les figures 8.23.

La méthode utilisée pour générer la surface de réponse s est une technique d'approximation de surface par des fonctions de base radiale. Ici, nous utilisons des fonctions de base radiale linéaires. Considérons n points d'interpolation x_i et les valeurs de la fonctions à ces points $f_i = f(x_i)$, alors la fonction d'interpolation par base radiale est donnée par :

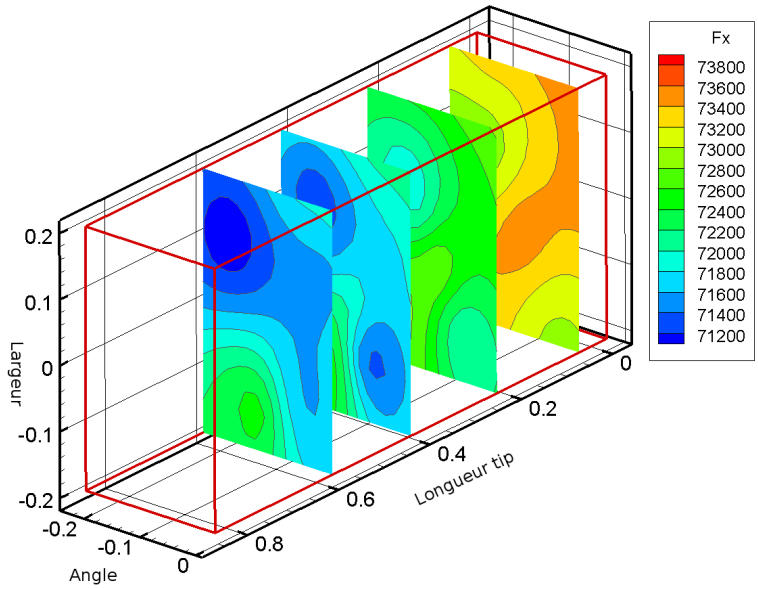
$$s(x) = \sum_{i=1}^{n} \lambda_i \Phi(\|x - x_i\|) \tag{8.3.1}$$

Dans le cas d'interpolation linéaire, $\Phi(r) = r$. Les λ_i sont des poids associé à chaque point d'interpolation, de façon à ce que $s(x_i) = f_i$.

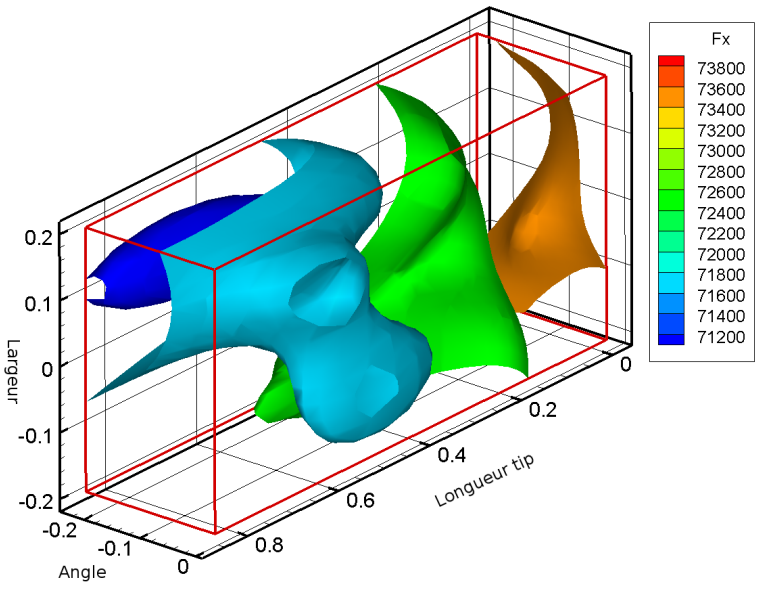
L'équation (8.3.1) revient à écrire un système linéaire de taille $n \times n$ où les λ_i sont les inconnues : $A\lambda = f$, où les éléments de la matrice A sont donnés par $a_{ki} = \Phi(\|x_k - x_i\|), k, i = 1, ..., n$, le vecteur λ contient les λ_i et le vecteur f contient les f_i.

Dans les deux figures, les variations de paramètres (Longueur, Angle, Largeur) sont représentées le long des axes, et la traînée F_x est représentée par une carte de couleur : rouge là où elle est élevée, bleu là où elle est faible.

La figure 8.23(a) représente des plans de coupe de l'espace des paramètres, où deux minima locaux peuvent être identifiés. Dans la figure 8.23(b), ce sont des iso-valeurs de la traînée qui sont extraites le long de la surface. Une région est identifiable comme étant une zone de minimum de la traînée au bord du domaine.



(a) Plans de coupe de la surface de réponse



(b) Iso-valeurs de la traînée F_x le long de la surface de réponse

FIGURE 8.23 – Vues de la surface de réponse

Conclusion

Cette étude a permis d'explorer un espace de variations de forme du bulbe à partir d'un design réalisé par un architecte. Nous avons construit un plan d'expérience, qui constitue la première étape d'un algorithme d'optimisation avec méta-modèle. L'ensemble des géométries

testées ont été générées automatiquement par le modeleur paramétrique.

Tous les 20 points d'échantillonnage distribués sur le domaine de variation des 3 paramètres de forme, longueur, angle et largeur, ont amélioré les performances du chalutier. Le meilleur point obtenu a diminué de $11,08\%$ la traînée de la carène originale sans bulbe, soit une amélioration de $3,64\%$ par rapport à la carène avec le bulbe initial.

Réaliser l'algorithme d'optimisation jusqu'au bout permettra certainement d'améliorer les résultats obtenus, et donc de trouver un bulbe optimal.

L'un des points bloquants de l'automatisation de FINE^{TM}/Marine est la génération du maillage. Pour surmonter ce verrou, nous prévoyons d'intégrer une méthode de déformation de maillage liée à la déformation de la surface de l'objet. Ainsi, le premier maillage serait réutilisé à chaque nouvelle forme.

Une autre amélioration importante sera l'intégration de nouveaux paramètres de forme pour le bulbe. En effet, en plus des trois paramètres choisis, les performances du bulbe dépendent aussi de l'aire des sections, la position verticale du centre de gravité des sections, le type de bulbe (delta, ovale, nabla) ou le volume total.

Chapitre 9

Conclusion et perspectives

9.1 Conclusion

Le travail mené durant cette thèse a permis le développement d'un modeleur paramétrique générique et adapté à l'utilisation dans une boucle d'optimisation automatique de forme.

Le modèle géométrique proposé se base sur une double paramétrisation des formes : géométrique et métier.

L'approche par paramétrisation géométrique permet de décrire un grand nombre d'objets différents de façon précise et efficace. Les objets sont décrits par un squelette composé d'une famille de courbes B-Splines : la génératrice et les courbes de section.

La génératrice décrit les caractéristiques principales des objets, tandis que les sections décrivent précisément les contours.

L'approche métier permet de piloter les déformations de l'objet avec des paramètres architecturaux pertinents d'un point de vue design ou physique. Cela permet de réduire considérablement le nombre de degrés de liberté du problème d'optimisation de forme. C'est aussi grâce à cette approche que nous pouvons proposer des déformations cohérentes et valides d'un point de vue architectural.

La méthode de déformation mise en place est basée sur la résolution d'un problème inverse. Il s'agit de trouver la géométrie correspondante à un jeu de paramètres métier cibles. Pour déterminer cette géométrie, un système de minimisation est résolu avec comme inconnues la position des points de contrôle des courbes composant le squelette.

Des termes de lissage et de contrôle de forme sont introduits dans le système de minimisation, permettant de s'assurer de la validité des formes générées.

Nous proposons plusieurs techniques de reconstruction de morceaux de surfaces paramétriques, pour passer de la représentation par squelette à une représentation classique des surfaces en 3D. Nous portons une attention particulière à générer des surfaces bien raccordées, continues et lisses.

Les géométries générées sont compatibles avec les logiciels de simulation numériques utilisés et facilement pilotables par une toolbox d'optimisation.

Enfin, nous avons réalisé des applications pratiques du modeleur, qui ont prouvé ses capacités à générer des formes valides et pertinentes.

Pour l'application du profil d'aile d'avion, une boucle d'optimisation automatique de forme a été implémentée. Le modeleur, le solveur et l'algorithme d'optimisation ont été entièrement liés et automatisés, nous permettant de réaliser des milliers de calculs dans des temps réduits et presque sans intervention humaine. L'étude a permis de déterminer une forme de profil optimisée, qui maximise la finesse tout en assurant un coefficient de portance supérieur à un seuil fixé.

Pour l'application au foil AC45, une boucle d'optimisation automatique de forme a aussi été implémentée. A la fin de l'étude, nous avons mis en évidence des tendances nettes de l'influence des paramètres de forme sur les performances du foil. Cependant, nous avons aussi constaté l'importance d'inclure des contraintes multi-physiques lors du traitement de problèmes aussi complexes. Ajouter un solveur structure dans la boucle pourrait éliminer les designs non constructibles, mais valides d'un point de vue géométrique et hydrodynamique.

Pour l'application au bulbe de chalutier, nous avons réalisé un plan d'expérience à partir d'un bulbe initialement dessiné par un architecte. Automatiser entièrement le logiciel FINE^{TM}/Marine est difficile, nous avons donc généré automatiquement un ensemble de géométries avec le modeleur, puis après avoir réaliser le maillage manuellement nous avons lancé les calculs automatiquement. L'exploration de l'espace des paramètres a déjà permis de déterminer des bulbes meilleurs que le bulbe initial (réduction de $11,08\%$ de la traînée par rapport à la coque sans bulbe et de $3,64\%$ par rapport à la coque avec le bulbe initial).

9.2 Futur travail envisagé

De nombreux verrous scientifiques ont été abordés et traités durant cette thèse, mais d'autres problèmes sont encore ouverts. De nouveaux travaux de recherche peuvent être menés pour améliorer et rendre encore plus générique le modeleur paramétrique.

Comme discuté en introduction, le modeleur pourra être étendu à d'autres applications : hélices, pales d'éolienne ou d'hydrolienne, fuselage ou ailes d'un avion, sous-marins, ballons dirigeables, câbles et *pipes* utilisés en offshore, etc.

Beaucoup de ces applications ne nécessitent pas d'adaptation du modeleur actuel, une simple extension des paramètres actuels suffira à traiter ces nouveaux cas. Cependant, pour traiter avec pertinence le cas de l'hélice par exemple, les sections du squelette devraient être obtenues en suivant des coordonnées cylindriques, comme c'est actuellement fait par les designers de pales. Étendre le modeleur a des objets décrits par des coordonnées cylindriques ou sphériques permettra de l'appliquer à davantage de formes.

De façon générale, pour toutes les formes traitables par le modeleur, une extension des paramètres de formes actuellement implémentés est nécessaire : intégrer des paramètres liés à l'aire ou au volume des objets, le centre de gravité, etc.

Cette extension ne présente pas d'écueils particuliers, mais pour la prise en compte du volume total de l'objet, les sections ne pourront plus être considérées comme indépendantes et les sous problèmes d'optimisation devront être résolus simultanément.

Le squelette actuel est composé de courbes "simples" : elles ne comportent qu'une seule partie et n'ont pas d'embranchements. Inclure la possibilité de courbes par morceaux ou avec des embranchements permettrait de décrire avec précision des géométries complexes.

De plus, le squelette pourrait être amélioré en tenant compte d'un deuxième jeu de sections. Les sections actuelles peuvent être vues comme les coupes transversales d'un plan de forme de navire, et la génératrice représente une des coupes longitudinales. Les coques sont aussi découpées avec des coupes horizontales, ou lignes d'eau, pour représenter la troisième direction. Notre modèle pourrait lui aussi intégrer un deuxième jeu de sections, qui correspondrait aux lignes d'eau.

Concernant le lien avec les solveurs numériques nécessitant un maillage volumique, comme *ISIS-CFD*, nous envisageons de développer ou d'intégrer un module de déformation de maillage. Cette approche permettrait de rendre le processus d'optimisation de forme automatique, puisque seul le premier maillage aura besoin d'être généré manuellement. De plus, si le maillage est seulement déformé, et non régénéré, le calcul de l'écoulement peut être redémarré à partir de l'écoulement précédent convergé. Cette technique permet de réduire considérablement le temps de convergence du nouveau calcul.

Une méthodologie envisageable serait d'identifier les sommets du maillage sur la surface déformable, puis de projeter ces points sur la surface déformée (reconstruite à partir du squelette). La projection se ferait de façon à conserver au mieux les propriétés d'orthogonalité du maillage, afin de maintenir une qualité des mailles la plus haute possible.

D'un point de vue pratique, le modeleur est actuellement implémenté en grande partie avec MATLAB. Certains algorithmes (l'approximation de courbes notamment) ont déjà été transférés en C++, qui est plus rapide et plus léger que MATLAB à l'exécution. Pour une utilisation industrielle, un passage complet à C++ est nécessaire. Nous avons déjà identifié les librairies de gestion des NURBS disponibles : openNURBS de Rhinocéros 3D^{TM} et GoTools du laboratoire SINTEF. De plus, le code actuel en MATLAB a été développé de façon générique, en se basant sur les concepts d'interfaces et d'héritage propres aux langages orientés objets.

Finalement, le développement d'une interface graphique propre au modeleur faciliterait son utilisation.

Le modeleur paramétrique développé durant cette thèse a vocation a être utilisé de façon industrielle par l'entreprise MyCFD. MyCFD développe des services d'automatisation de calculs numériques. Les architectes font appels à ce type de services lorsqu'ils sont dans la phase de conception d'un nouveau design. MyCFD pourra proposer une offre d'optimisation automatique de forme couplée à ses services d'automatisation, pour que les architectes puissent explorer l'espace des variations possibles de leur design, ou identifier automatiquement le design optimal grâce à une boucle d'optimisation de forme.

Bibliographie

[Abraham et al., 2005] Abraham, A., Koppen, M., Grosan, C., and Oltean, M. (2005). Multiobjective optimization using adaptive pareto archived evolution strategy. *Intelligent Systems Design and Applications, International Conference on*, pages 558–563.

[Aftosmis et al., 1999] Aftosmis, M., Delanaye, M., and Haimes, R. (1999). Automatic generation of cfd-ready surface triangulations from cad geometry. *37th Aerospace Sciences Meeting and Exhibit. Reno,NV,U.S.A.*.

[Aguilar, 1996] Aguilar, J.-C. (1996). *Optimisation de formes hydrodynamiques. Couche limite intrinseque tridimensionnelle*. PhD thesis, Ecole des Mines de Paris.

[Alexa, 2002] Alexa, M. (2002). Recent advances in mesh morphing. *Computer Graphics Forum*, 21(2) :173–198.

[Alexa, 2003] Alexa, M. (2003). Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, 19(2) :105–114.

[Allaire et al., 2002] Allaire, G., Jouve, F., and Toader, A.-M. (2002). A level-set method for shape optimization. *C. R. Acad. Sci. Paris*, 334(12) :1125–1130.

[Ang et al., 2015] Ang, J., Goh, C., and Li, Y. (2015). Hull form design optimisation for improved efficiency and hydrodynamic performance of 'ship-shaped' offshore vessels. In *ICCAS 2015 - International Conference on Computer Applications in Shipbuilding*.

[Baiwei et al., 2011] Baiwei, F., Hu, C., Liu, Z., Zhan, C., and Chang, H. (2011). Ship resistance performance optimization design based on CAD/CFD. In *3rd International Conference on Advanced Computer Control, Harbin*.

[Blake and Isard, 1998] Blake, A. and Isard, M. (1998). *Active contours - the application of techniques from graphics, vision, control theory and statistics to visual tracking of shapes in motion*. Springer.

[Blanchard et al., 2013] Blanchard, L., Berrini, E., Duvigneau, R., Roux, Y., Mourrain, B., and Jean, E. (2013). Bulbous bow shape optimization. In *5th International Conference on Computation Mehods in Marine Engineering (Marine 2013)*, pages 412–423.

[Bo et al., 2012] Bo, P., Ling, R., and Wang, W. (2012). A revisit to fitting parametric surfaces to point clouds. *Computers & Graphics*, 36(5) :534–540.

[Brizzolara et al., 2015] Brizzolara, S., Vernengo, G., Pasquinucci, C. A., and Harries, S. (2015). Significance of parametric hull form definition on hydrodynamic performance optimization. In *6th International Conference on Computation Mehods in Marine Engineering (Marine 2015)*, pages 254–265.

[Capell et al., 2002] Capell, S., Green, S., Curless, B., Duchamp, T., and Popović, Z. (2002). Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph.*, 21(3) :586–593.

[Catmull and Clark, 1978] Catmull, E. and Clark, J. (1978). Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6) :350–355.

[Ceruti et al., 2014] Ceruti, A., Liverani, A., and Caligiana, G. (2014). Fairing with neighbourhood lod filtering to upgrade interactively b-spline into class-a curve. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 8(2) :67–75.

[Charvet, 1991] Charvet, T. (1991). Simulation numérique de l'écoulement autour de voiles de bateau. In *3ème Journées de l'Hydrodynamique*.

[Charvet et al., 1996] Charvet, T., Hauville, F., and Huberson, S. (1996). Numerical simulation of the flow around sails in real sailing conditions. *Journal of Wind Engineering and Industrial Aerodynamics*, 63(1-3) :111–129.

[Coleman and Verma, 2000] Coleman, T. F. and Verma, A. (2000). Admit-1 : Automatic differentiation and matlab interface toolbox. *ACM Trans. Math. Softw.*, 26(1) :150–175.

[Collette and Siarry, 2002] Collette, Y. and Siarry, P. (2002). *Optimisation multiobjectif*. Algorithmes. Eyrolles, Paris.

[Couser et al., 2011] Couser, P., Harries, S., and Tillig, F. (2011). Numerical hull series for calm water and sea-keeping. In *COMPIT'11, 10th International Conference on Computer and IT Applications in the Maritime Industries*, pages 202–212.

[Damon, 2005] Damon, J. (2005). Determining the geometry of boundaries of objects from medial data. *International Journal of Computer Vision*, 63(1) :45–64.

[Dawson, 1977] Dawson, C. (1977). Method for solving ship wave problems. In *Proceedings of the 2nd International Conference on Numerical Ship Hydrodynamics*, pages 305–318.

[De Boor, 2001] De Boor, C. (2001). *A practical guide to splines ; rev. ed.* Applied mathematical sciences. Springer, Berlin.

[De Gournay, 2005] De Gournay, F. (2005). *Optimisation de formes par la méthode des lignes de niveaux*. PhD thesis, ECOLE POLYTECHNIQUE.

[De Nayer, 2008] De Nayer, G. (2008). *Intéractions Fluides-Structures pour les corps élancés*. PhD thesis, Ecole Centrale de Nantes, Laboratoire de Mécanique des Fluides.

[Deb and Kalyanmoy, 2001] Deb, K. and Kalyanmoy, D. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA.

[Deng et al., 2015] Deng, B., Bouaziz, S., Deuss, M., Kaspar, A., Schwartzburg, Y., and Pauly, M. (2015). Interactive design exploration for constrained meshes. *Computer-Aided Design*, 61 :13 – 23. Steering Architectural Form.

[DeRose et al., 1998] DeRose, T., Kass, M., and Truong, T. (1998). Subdivision surfaces in character animation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 85–94, New York, NY, USA. ACM.

[Desbrun et al., 2002] Desbrun, M., Meyer, M., and Alliez, P. (2002). Intrinsic Parameterizations of Surface Meshes. *Computer Graphics Forum*.

[Dokken and Skytt, 2006] Dokken, T. and Skytt, V. (2006). Intersection algorithms and cagd. *Geometrical Modeling, Numerical Simulation, and Optimization : Industrial Mathematics at SINTEF*, pages 41–90.

[Drela, 1989] Drela, M. (1989). *XFOIL : An Analysis and Design System for Low Reynolds Number Airfoils*, pages 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Durand, 2012] Durand, M. (2012). *Interaction fluide-structure souple et légère, applications aux voiliers*. PhD thesis, Ecole Centrale de Nantes.

[Durand et al., 2014] Durand, M., Leroyer, A., Lothodé, C., Hauville, F., Visonneau, M., Floch, R., and Guillaume, L. (2014). Fsi investigation on stability of downwind sails with an automatic dynamic trimming. *Ocean Engineering*, 90 :129–139. Innovation in High Performance Sailing Yachts - INNOVSAIL.

[Duvigneau and Chandrashekar, 2012] Duvigneau, R. and Chandrashekar, P. (2012). Kriging-based optimization applied to flow control. *International Journal for Numerical Methods in Fluids*, 69(11) :1701–1714.

[Duvigneau and Visonneau, 2003] Duvigneau, R. and Visonneau, M. (2003). Shape optimization strategies for complex applications in computational fluid dynamics. In *2nd International Conference on Computer Applications and Information Technology in the Maritime Industries*, pages 999–1006.

[Duvigneau et al., 2003] Duvigneau, R., Visonneau, M., and Deng, G. (2003). On the role played by turbulence closures in hull shape imization at model and full scale. *J. Marine Science and Technology*, 8(1) :11–25.

[Eck and Hadenfeld, 1995] Eck, M. and Hadenfeld, J. (1995). *Local Energy Fairing of B-spline Curves*, pages 129–147. Springer Vienna, Vienna.

[El Majd et al., 2008] El Majd, B., Désidéri, J., and Duvigneau, R. (2008). Multilevel strategies for parametric shape optimization in aerodynamics. *Revue européenne de mécanique numérique*, 17(1-2).

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

[Faltinsen, 2006] Faltinsen, O. M. (2006). *Hydrodynamics of High-Speed Marine Vehicles*. Cambridge University Press.

[Fletcher, 1987] Fletcher, R. (1987). *Practical Methods of Optimization ; (2Nd Ed.)*. Wiley-Interscience, New York, NY, USA.

[Floater, 1997] Floater, M. S. (1997). Parametrization and smooth approximation of surface triangulations. *Comput. Aided Geom. Des.*, 14(3) :231–250.

[Floater, 2003] Floater, M. S. (2003). Mean value coordinates. *Comput. Aided Geom. Des.*, 20(1) :19–27.

[Froude and Froude, 1888] Froude, W. and Froude, R. (1888). *The Resistance of Ships*. Naval professional papers. U.S. Government Printing Office.

[Geremia et al., 2011] Geremia, P., Schumacher, T., and De Villiers, E. (2011). Fast robust design optimisation methods applied to ship hydrodynamics. In *4th International Conference on Computation Mehods in Marine Engineering (Marine 2011)*, pages 619–725.

[Gibson and Mirtich, 1997] Gibson, S. F. F. and Mirtich, B. (1997). A survey of deformable modeling in computer graphics. Technical report, MERL.

[Gill et al., 1981] Gill, P. E., Murray, W., and Wright, M. H. (1981). *Practical optimization*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

[Gonzalez-Hidalgo et al., 2007] Gonzalez-Hidalgo, M., Mir, A., and Nicolau, G. (2007). An evolution model of parametric surface deformation using finite elements based on b-splines. In *Computational Modelling of Objects Represented in Images*, pages 205–210. Taylor and Francis Group.

[Gravesen and Ungstrup, 2002] Gravesen, J. and Ungstrup, M. (2002). Constructing invariant fairness measures for surfaces. *Advances in Computational Mathematics*, 17(1) :67–88.

[Guha and Falzarano, 2015] Guha, A. and Falzarano, J. (2015). Application of multi objective genetic algorithm in ship hull optimization. *Ocean Engineering*, 5(2) :91 – 107.

[Guido, 1997] Guido, Y. (1997). *Controle et optimisation de forme dans les equations de Navier-Stokes*. PhD thesis, Ecole des Mines de Paris.

[Haron et al., 2012] Haron, H., Rehman, A., Adi, D., Lim, S., and Saba, T. (2012). Parameterization method on b-spline curve. *Mathematical Problems in Engineeringn*, vol. 2012.

[Hock et al., 2016] Hock, J., Goh, C., and Li, Y. (2016). Hybrid evolutionary shape manipulation for efficient hull form design optimisation. In *COMPIT'15, 15th International Conference on Computer and IT Applications in the Maritime Industries*, pages 264–279.

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

[Hormann, 2000] Hormann, K. (2000). Mips : An efficient global parametrization method. In *ACM Press/Addison-Wesley Publishing Co.*

[Hoschek, 1988] Hoschek, J. (1988). Intrinsic parametrization for approximation. *Comput. Aided Geom. Des.*, 5(1) :27–31.

[Huberson, 1986] Huberson, S. (1986). *Modelisation asymptotique et simulation numerique d'ecoulements tourbillonaires*. PhD thesis, Université Pierre et Marie Curie (ParisVI)- LIMSI-CNRS.

[Huetz and Guillerm, 2014] Huetz, L. and Guillerm, P. E. (2014). Database building and statistical methods to predict sailing yacht hydrodynamics. *Ocean Engineering*, 90 :21 – 33. Innovation in High Performance Sailing Yachts - {INNOVSAIL}.

[Iman et al., 1981] Iman, R. L., Helton, J. C., and Campbell, J. E. Y. (1981). An approach to sensitivity analysis of computer models, part 1. introduction, input variable selection an preliminary variable assessment. *Journal on Quality Technology*, 13(3) :174–183.

[Jacquin et al., 2004] Jacquin, E., Derbanne, Q., Cordier, S., and Alessandrini, B. (2004). Hull form optimization using a free surface ranse solver. In *25th Symposium on Naval Hydrodynamics*, pages 1–14.

[Jeong et al., 2005] Jeong, S., Murayama, M., and Yamamoto, K. (2005). Efficient optimization design method using kriging model. *Journal of Aircraft*, 42(2) :413–420.

[Johan et al., 2000] Johan, H., Koiso, Y., and Nishita, T. (2000). Morphing using curves and shape interpolation techniques. In *Proceedings the Eighth Pacific Conference on Computer Graphics and Applications*, pages 348–454.

[Jones, 2001] Jones, D. R. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21 :345–383.

[Jones et al., 1998] Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4) :455–492.

[Ju and Goldman, 2003] Ju, T. and Goldman, R. (2003). Morphing rational b-spline curves and surfaces using mass distributions. In *Eurographics 2003 - Short Presentations*. Eurographics Association.

[Ju et al., 2005] Ju, T., Schaefer, S., and Warren, J. D. (2005). Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.*, 24(3) :561–566.

[Kang and Lee, 2012] Kang, J. and Lee, B. (2012). Geometric interpolation and extrapolation for rapid generation of hull forms. In *COMPIT'12, 11th International Conference on Computer and IT Applications in the Maritime Industries*, pages 202–212.

[Katz and Plotkin, 2001] Katz, J. and Plotkin, A. (2001). *Low-Speed Aerodynamics*. Cambridge Aerospace Series. Cambridge University Press.

[Kaul and Rossignac, 1991] Kaul, A. and Rossignac, J. (1991). Solid-interpolating deformations : Construction and animation of pips. In *EG 1991-Technical Papers*. Eurographics Association.

[Kent et al., 1992] Kent, J. R., Carlson, W. E., and Parent, R. E. (1992). Shape transformation for polyhedral objects. *SIGGRAPH Comput. Graph.*, 26(2) :47–54.

[Knowles and Corne, 1999] Knowles, J. and Corne, D. (1999). The Pareto archived evolution strategy : a new baseline algorithm for Pareto multiobjective optimisation. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1.

[Kostas et al., 2015] Kostas, K., Ginnis, A., Politis, C., and Kaklis, P. (2015). Ship-hull shape optimization with a t-spline based bem-isogeometric solver. *Computer Methods in Applied Mechanics and Engineering*, 284 :611 – 622. Isogeometric Analysis Special Issue.

[Kostas et al., 2016] Kostas, K. V., Ginnis, A. G., Politis, C. G., and Kaklis, P. (2016). Shape-optimization of 2d hydrofoils using an isogeometric bem solver. *Computer-Aided Design*.

[Kracht, 1978] Kracht, A. (1978). Design of bulbous bows. *SNAME Transactions*, 86 :197–217.

[Lackenby, 1950] Lackenby, H. (1950). On the systematic geometrical variation of ship forms. *Trans. INA*, 92 :289–315.

[Lee et al., 2000] Lee, A., Moreton, H., and Hoppe, H. (2000). Displaced subdivision surfaces. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 85–94, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Lee, 1989] Lee, E. T. Y. (1989). Choosing nodes in parametric curve interpolation. *Comput. Aided Des.*, 21(6) :363–370.

[Leroyer, 2004] Leroyer, A. (2004). *Etude du couplage ecoulement/mouvement pour des corps solides ou à déformation imposée par résolution des équations de Navier-Stokes. Contribution à un modélisation numérique de la cavitation*. PhD thesis, Ecole Centrale de Nantes et Université de Nantes, Laboratoire de Mécanique des Fluides.

[Loop and Schaefer, 2008] Loop, C. and Schaefer, S. (2008). Approximating catmull-clark subdivision surfaces with bicubic patches. *ACM Trans. Graph.*, 27(1) :8 :1–8 :11.

[Lothodé et al., 2013] Lothodé, C., Durand, M., Roux, Y., Leroyer, A., Visonneau, M., and Dorez, L. (2013). Dynamic fluid structure interaction of a foil. In *Innov'Sail 2013, 26th - 28th June*, pages 1–6.

[Lü, 2009] Lü, W. (2009). Curves with chord length parameterization. *Comput. Aided Geom. Des.*, 26(3) :342–350.

[MacPherson et al., 2016] MacPherson, D., Harries, S., Broenstrup, S., and Dudka, J. (2016). Numerical hull series for calm water and sea-keeping. In *COMPIT'16, 15th International Conference on Computer and IT Applications in the Maritime Industries*, pages 204–217.

[Maisonneuve et al., 2003] Maisonneuve, J., Harries, S., Marzi, J., Raven, H., Viviani, U., and Piippo, H. (2003). Towards optimal design of ship hull shapes. *INTERNATIONAL MARINE DESIGN*, 2 :31–42.

[Mason and Thomas, 2007] Mason, A. and Thomas, G. (2007). Stochastic optimisation of iacc yachts. In *COMPIT'07, 6th International Conference on Computer and IT Applications in the Maritime Industries*, pages 122–141.

[MATLAB, 2015] MATLAB (2015). *version 8.6.0 (R2015b)*. The MathWorks Inc., Natick, Massachusetts.

[McKay et al., 1979] McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2) :239–245.

[Michalewicz, 1996] Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag Berlin Heidelberg.

[Michell, 1898] Michell, J. (1898). The wave resistance of a ship. *Phil. Mag.*, 5(45) :106–123.

[Mukesh et al., 2014] Mukesh, R., Lingadurai, K., and Selvakumar, U. (2014). Airfoil shape optimization using non traditional optimization technique and its validation. *Journal of King Saud University, Engineering Sciences*, 26(2) :191–197.

[Mun et al., 2003] Mun, D., Han, S., Kim, J., and Oh, Y. (2003). A set of standard modeling commands for the history-based parametric approach. *Computer-Aided Design*, 35(13) :1171 – 1179.

[Nealen et al., 2006] Nealen, A., Müller, M., Keiser, R., Boxerman, E., and Carlson, M. (2006). Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4) :809–836.

[Nelder and Mead, 1965] Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, 7 :308–313.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, Berlin.

[Papanikolaou et al., 2011] Papanikolaou, A., Harries, S., Wilken, M., and Zaraphonitis, G. (2011). Integrated ship design and multiobjective optimization approach to ship design. In *International Conference on Computer Applications in Shipbuilding*, pages 1–12.

[Patrikalakis and Maekawa, 2002] Patrikalakis, N. M. and Maekawa, T. (2002). *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer.

[Paulet and Presles, 1999] Paulet, D. and Presles, D. (1999). *Architecture navale, connaissance et pratique*. Editions de la Villette, Paris, France.

[Peri et al., 2009] Peri, D., Campana, E. F., Kandasamy, M., Ooi, S. K., Carrica, P., Stern, F., Osborne, P., Macdonald, N., and de Waal, N. (2009). Potential flow based optimization of a high speed, foil-assisted, semi-planning catamaran for low wake. In *10th International Conference on Fast Sea Transportation FAST 2009, Athens, Greece*.

[Peri and Diez, 2013] Peri, D. and Diez, M. (2013). Robust design optimization of a monohull for wave wash minimization. In *5th International Conference on Computation Mehods in Marine Engineering (Marine 2013)*, pages 89–100.

[Peri et al., 2001] Peri, D., Rossetti, M., and Campana, E. (2001). Design optimization of ship hulls via cfd techniques. *Journal of Ship Research*, 10 :140–149.

[Peters and Reif, 2008] Peters, J. and Reif, U. (2008). *Subdivision Surfaces*. Geometry and Computing. Springer Berlin Heidelberg.

[Piegl and Tiller, 1997] Piegl, L. and Tiller, W. (1997). *The NURBS Book (2Nd Ed.)*. Springer-Verlag New York, Inc., New York, NY, USA.

[Queutey and Visonneau, 2007] Queutey, P. and Visonneau, M. (2007). An interface capturing method for free-surface hydrodynamic flows. *Computers & Fluids*, 36(9) :1481–1510.

[Rasmussen and Williams, 2005] Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

[Rehbach, 1977] Rehbach, C. (1977). Numerical calculation of three dimensional unsteady flows with vortex sheets. *La Recherche Aérospatiale*, 1 :289–298.

[Roux, 1999] Roux, Y. (1999). *Étude de l'amortissement visqueux dans les problèmes de tenue à la mer d'un flotteur immergé*. PhD thesis. Thèse de doctorat dirigée par Huberson, Serge Mécanique des fluides Le Havre 1999.

[Rozza et al., 2013] Rozza, G., Koshakji, A., and Quarteroni, A. (2013). Free form deformation techniques applied to 3d shape optimization problems. *Communications in Applied and Industrial Mathematics*, 4.

[Samareh, 2005] Samareh, J. A. (2005). Geometry and grid/mesh generation issues for cfd and csm shape optimization. *Optimization and Engineering*, 6(1) :21–32.

[Schafer et al., 2014] Schafer, H., Keinert, B., Niessner, M., Buchenau, C., Guthe, M., and Stamminger, M. (2014). Real-time deformation of subdivision surfaces from object collisions. In *Proceedings of the 6th High-Performance Graphics Conference*, pages 1–8. EG.

[Sederberg and Parry, 1986] Sederberg, T. W. and Parry, S. R. (1986). Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.*, 20(4) :151–160.

[Skouras et al., 2012] Skouras, M., Thomaszewski, B., Bickel, B., and Groos, M. (2012). Computational design of rubber balloons. *Comput. Graph. Forum*, 31(2pt4) :835–844.

[Smith et al., 1956] Smith, A., Company, D. A., and Gamberoni, N. (1956). *Transition, Pressure Gradient and Stability Theory*. Douglas Aircraft Company, El Segundo Division.

[Sobieczky, 1999] Sobieczky, H. (1999). *Parametric Airfoils and Wings*, pages 71–87. Vieweg+Teubner Verlag, Wiesbaden.

[Sorkine et al., 2004] Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., and Seidel, H.-P. (2004). Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 175–184, New York, NY, USA. ACM.

[Tahara et al., 2008] Tahara, Y., Peri, D., Campana, E. F., and Stern, F. (2008). Computational fluid dynamics-based multiobjective optimization of a surface combatant using a global optimization method. *Journal of Marine Science and Technology*, 13(2) :95–116.

[Tang et al., 2014] Tang, C., Sun, X., Gomes, A., Wallner, J., and Pottmann, H. (2014). Form-finding with polyhedral meshes made simple. *ACM Trans. Graph.*, 33(4) :70 :1–70 :9.

[Thomassen et al., 2005] Thomassen, J. B., Johansen, P., and Dokken, T. (2005). Closest points, moving surfaces and algebraic geometry. *Mathematical Methods for Curves and Surfaces : Tromso 2004*, pages 351–362.

[Torczon, 1989] Torczon, V. J. (1989). *Multidirectional search : A direct search algorithm for parallel machines*. PhD thesis, Rice University.

[Tutte, 1963] Tutte, W. T. (1963). How to Draw a Graph. *Proceedings of the London Mathematical Society*, s3-13(1) :743–767.

[Van Ingen and Technische Hogeschool Delft, 1956] Van Ingen, J. and Technische Hogeschool Delft, V. (1956). *A SUGGESTED SEMI-EMPIRICAL METHOD FOR THE CALCULATION OF THE BOUNDARY LAYER TRANSITION REGION.* TH Delft, Delft.

[van Oossanen, 1985] van Oossanen, P. (1985). The development of the 12 meter class yacht Australia II. In *Proceedings of the 7th CSYS*. SNAME.

[Vanderplaats, 1987] Vanderplaats, G. N. (1987). *Numerical Optimization Techniques*, pages 197–239. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Wackers et al., 2010] Wackers, J., Ait Said, K., Deng, G., Mizine, I., Queutey, P., and Visonneau, M. (2010). Adaptive grid refinement applied to RANSE ship flow computation. *28th ONR Symposium on Naval Hydrodynamics*.

[Wackers et al., 2012] Wackers, J., Deng, G., Leroyer, A., Queutey, P., and Visonneau, M. (2012). Adaptive grid refinement algorithm for hydrodynamic flows. *Computers & Fluids*, 55 :85–100.

[Wang et al., 2006] Wang, W., Pottmann, H., and Liu, Y. (2006). Fitting b-spline curves to point clouds by curvature-based squared distance minimization. *ACM Trans. Graph.*, 25(2) :214–238.

[Warren, 1996] Warren, J. (1996). Barycentric coordinates for convex polytopes. *Advances in Computational Mathematics*, 6(1) :97–108.

[Warren and Weimer, 2001] Warren, J. and Weimer, H. (2001). *Subdivision Methods for Geometric Design : A Constructive Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

[Yoshizawa et al., 2007] Yoshizawa, S., Belyaev, A. G., and Seidel, H. P. (2007). Skeleton-based variational mesh deformations. *Comput. Graph. Forum*, 26(3) :255–264.

[Zerbinati et al., 2011] Zerbinati, A., Desideri, J.-A., and Duvigneau, R. (2011). Comparison between MGDA and PAES for Multi-Objective Optimization. Research Report RR-7667, INRIA.

[Zhang et al., 2001] Zhang, C., Zhang, P., and Cheng, F. F. (2001). Fairing spline curves and surfaces by minimizing energy. *Computer-Aided Design*, 33(13) :913 – 923.

[Zhou et al., 2007] Zhou, K., Huang, X., Xu, W., Guo, B., and Shum, H.-Y. (2007). Direct manipulation of subdivision surfaces on GPUs. *ACM Trans. Graph.*, 26(3) :Article 91.

[Zockler et al., 2000] Zockler, M., Stalling, D., and Hege, H.-C. (2000). Fast and intuitive generation of geometric shape transitions. *The Visual Computer*, 16(5) :241–253.