Optimisation de formes de coques minces pour des géométries complexes.

Sarah Julisson

NNT : 2016SACLV106

THÈSE DE DOCTORAT

de

L'Université Paris-Saclay

École doctorale de mathématiques Hadamard (EDMH, ED 574)

Établissement d'inscription : Université de Versailles Saint-Quentin-en-Yvelines

Établissements d'accueil : Technocentre Renault,
Institut de Recherche Technologique SystemX

Laboratoire d'accueil : Laboratoire de mathématiques de Versailles, UMR 8100 CNRS

Spécialité de doctorat : Mathématiques appliquées

Sarah JULISSON

Optimisation de formes de coques minces pour des géométries complexes

Date de soutenance : 2 décembre 2016

Après avis des rapporteurs : Adel BLOUZA (Université de Rouen)
Bijan MOHAMMADI (Université de Montpellier)

Jury de soutenance :

Adel BLOUZA	(Université de Rouen)	Rapporteur
Christophe CHALONS	(Université Paris-Saclay)	Examinateur
Philippe DESTUYNDER	(Conservatoire National des Arts et Métiers)	Président du jury
Laurent DUMAS	(Université Paris-Saclay)	Directeur de thèse
Christian FOURCADE	(Renault S.A.S)	Examinateur
Frédéric HECHT	(Université Pierre et Marie Curie)	Examinateur
Paul de NAZELLE	(Institut de Recherche Technologique SystemX)	Invité

université PARIS-SACLAY

École doctorale de mathématiques Hadamard (EDMH)

Titre : Optimisation de formes de coques minces pour des géométries complexes.

Mots clefs : optimisation de formes, coques minces, analyse isogéométrique, méthode adjointe

Résumé : Au cours des processus de conception, l'optimisation de formes apporte aux industriels des solutions pour l'amélioration des performances des produits. En particulier, les structures minces qui constituent environ 70% d'un véhicule, sont une préoccupation dans l'industrie automobile. La plupart des méthodes d'optimisation pour ces structures surfaciques présentent certaines limites et nécessitent des expertises à chaque niveau de la procédure d'optimisation.

L'objectif de cette thèse est de proposer une nouvelle stratégie d'optimisation de formes pour les coques minces. L'approche présentée consiste à exploiter les équations de coques du modèle de Koiter en se basant sur une analyse isogéométrique. Cette méthode permet de réaliser des simulations sur la géométrie exacte en définissant la forme à l'aide de patchs CAO. Les variables d'optimisation choisies sont alors les points de contrôle permettant de piloter leur forme. La définition des patchs permet également de dégager un gradient de forme pour l'optimisation à l'aide d'une méthode adjointe.

Cette méthode a été appliquée pour des critères mécaniques issus des bureaux d'études Renault. Des résultats d'optimisation pour un critère de compliance sont présentés. La définition et l'implémentation de critères vibro-acoustiques sont discutés à la fin de cette thèse. Les résultats obtenus témoignent de l'intérêt de la méthode. Toutefois, de nombreux développements seront nécessaires avant d'être en mesure de l'appliquer dans l'industrie.

Title : Shape optimization of thin shell structures for complex geometries.

Keywords : shape optimization, thin shell structures, isogeometric analysis, adjoint method

Abstract : During the design process, optimization of shapes offers manufacturers solutions for improving products performances. In particular, thin shell structures that represent about 70 % of a vehicle, are a concern in the automotive industry. Most optimization methods for surface structures have limitations and require expertise at every level of the optimization procedure.

The aim of this thesis is to propose a new strategy for the shape optimization of thin shell structures. The approach presented rely on using the Koiter's shell model based on an isogeometric analysis. This method allows for simulations on the exact geometry by defining the shape using CAD patches. Selected optimization variables are the control points used to control the shape of the CAD patches. Variations of these points allows to scan a wide design space with few parameters. The definition of patchs also enables to find a gradient with respect to the shape for the optimization by using the adjoint state method.

This method was applied to mechanical criteria from the Renault design offices. Optimization results for a compliance criterion are presented. The definition and implementation of vibro-acoustic criteria are discussed at the end of this thesis. The results demonstrate the interest of the method. However, many developments will be needed before being able to apply it in the industry.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

« L'important n'est pas de convaincre,
mais de donner à réfléchir. »

Bernard Werber

Remerciements

Au cours de mes trois années de thèse, une question que l'on m'a souvent posée était « mais tu n'entends vraiment rien avec ton casque sur les oreilles ? ». Il est vrai que la musique m'a accompagnée tout au long de ma thèse. Mais ce travail n'aurait pu aboutir sans des personnes exceptionnelles que je souhaite à présent remercier. Il est difficile pour moi de mettre des mots pour exprimer ma gratitude alors, encore une fois, je vais me tourner vers la musique car, comme l'a dit André Esparcieux, « *il est des sentiments si intraduisibles, qu'il faut la musique pour les suggérer* ».

Je tiens tout d'abord à remercier mon directeur de thèse Laurent Dumas qui m'a accordé sa confiance, soutenue pendant trois ans et apporté la réponse à la question « *what makes a good man ?* ». Merci à Christian de m'avoir rappelé de voir chaque difficulté comme « *just another brick and I am sledgehammer* ».

Merci à Frédéric, Marc, Sylvain, Mickaël et Pascal de m'avoir aussi bien accueillie au sein de l'équipe optimisation « *we are on each other team* ».

Ces trois années n'auraient pas été les mêmes sans les échanges avec Jean-Alexis ou Aurélie qui rendent « *happy* », les discussions du matin avec François qui peut librement dire « *don't threaten me with a good time* » et les conseils et encouragements de Pierre « *work it harder make it better, do it faster makes us stronger* ».

Mes sorties running ne pourront plus se faire sans fredonner « *we are the champions* » grâce au sourire, à la douceur et à la générosité de Virginie. Les aventures à l'autre bout du monde et les « battles » musicaux avec Timothée ont fait naître une amitié que l'on peut qualifier de « *total eclipse of the heart* ». Un grand merci à Paul pour sa bonne humeur, sa gentillesse et de m'avoir toujours dit « *I am free to be the greatest here tonight* ».

L'encadrement d'un ou une stagiaire peut être un moment délicat au cours d'une thèse sauf si cette personne est géniale comme Idil : « *everything about you is happiness.* ».

Ces travaux n'auraient pas leur point final sans l'approbation et remarques pertinentes de messieurs Blouza, Mohammadi, Chalons, Hecht et Destuynder qui ont accepté de juger ce travail et pour cela,

je les remercie sincèrement. « *It's like the finish line where everything just ends* ».

Du début à la fin, ma famille m'a apporté un soutien sans faille. Merci à ma mère et mes frères, et comme toujours « *I've bet my life on you* ».

Enfin, je voudrais remercier mon Laurent pour avoir toujours été là pour moi, « *you are my angel, come from way above to bring me love* ».

The Heavy - What Makes a Good Man ?
Rihanna - Sledgehammer
Lorde - Team
Pharrell Williams - Happy
Panic at the Disco ! - Don't Threaten me With a Good Time
Daft Punk - Harder, Better, Faster, Stronger
Queen - We are the Champion
Bonnie Tyler - Total Eclipse of the Heart
Sia - The Greatest
Muse - Bliss
Snow Patrol - The Finish Line
Imagine Dragons - I've Bet my Life
Massive Attack - Angel

Table des matières

Résumé **iii**

Remerciements **vii**

Introduction **1**

1 Principes généraux de l'optimisation de formes **3**
1.1 Présentation de l'optimisation de formes . . . 3
1.1.1 Rappels sur la définition d'un problème d'optimisation . . . 3
1.1.2 Définition de l'optimisation de formes . . . 4
1.1.3 Description de la forme . . . 6
1.1.4 Bilan sur l'optimisation de formes . . . 9
1.2 L'optimisation de formes surfaciques . . . 9
1.2.1 Contexte industriel . . . 9
1.2.2 Paramétrage des formes surfaciques . . . 10
1.2.3 Méthodes d'optimisation . . . 12
1.3 Critères de dimensionnement en automobile . . . 15
1.3.1 Les critères vibratoires . . . 15
1.3.2 Les critères d'endurance . . . 15
1.3.3 Les critères de déformation en grands déplacements . . . 16
1.4 Synthèse du chapitre . . . 17

2 Analyse isogéométrique et équations de coques **19**
2.1 Introduction . . . 19
2.2 Les patchs CAO . . . 20
2.2.1 Les patchs de Bézier . . . 20
2.2.2 Les surfaces B-splines . . . 21
2.2.3 Les patchs NURBS . . . 24
2.3 Le concept de l'analyse isogéométrique . . . 26
2.3.1 Définition de l'analyse géométrique . . . 26
2.3.2 Maillage des NURBS . . . 26

2.3.3 Méthodes numériques en analyse isogéoémtrique . . . 26
2.3.4 Bilan sur l'analyse isogéométrique . . . 28
2.4 Les équations de coques . . . 28
2.4.1 Description de la géométrie d'une coque mince . . . 28
2.4.2 Éléments de géométrie différentielle . . . 29
2.4.3 Le modèle linéaire de Koiter . . . 31
2.4.4 Mise en œuvre éléments finis . . . 32
2.4.5 Stratégie pour l'optimisation de formes . . . 33
2.5 Synthèse du chapitre . . . 33

3 Mise en place des simulations numériques de coques minces 35
3.1 Introduction . . . 36
3.2 Le raccordement entre patchs . . . 36
3.2.1 Le raccordement entre patchs CAO en isogéométrie . . . 36
3.2.2 Mise en œuvre pratique . . . 39
3.3 Méthode de résolution numérique . . . 45
3.3.1 Mise en œuvre de la méthode des éléments finis . . . 45
3.4 Présentation du code de calcul OptSurf . . . 49
3.4.1 Étape 1 : lecture du modèle CAO . . . 50
3.4.2 Étape 2 : maillage des domaines de référence . . . 50
3.4.3 Étape 3 : assemblage des matrices . . . 50
3.4.4 Étape 4 : résolution du problème de coques . . . 51
3.4.5 Étape 5 : le post-traitement . . . 55
3.4.6 Bilan sur le code OptSurf . . . 55
3.5 Résultats de quelques benchmarks . . . 55
3.5.1 Identification des modes de corps rigide . . . 56
3.5.2 Cas test classiques de déformation de coques . . . 58
3.5.3 Conclusion sur les résultats des benchmarks . . . 60
3.6 Extension de la méthode à des configurations industrielles . . . 61
3.7 Synthèse du chapitre . . . 63

4 Optimisation de formes de coques pour un critère de compliance 67
4.1 Introduction . . . 68
4.2 Définition du problème d'optimisation . . . 68
4.2.1 Le critère à minimiser . . . 68
4.2.2 Les variables d'optimisation . . . 69
4.2.3 Les contraintes . . . 69
4.2.4 Formulation du problème d'optimisation . . . 71
4.3 Calcul des gradients . . . 71
4.3.1 Différentielle de la fonction objectif et méthode de l'état adjoint . . . 71
4.3.2 Calcul du gradient de la contrainte de l'aire . . . 73
4.3.3 Implémentation des gradients . . . 73
4.3.4 Comparaison des gradients avec des différences finies . . . 74
4.4 L'algorithme d'optimisation . . . 77
4.4.1 Résolution du problème de barrière . . . 77
4.4.2 Recherche linéaire . . . 78
4.4.3 Les critères d'arrêt . . . 79
4.5 Résultats d'optimisation . . . 80
4.5.1 Un plateau soumis à son propre poids . . . 80
4.5.2 Un cube sous pression . . . 82
4.5.3 Un bac de roue de secours . . . 84
4.6 Synthèse du chapitre . . . 86

5 Optimisation de formes de coques pour un critère dynamique 87
5.1 Introduction . . . 88
5.2 Définition de la réponse forcée . . . 88
5.2.1 Formulation variationnelle de la réponse forcée . . . 88
5.2.2 Définition de l'amortissement . . . 90
5.3 Méthodes de résolution numérique . . . 91
5.3.1 La méthode directe . . . 91
5.3.2 La méthode de superposition modale . . . 91
5.3.3 La méthode des accélérations modales . . . 92
5.4 Implémentation de la réponse forcée harmonique . . . 93
5.4.1 Identification des coefficients pour l'amortissement . . . 93
5.4.2 Validation de l'implémentation . . . 93
5.5 Optimisation de formes en régime harmonique . . . 95
5.5.1 Définition du critère . . . 95
5.5.2 Calcul du gradient . . . 96
5.5.3 Validation du gradient . . . 99
5.5.4 Résultats d'optimisation . . . 101
5.6 Méthode de résolution numérique en régime transitoire . . . 102
5.6.1 Motivations . . . 102
5.6.2 Résolution numérique . . . 102
5.6.3 Résultats de benchmarks . . . 104
5.7 Synthèse du chapitre . . . 106

Conclusions et perspectives 107

A Classes et routines principales du code OptSurf 109
A.1 La classe Shape . . . 109
A.2 Calcul et stockage des matrices . . . 112
A.3 Les classes pour l'optimisation . . . 117
A.4 Exemple de fichier de commande pour OptSurf . . . 121

Bibliographie 123

Table des figures

1.1 Schéma de la citerne - Les variables x_1, x_2 et x_3 sont les variables d'optimisation qui pilotent la forme de la citerne. 4
1.2 Plaque soumise à une pression uniforme sur Γ_N et encastrée sur Γ_D 5
1.3 Illustration de l'optimisation de formes paramétrique : la forme est modifiée par variation des paramètres x_1, x_2 et x_3. 7
1.4 Illustration de l'optimisation de formes géométrique : la forme est modifiée par variation des frontières sans changement de la topologie. 7
1.5 Méthode de Hadamard - perturbation de la frontière Ω par un vecteur $\vec{\theta}(\vec{x})$. 8
1.6 Illustration de l'optimisation de formes topologique : la topologie de la forme peut être modifiée. 8
1.7 Vue éclatée de la Mégane où l'on peut distinguer les éléments surfaciques (source : [de Nazelle, 2013]). 9
1.8 Exemple de paramétrage CAO. 10
1.9 Déroulement de l'optimisation paramétrique. 10
1.10 Reconstruction du modèle CAO d'une sphère à partir d'un maillage. 11
1.11 Test acoustique du groupe motopropulseur - source : www.nordfranceinvest.fr 15
1.12 Essais de fatigue sur des pédales - source : Laboratoire Centre Technologie de l'Automobile de Galice. 16
1.13 Crash test - source : EuroNCAP 2015. 16

2.1 Déroulement d'une simulation 20
2.2 Exemple de deux courbes de Bézier de degré 3 pilotés par 4 points de contrôle 21
2.3 Surface de Bézier de degré 3 maîtrisée par 16 points de contrôle. 22
2.4 Exemple de deux courbes B-splines avec 7 points de contrôle - les • indiquent les points de rupture du vecteur de nœuds (c'est-à-dire lorsque $\xi_{i+1} \neq \xi_i$). 23
2.5 Modélisation d'un cercle parfait à l'aide une courbe NURBS pilotée par 9 points de contrôles et poids associés. 24
2.6 Répresentation d'une sphère exacte à l'aide d'une surface NURBS 25
2.7 Exemple de maillage d'une surface NURBS à partir de ses vecteurs de nœuds. 27
2.8 Géométrie d'une coque mince et fonction de forme 29
2.9 Deux exemples de paramétrage de la fonction de forme. 29

2.10 Dégrés de liberté d'un triangle d'Argyris. 32

3.1 Modèle CAO d'une doublure de capot Renault 36
3.2 Continuité $\mathcal{G}^1$ entre deux courbes B-splines 37
3.3 Continuité $\mathcal{G}^1$ entre deux surfaces NURBS 38
3.4 Représentation de la bande de flexion pour le raccordement entre patchs. 38
3.5 Jonction entre deux coques minces. 40
3.6 Exemple d'une charnière rigide. 42
3.7 Exemple d'une charnière élastique. 43
3.8 Discrétisation des domaines de référence : définition des fonctions affines par morceaux F et $\tilde{F}$. 48
3.9 Conditions de jonctions sur un triangle d'Argyris (valeurs aux nœuds, premières dérivées et dérivées secondes aux sommets du triangle). 48
3.10 Exemple de discrétisation de patchs non conformes. 49
3.11 Procédure de maillage des domaines de référence. 51
3.12 Exemple d'application des conditions aux limites. 51
3.13 Exemple d'un crosspoint commun à quatre patchs. 53
3.14 Organisation du code de calcul OptSurf. 56
3.15 Géométrie de la plaque avec un et quatre patchs. 57
3.16 Déformées modales de la plaque avec un patch pour les modes 7,8 et 9. 58
3.17 Déformées modales de la plaque avec quatre patchs pour les modes 7,8 et 9. 58
3.18 Géométrie des deux plaques perpendiculaires et 7ième déformée modale. 59
3.19 Géométrie du cylindre et 7ème déformée modale. 60
3.20 Géométrie de la plaque avec un et trois patchs. 61
3.21 Déplacements d'une plaque sous pression uniforme. 61
3.22 Déplacement du centre de la plaque sous pression en fonction de la discrétisation. . . . 62
3.23 Géométrie du paraboloïde hyperbolique. 63
3.24 Déplacements du paraboloïde hyperbolique. 63
3.25 Déplacement du centre du paraboloïde hyperbolique en fonction de la discrétisation. . 63
3.26 Description du problème de la toiture de Scordelis Lo. 64
3.27 Déplacements de la toiture de Scordelis Lo. 64
3.28 Déplacement du point O de la toiture de Scordelis Lo en fonction de la discrétisation. . 65
3.29 Modèle CAO d'un disque et sa surface sous-jacente. 65
3.30 Modèle CAO d'un capot et ses surfaces sous-jacentes - Source : modèle CAO de Jean-François Remacle. 65

4.1 Exemple de figures où les patchs s'intersectent. 70
4.2 Gestion des points de contrôle des jonctions entre patchs. 74
4.3 Points de contrôle de la géométrie du paraboloïde hyperbolique. 75
4.4 Évolution de la différence finie en fonction du paramètre h. 75
4.5 Comparaison entre les dérivées de la compliance calculées avec OptSurf et leurs approximations pas différences finies. 76
4.6 Comparaison entre les dérivées de l'aire calculées avec OptSurf et leurs approximations par différences finies. 77
4.7 Géométrie du plateau avec la position des points de contrôle. 80
4.8 Évolution de la compliance et de l'aire du plateau en fonction des itérations. 81
4.9 Comparaison entre la forme originale du plateau et l'optimum trouvé. 81
4.10 Comparaison des déformations du plateau (déplacements $\times 100$). 81
4.11 Géométrie du cube avec la position des points de contrôle. 82
4.12 Évolution de la compliance et de l'aire en fonction des itérations pour le cube sous pression. 83
4.13 Comparaison entre la forme originale et l'optimum trouvé. 83
4.14 Comparaison des déformations avant et après optimisation (déplacement $\times 100$). 83
4.15 Géométrie du bac de roue. 84

4.16 Évolution de la compliance et de l'aire pour le problème du bac de roue en fonction des itérations. 84
4.17 Forme du bac de roue à l'issue de l'optimisation. 85
4.18 Forme du fond du bac de roue. 85
4.19 Comparaison des déformations avant et après optimisation du bac de roue. 85

5.1 Géométrie et paramètres matériau de la plaque du cas test NAFEMS. 93
5.2 Réponse au centre de la plaque en fonction des fréquences d'excitation. 94
5.3 Géométrie de la plaque avec en rouge les points de contrôle 100
5.4 Comparaison entre le gradient de la vitesse normale par méthode adjointe et par différences finies 100
5.5 Évolution de la vitesse normale moyenne et de l'aire de la plaque en fonction des itérations. 101
5.6 Comparaison entre la forme originale de la plaque et la formé optimisée 101
5.7 Géométrie de la plaque du benchmark NAFEMS 13T 104
5.8 Évolution de u_z au centre de la plaque en fonction du temps 105
5.9 Configuration pour le problème de calotte sphérique 105
5.10 Évolution de u_z du sommet de la calotte en fonction du temps 106

Liste des tableaux

3.1 Comparaison des fréquences propres (en Hz) entre les valeurs théoriques et les résultats de OptSurf. 57
3.2 Comparaison des fréquences propres (en Hz) des deux plaques perpendiculaires. 59
3.3 Comparaison des fréquences propres (en Hz) du cylindre. 60
3.4 Nombre d'éléments et de degrés de liberté en fonction de la discrétisation par arête. . . 62

5.1 Comparaison entre les valeurs théoriques et calculées pour le benchmark de réponse forcée harmonique. 94
5.2 Comparaison du gradient par méthode adjointe selon la coordonnée z et par différences finies pour plusieurs points de contrôle de la plaque. 100
5.3 Comparaison entre les valeurs calculées et théoriques pour le cas NAFEMS 13T. 104

Introduction

L'industrie automobile doit constamment relever de nombreux défis liés à des normes sécuritaires strictes et des mesures environnementales exigeantes. Les constructeurs comme Renault font également face à la concurrence. Ils doivent être en mesure de proposer des véhicules de plus en plus performants au public et ceci le plus rapidement possible. Dans le but d'accélérer les processus de conception des nouveaux modèles, les outils numériques sont principalement employés, les essais réels se plaçant désormais en dernière phase de validation. L'optimisation automatique de formes présente dans ce contexte un intérêt certain. Elle permet de déterminer la meilleure configuration à donner à une pièce pour qu'elle puisse remplir au mieux sa fonction.

Aujourd'hui, l'une des principales préoccupations au sein de Renault est liée à la conception de structures surfaciques. En effet, les pièces en tôle mince sont les principaux éléments constituant une caisse automobile. Leur dimensionnement tient compte de trois familles de phénomènes : la vibro-acoustique, la tenue à la fatigue et le crash. Les méthodes classiquement utilisées pour l'optimisation de formes surfaciques présentent à ce jour certaines limites. Par exemple, les méthodes d'optimisation permettant d'explorer au maximum l'espace de design ne peuvent pas fournir de façon directe des résultats sous forme de modèle paramétré car elles nécessitent l'utilisation d'un maillage de la pièce étudiée. La conversion devant ensuite être mise en place a tendance à dégrader les résultats fournis par l'optimisation. Afin de remédier aux problèmes soulevés par les maillages de façon générale, une nouvelle méthode a été proposée pour remplacer les simulations par éléments finis classiques : l'analyse isogéométrique. Celle-ci propose en effet d'intégrer la définition exacte des géométries au sein des méthodes numériques utilisées pour les simulations.

L'objectif de ce manuscrit est de proposer une nouvelle méthode permettant de traiter des problèmes complexes d'optimisation, à l'aide de l'analyse isogéométrique, pour des critères de dimensionnement automobile utilisés chez Renault. Cette méthode consiste à exploiter les équations de coques en intégrant dans leur définition les informations géométriques issues des modèles paramétrés. Ceux-ci sont composés de patchs dont la forme est régie par des entités appelées points de contrôle qui seront naturellement choisis comme variables d'optimisation.

Le premier chapitre pose le cadre de l'optimisation de formes. Les principes généraux de l'optimisation sont rappelés ainsi que le contexte de la problématique industrielle. Dans un second temps, le chapitre

2 présente certains éléments (ou patchs) en lien avec les outils de la conception assistée par ordinateur (CAO) pour ensuite introduire l'analyse isogéométrique. Le modèle de coque de Koiter retenu pour les simulations des structures minces est également formulé. Le chapitre 3 traite de la mise en place dans les simulations de la méthode proposée dans le cadre de structures composées de plusieurs patchs. Le chapitre 4 développe la stratégie d'optimisation, suivant une méthode adjointe, mise en place pour un critère de compliance. Une procédure d'optimisation est ensuite présentée et testée sur plusieurs configurations dont une de nature industrielle. Les calculs des gradients par rapport à la forme et l'algorithme d'optimisation choisi y sont en particulier détaillés. Les problèmes d'optimisation de formes pour des critères en lien avec la réponse forcée des structures sont enfin abordés dans le chapitre 5. Ce chapitre met en avant un critère lié à la dynamique pouvant servir en première approche à un dimensionnement vibro-acoustique.

CHAPITRE

1 Principes généraux de l'optimisation de formes

Sommaire

1.1 Présentation de l'optimisation de formes . . . 3
1.1.1 Rappels sur la définition d'un problème d'optimisation . . . 3
1.1.2 Définition de l'optimisation de formes . . . 4
1.1.3 Description de la forme . . . 6
1.1.3.1 Optimisation de formes paramétrique . . . 6
1.1.3.2 Optimisation de formes géométrique . . . 7
1.1.3.3 Optimisation de formes topologique . . . 7
1.1.4 Bilan sur l'optimisation de formes . . . 9
1.2 L'optimisation de formes surfaciques . . . 9
1.2.1 Contexte industriel . . . 9
1.2.2 Paramétrage des formes surfaciques . . . 10
1.2.3 Méthodes d'optimisation . . . 12
1.2.3.1 Optimisation par plans d'expériences et surfaces de réponse . . . 12
1.2.3.2 Optimisation par méthodes de gradient . . . 13
1.2.3.3 Outils numériques existants . . . 14
1.3 Critères de dimensionnement en automobile . . . 15
1.3.1 Les critères vibratoires . . . 15
1.3.2 Les critères d'endurance . . . 15
1.3.3 Les critères de déformation en grands déplacements . . . 16
1.4 Synthèse du chapitre . . . 17

1.1 Présentation de l'optimisation de formes

1.1.1 Rappels sur la définition d'un problème d'optimisation

L'optimisation est un vaste domaine de recherche dont les applications concernent de nombreuses disciplines telles que la mécanique, l'économie, la gestion ou encore les sciences biologiques. Par définition, un problème d'optimisation consiste à trouver la configuration donnant la meilleure valeur à

une certaine quantité en tenant compte, si besoin, de contraintes. Prenons l'exemple d'un problème de dimensionnement de la citerne d'un camion-citerne. L'objectif est de consommer le moins de matière possible pour fabriquer la citerne sachant que cette dernière doit contenir un volume fixé. La quantité à minimiser, appelée fonction objectif ou fonction coût, est donc la surface de la citerne. Comme le montre la figure 1.1, la citerne est composée d'un cylindre de rayon x_1 et de longueur x_2 auquel deux calottes sphériques de hauteur x_3 s'ajoutent aux extrémités. Dans cette configuration, les variables x_1, x_2 et x_3 sont les variables d'optimisation. En plus du volume fixé, disons à 35 m^3, d'autres contraintes peuvent être prises en compte. En effet, nous pouvons considérer que le diamètre de la citerne ne peut excéder la taille de la cabine du camion soit 2,4 m et la longueur ne doit pas dépasser une certaine limite 13,2 m ici. Puisque les variables d'optimisation représentent des dimensions, il paraît naturel de choisir l'ensemble des réels positifs comme espace des variables admissibles. En d'autres termes, les variables (x_1, x_2, x_3) prennent leurs valeurs dans $\mathbb{R}_+$.

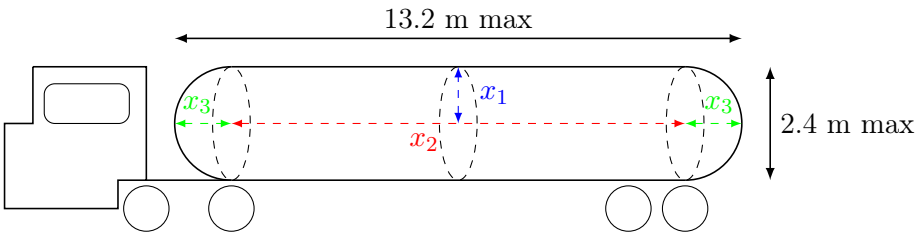


FIGURE 1.1: Schéma de la citerne - Les variables x_1, x_2 et x_3 sont les variables d'optimisation qui pilotent la forme de la citerne.

Ce problème d'optimisation classique peut être formalisé mathématiquement par l'équation suivante :

$$\begin{cases} \min\limits_{(x_1,x_2,x_3)\in(\mathbb{R}_+)^3} 2\pi(x_1^2 + x_3^2 + x_1x_2) \\ \text{tel que} \left| \begin{array}{l} x_2 + 2x_3 \leq 13.2 \\ 2x_1 \leq 2.40 \\ \dfrac{2\pi x_3^2}{3}\left(3\dfrac{x_1^2 + x_3^2}{2x_3} - x_3\right) + \pi x_3 x_1^2 = 35 \end{array} \right. \end{cases} \tag{1.1.1}$$

La résolution du problème (1.1.1) et, de façon générale, de tout problème d'optimisation peut être effectuée au moyen de différents algorithmes d'optimisation basés sur des méthodes déterministes (avec ou sans gradient) ou sur des méthodes stochastiques. Certaines de ces méthodes seront présentées dans la suite de ce chapitre.

L'écriture de ce problème simple permet de mettre en évidence les différents éléments constituant un problème d'optimisation : un espace de variables admissibles, une fonction objectif et selon les problèmes étudiés, des contraintes. Nous retrouverons ces éléments lors de la définition des problèmes d'optimisation de formes, sujet au cœur de ce manuscrit. Des ouvrages comme [Ciarlet, 2007] ou [Culioli, 2012] expliquent de façon approfondie les règles de l'optimisation.

1.1.2 Définition de l'optimisation de formes

L'optimisation de formes, aussi appelée optimisation structurale, est une branche de l'optimisation issue des questions de contrôle optimal. Elle a pour finalité la conception de la meilleure forme à donner à une pièce au regard de certains critères et contraintes. Plusieurs documents, [Delfour et Zolésio, 2001] ou [Henrot et Pierre, 2005] par exemple, proposent des introductions à l'optimisation de formes.

Il existe un très grand nombre de problèmes d'optimisation de formes connus. Le plus célèbre étant celui de la reine Didon lors de la fondation de l'actuelle Carthage. Le seigneur des terres où elle souhaite

s'établir accepte de lui céder la superficie « d'autant qu'il en pourrait tenir dans la peau d'un bœuf ». L'histoire raconte que la reine découpa en fines lanières la peau du bœuf et, qu'en s'appuyant sur la côte, forma un arc de cercle. Il est connu que la solution de ce problème isopérimétrique permettant d'obtenir la plus grande surface, est effectivement l'arc de cercle. Ce problème peut être vu de la manière suivante : il faut trouver, dans le plan, la courbe donneé par la fonction $x \mapsto u(x)$ et de longueur fixée l, qui entoure avec le segment $[a, b]$ reliant ses deux extrémités, le domaine ayant l'aire maximale. Il faut donc :

$$\text{trouver } u \text{ tel que} \quad \int_a^b u(x)\,dx \quad \text{est maximale, avec} \quad \int_a^b \sqrt{1+u'(x)^2}\,dx = l. \tag{1.1.2}$$

Le problème inverse est également souvent étudié : déterminer la surface minimale pour une frontière donnée. Soit Γ une courbe fermée de $\mathbb{R}^3$. Supposons que la surface soit paramétrée par $f(x, y) = (x, y, u(x, y))$ et que Γ est l'image du bord $\partial\Omega$ d'un domaine Ω de $\mathbb{R}^2$. Le problème est alors :

$$\text{trouver } u \text{ tel que} \quad \int_\Omega \sqrt{1+|\nabla u|^2}\,d\Omega \text{ est minimale avec} \quad u_{|\partial\Omega} = \Gamma. \tag{1.1.3}$$

Les problèmes de surfaces minimales les plus répandus restent les problèmes de la forme des bulles de savon et plus généralement les problèmes de surfaces capillaires.

Ces deux types de problème à caractère académique, ne sont bien évidemment pas les seuls. L'optimisation de formes présente un très grand intérêt dans les domaines associés à la mécanique de façon générale. En effet, l'objectif le plus courant est de trouver la forme optimale face à un phénomène physique précis. Un problème d'optimisation de formes est caractérisé par trois entités [Allaire, 2007] :

- un modèle : le plus souvent une équation aux dérivées partielles décrivant le comportement mécanique de la structure pour connaître l'état du système (équation de l'élasticité, de Navier-Stokes, équation de Maxwell etc...) ;
- un ensemble admissible de variables d'optimisation qui délimite l'espace de conception (par exemple : domaine limité par le respect des contraintes d'emboutissage) ;
- un critère (fonction objectif) qui est la prestation à améliorer (compliance, acoustique, masse etc...).

Une illustration dans le cadre de la mécanique des structures est l'optimisation d'une plaque occupant un domaine Ω de $\mathbb{R}^3$, soumise à une pression uniforme p sur une partie Γ_N de sa frontière (figure 1.2). Le matériau dont elle est constituée, de masse volumique ρ, est élastique, homogène et isotrope. Supposons que la plaque est encastrée sur une partie de son bord $\Gamma_D \subset \partial\Omega$. Les forces volumiques sont négligées. Le modèle utilisé est celui de l'élasticité linéarisée. Le critère choisi est la minimisation de l'énergie de déformation ou compliance. Pour que la forme soit admissible, la masse m de la structure tout comme la partie de sa frontière encastrée doit rester inchangée. Ce problème peut être formulé de la manière suivante :

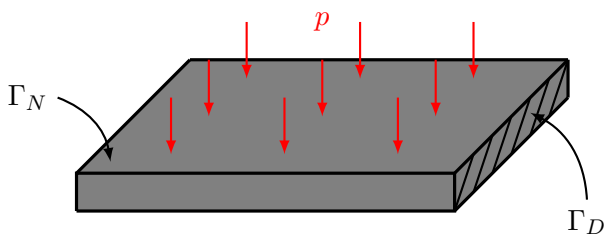


FIGURE 1.2: Plaque soumise à une pression uniforme sur Γ_N et encastrée sur Γ_D

$$\text{trouver } \Omega \in \mathcal{G}_{ad} \quad \text{tel que} \quad \int_\Omega \vec{g}(\vec{x}).\vec{u}(\vec{x})\, dx \quad \text{est minimale} \tag{1.1.4}$$

où le déplacement $\vec{u}(\vec{x}) : \Omega \to \mathbb{R}^3$ vérifie :

$$\begin{cases} -\text{div}\underline{\underline{\sigma}} = 0 & \text{dans } \Omega \\ \vec{u}(\vec{x}) = 0 & \text{sur } \Gamma_D \\ \underline{\underline{\sigma}}.\vec{n}(\vec{x}) = \vec{g}(\vec{x}) = -p.\vec{n}(\vec{x}) & \text{sur } \Gamma_N \\ \underline{\underline{\sigma}}.\vec{n}(\vec{x}) = 0 & \text{sur } \partial\Omega \backslash \Gamma_D \cup \Gamma_N \end{cases} \tag{1.1.5}$$

avec

- $\underline{\underline{\sigma}}(\vec{x}) = \underline{\underline{C}}(\vec{x}) : \underline{\underline{\varepsilon}}(\vec{x})$, le tenseur des contraintes
- $\underline{\underline{C}}(\vec{x})$ est le tenseur d'élasticité ;
- $\underline{\underline{\varepsilon}} = \dfrac{1}{2}(\nabla\vec{u}(\vec{x}) + \nabla\vec{u}(\vec{x}))^T$ est le tenseur des déformations linéarisé ;
- $\vec{n}(\vec{x})$ le vecteur normal à la surface en $\vec{x}$;

et

$$\mathcal{G}_{ad} = \left\{ \Omega \subset \mathbb{R}^3, \quad \Gamma_D \subset \partial\Omega, \quad \int_\Omega \rho\, dx = m \right\}. \tag{1.1.6}$$

Une autre application, cette fois issue de l'aéronautique et donc en lien avec la mécanique des fluides, est la recherche de la forme optimale de l'aile d'un avion qui minimise la traînée. Nous ne détaillerons pas ce problème assez complexe mais rappellerons que la fonction objectif $J(\Omega)$ est donnée par :

$$J(\Omega) = F\vec{u}_\infty \quad \text{tel que} \quad F = \int_\Omega \left[\mu(\nabla\vec{u} + \nabla\vec{u}^T) - \frac{2\mu}{3}\text{div}\vec{u} \right] \vec{n}\, d\Omega - \int_\Omega \vec{p}.\vec{n}\, d\Omega \tag{1.1.7}$$

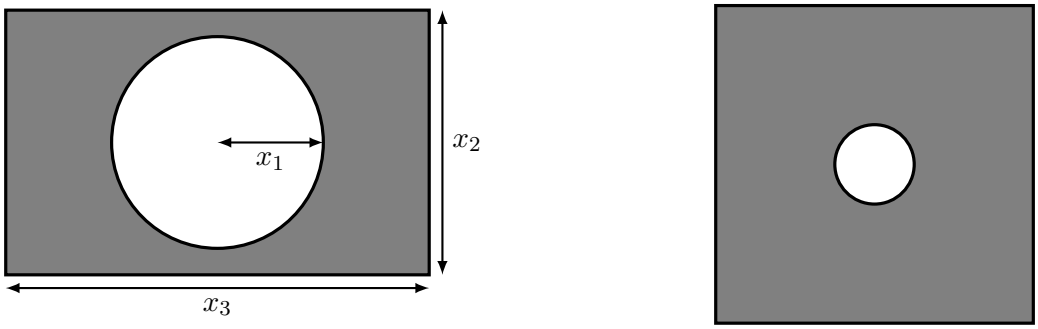
avec $\vec{u}$ la vitesse du fluide, $\vec{u}_\infty$ la vitesse à l'infini, μ la viscosité, $\vec{n}$ la normale extérieure au domaine occupé par le fluide et $\vec{p}$ la pression de ce dernier. Les différentes quantités ($\vec{u}$ et $\vec{p}$) sont déterminées par les équations de Navier-Stokes. Une explication complète concernant ce problème et l'optimisation de formes en mécanique des fluides est présentée dans [Mohammadi et Pironneau, 2009]. Les questions d'optimisation de formes qui seront abordées au sein de ce manuscrit ne traiteront que de problèmes issus de la mécanique des structures et seront proches dans leur formulation du cas de la plaque sous pression évoqué ci-dessus.

1.1.3 Description de la forme

L'exemple de la plaque sous pression comprend les trois éléments d'un problème d'optimisation de formes listés auparavant : un modèle (1.1.5), un critère (1.1.4) et un ensemble de formes admissibles (1.1.6). Il reste toutefois un point essentiel à clarifier. Dans l'exemple de la plaque, la variable d'optimisation est le domaine occupé par la structure Ω. Il faut donc choisir comment représenter le domaine Ω. En fonction du paramétrage de la forme, le problème d'optimisation associé peut être classé dans un des trois types d'optimisation de formes : paramétrique, géométrique ou topologique.

1.1.3.1 Optimisation de formes paramétrique

Dans le cadre de l'optimisation de formes paramétrique, la forme est régie par un petit nombre de paramètres. Ces paramètres ont en général une signification physique et la dépendance du modèle vis à vis de ces derniers est explicite. Il peut s'agir de longueur, de rayon, d'angle ou encore d'épaisseur. Au cours de l'optimisation, la forme sera modifiée au travers de l'évolution des paramètres présents dans le modèle. Cette approche a pour effet de limiter l'exploration de l'espace de conception puisqu'elle n'offre pas de variation de la frontière de la forme ou de sa topologie. La forme optimale trouvée sera semblable à l'initiale mais avec des dimensions différentes. [Banichuk, 2010] et [Haslinger et Mäkinen, 2003] avancent plusieurs résultats concernant l'optimisation de formes paramétrique.



(a) Forme originale (b) Forme finale

FIGURE 1.3: Illustration de l'optimisation de formes paramétrique : la forme est modifiée par variation des paramètres x_1, x_2 et x_3.

1.1.3.2 Optimisation de formes géométrique

L'optimisation géométrique contrairement à l'optimisation paramétrique, permet de faire varier la frontière du domaine d'étude. Cependant, elle ne peut pas changer la topologie de la forme. De façon imagée, nous pouvons dire que l'optimisation géométrique ne permet pas de faire apparaître des trous en dimension 2 ou des boucles et des anses en dimension 3.

(a) Forme originale (b) Forme finale

FIGURE 1.4: Illustration de l'optimisation de formes géométrique : la forme est modifiée par variation des frontières sans changement de la topologie.

Avec cette méthode, il devient plus complexe de décrire la forme. La méthode proposée par Hadamard [Hadamard, 1908] est la plus utilisée. Elle permet surtout de définir un cadre pour la dérivation par rapport au domaine afin de pouvoir utiliser des algorithmes d'optimisation basés sur les gradients. Appelons Ω_0 la forme initiale dite domaine de référence. La forme Ω qui est une variation de Ω_0 est le résultat d'une perturbation $\vec{\theta}(\vec{x})$. $\vec{\theta}(\vec{x})$ est une fonction vectorielle de $\mathbb{R}^n$ (n=2 ou n=3) à valeurs dans $\mathbb{R}^n$ telle que :

$$\Omega = \{\vec{x} + \vec{\theta}(\vec{x}) \quad | \quad \vec{x} \in \Omega_0\} \tag{1.1.8}$$

$\vec{\theta}(\vec{x})$ agit comme un vecteur de déplacement à appliquer à Ω_0 pour déterminer Ω (voir figure 1.5). Le calcul d'une dérivée par rapport au domaine est obtenu en dérivant par rapport à $\vec{\theta}$. Pour davantage de précisions se référer à [Dervieux et Palmerio, 1976], [Sokolowski et Zolésio, 1992] et [Laporte et Le Tallec, 2003].

1.1.3.3 Optimisation de formes topologique

Le dernier type d'optimisation, l'optimisation topologique, est celui qui laisse le plus de liberté quant à l'exploitation de l'espace de design. À la différence de l'optimisation géométrique ou paramétrique, l'optimisation topologique autorise le changement de topologie. Les seules restrictions applicables à ce

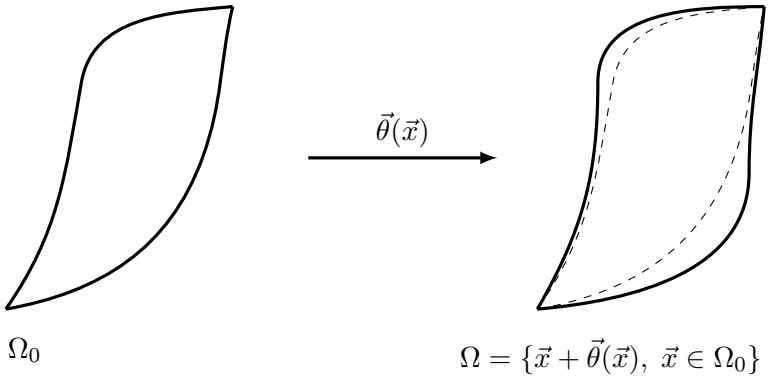
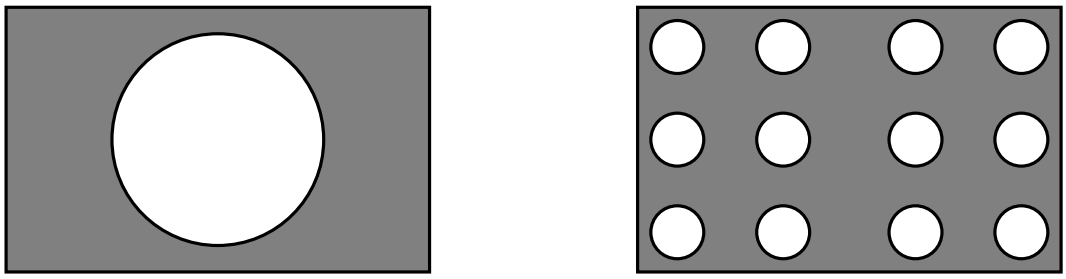


FIGURE 1.5: Méthode de Hadamard - perturbation de la frontière Ω par un vecteur $\vec{\theta}(\vec{x})$.

type de problèmes sont celles établies par l'espace des formes admissibles $\mathcal{G}_{ad}$. Les deux approches les plus populaires sont la méthode d'homogénéisation et celle des lignes de niveau.



(a) Forme originale (b) Forme finale

FIGURE 1.6: Illustration de l'optimisation de formes topologique : la topologie de la forme peut être modifiée.

La première approche consiste à voir le problème d'optimisation comme une optimisation de la densité de matière composant la structure [Bendsøe et Sigmund, 2004]. La fonction associée à la densité prend des valeurs entre 0 et 1, 0 représentant le vide (absence de matière) et 1 un matériau plein. Les valeurs entre les bornes sont associées à des matériaux poreux ou composites. Grâce à ce procédé, la topologie de la forme peut changer puisque les modifications du domaine ne sont pas dues à un simple déplacement de la frontière. Par ailleurs, la présence de matériau composite n'étant pas toujours recherchée, une pénalisation sur ces types de matériaux peut être appliquée dans l'algorithme d'optimisation pour s'assurer d'avoir des matériaux pleins. Cette approche avec pénalisation porte le nom de méthode SIMP pour *Solid Isotropic Material with Penalisation*.

La seconde technique, la méthode des lignes niveau [Allaire *et al.*, 2004], combine à la fois l'algorithme des lignes de niveau de Osher et Sethian [Osher et Santosa, 1988] et la dérivation du domaine au sens de Hadamard. Soient Ω la forme et ψ une fonction de $\mathbb{R}^n$ à valeurs dans $\mathbb{R}$. La fonction ψ, appelée fonction de ligne de niveau, vérifie :

$$\begin{cases} \psi(\vec{x}) < 0 & \text{si} \quad \vec{x} \in \Omega \backslash \partial\Omega; \\ \psi(\vec{x}) > 0 & \text{si} \quad \vec{x} \notin \Omega; \\ \psi(\vec{x}) = 0 & \text{si} \quad \vec{x} \in \partial\Omega. \end{cases} \tag{1.1.9}$$

La méthode de Hadamard permet tout d'abord de déterminer un gradient par rapport à la forme. Au cours de l'optimisation, la fonction ψ évolue dans tout l'espace. Le bord de la forme optimale est déterminé à la fin de l'itération en exploitant la ligne de niveau 0.

1.1.4 Bilan sur l'optimisation de formes

Dans cette première partie, nous avons introduit le concept d'optimisation de formes. La présentation qui en a été faite ne fait que rappeler les grands principes. La définition du problème avec le choix du modèle, des critères et de l'espace de solutions admissibles occuperont une place centrale dans ce manuscrit. Un exposé complet sur les différents types d'optimisation de formes (paramétrique, géométrique et topologique) est disponible dans [Allaire, 2007]. Il est important de rappeler que chacune de ces approches fait intervenir une paramétrisation différente et que, par conséquent, le choix d'une méthode plutôt qu'une autre est directement lié au type de problème étudié. Certains points centraux de l'optimisation de formes n'ont pas été abordés ici. Par exemple, la question de l'existence de formes optimales pour les différents types d'optimisation est essentielle dans ce cadre. Là encore, nous renvoyons à la bibliographie où ce sujet est traité de façon exhaustive [Murat et Simon, 1976, Chambolle, 2003, Chenais, 1975]. Dans la partie suivante, nous verrons quelle place occupent ces différentes approches au sein de l'industrie et plus particulièrement chez Renault. Un rappel sur les méthodes et plus précisément sur les algorithmes d'optimisation déployés dans l'industrie sera également fait.

1.2 L'optimisation de formes surfaciques

1.2.1 Contexte industriel

À l'heure où l'industrie automobile fait face à de nombreux défis (écologie, sécurité etc...), l'optimisation est de plus en plus présente dans le processus numérique de conception. Les pièces qui constituent un véhicule peuvent être réparties en deux catégories : les pièces volumiques (culasse, bielles etc.) et les pièces surfaciques (pavillon, plancher ou tablier par exemple). En automobile, on considère que ces dernières représentent environ 70% d'une voiture comme l'illustre la figure 1.7. Leur optimisation présente donc un grand intérêt pour les constructeurs.

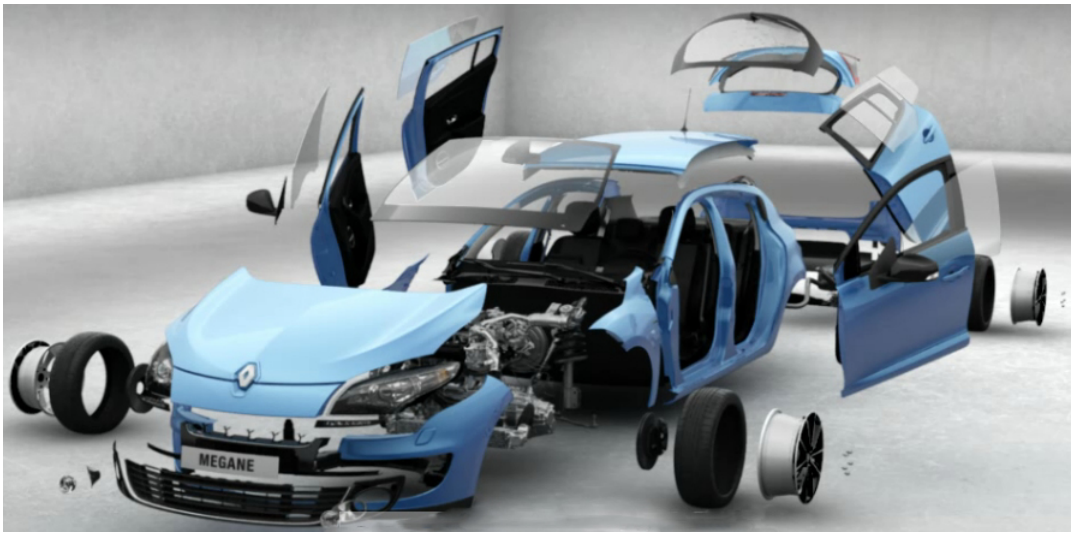


FIGURE 1.7: Vue éclatée de la Mégane où l'on peut distinguer les éléments surfaciques (source : [de Nazelle, 2013]).

Depuis plusieurs années, Renault consacre des ressources aux questions d'optimisation. Renault est partenaire du projet RODIN (*Robust structural Optimization for Design in INdustry*) au sein duquel est développé un logiciel d'optimisation de formes topologique basé sur la méthode des lignes de niveaux [Abballe *et al.*, 2015]. À terme, ce logiciel appartiendra aux solutions pour l'optimisation des pièces volumiques chez Renault. Pour en revenir aux structures surfaciques, plusieurs approches sont utilisées. Leurs principales différences se situent au niveau du paramétrage de la forme et des méthodes d'optimisation.

1.2.2 Paramétrage des formes surfaciques

Comme nous l'avons vu à la section 1.1.3, il existe différents types de paramétrage de formes. Chez Renault, l'optimisation topologique est généralement réservée aux pièces massives. Pour des pièces surfaciques, les autres paramétrages, paramétrique et géométrique, sont couramment utilisés.

L'optimisation de formes paramétrique est une approche naturelle en industrie. Les pièces à optimiser sont en général définies sous forme de modèles CAO (*Conception Assistée par Ordinateur*) pilotés par des paramètres CAO aussi appelés cotes (des longueurs, des angles etc.) permettant de dessiner la forme (voir figure 1.8) .

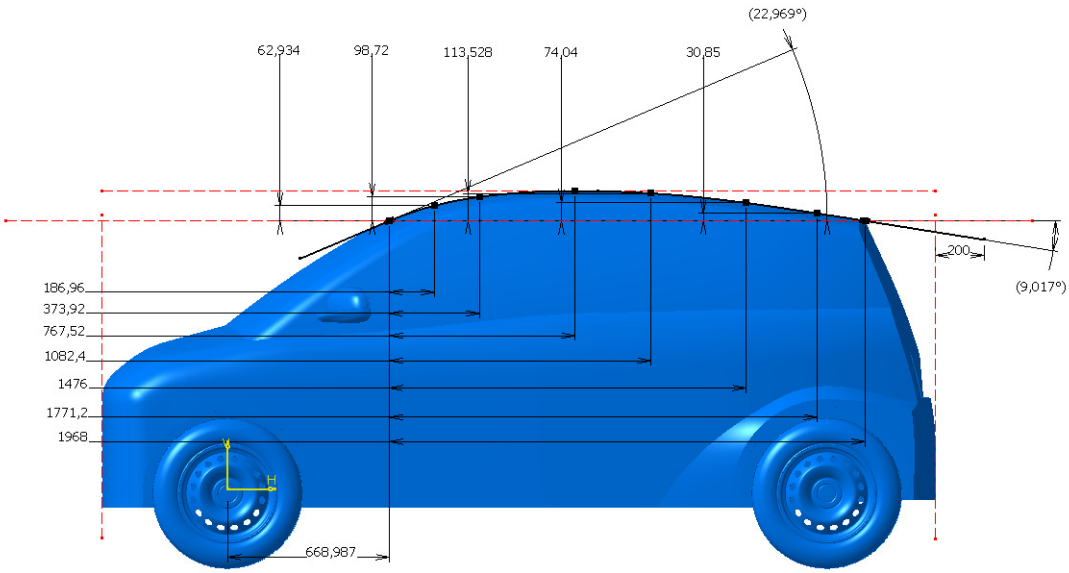


FIGURE 1.8: Exemple de paramétrage CAO.

Ce sont ces cotes qui vont être les variables d'optimisation. En pratique, la marche à suivre est la suivante (figure 1.9) : créer un modèle CAO de la forme en question, faire une simulation, modifier les paramètres en fonction des résultats obtenus et, enfin, mettre à jour le modèle CAO [Hardee *et al.*, 1999, Brujic *et al.*, 2010].

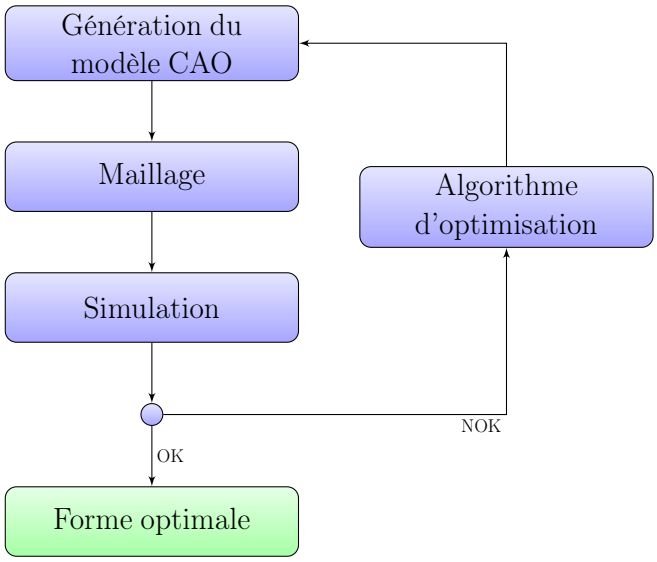


FIGURE 1.9: Déroulement de l'optimisation paramétrique.

La simplicité de cette technique en fait une méthode attractive. Ce procédé possède également un autre avantage : la forme finale obtenue est un modèle CAO dont tous les paramètres pour le dessiner sont connus, ce qui est appréciable pour la mise en plan et la suite des processus industriel. Cette qualité ne se retouve pas en optimisation géométrique. Toutefois, rappelons qu'avec cette pratique, l'espace de design reste restreint. D'autre part, d'un point de vue opérationnel, la mise en place d'une telle boucle d'optimisation, peu intrusive vis à vis des codes de calculs, peut néanmoins s'avérer fastidieuse. En effet, il est nécessaire de mettre au point un certain nombre de scripts afin d'automatiser l'enchaînement des différents logiciels.

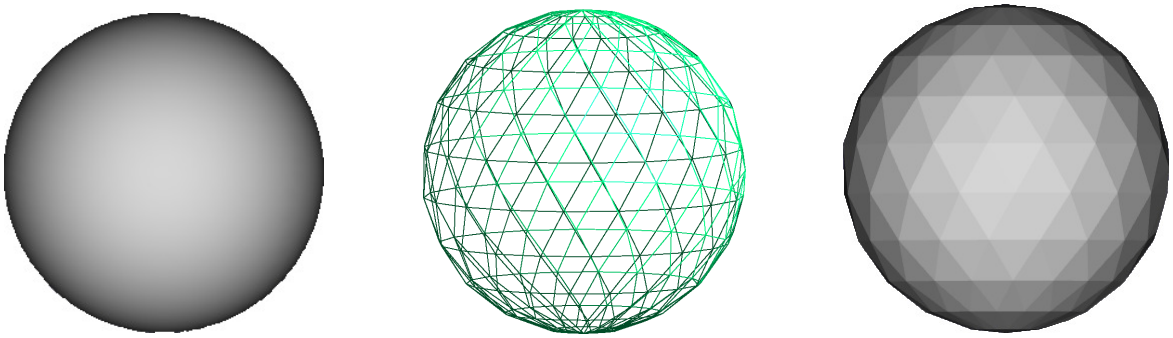
En optimisation géométrique, la forme est optimisée par variation de sa frontière. D'un point de vue numérique, la frontière est définie par le maillage de la forme. Pour faire varier la frontière, il faut faire varier le maillage [Braibant et Morelle, 1990]. Deux possibilités s'offrent aux concepteurs pour le choix des variables d'optimisation [Bruyneel *et al.*, 2014] :

- La méthode « free form » où chaque position de chaque nœud du maillage est une variable d'optimisation. Ce type d'approche conduit à des problèmes en grande dimension. En laissant libre d'agir chacun des nœuds, la frontière peut perdre de sa régularité.

- Le « morphing » de maillage ou optimisation topographique où seulement certains groupes de nœuds sont les variables d'optimisation. Au cours de l'optimisation, les nœuds au sein de chaque groupe bougent de concert.

L'un des inconvénients de ce type de méthode est qu'au cours de l'optimisation, les nombreuses variations des nœuds peuvent dégrader la qualité du maillage et par suite, la validité des résultats de simulation. À la suite de ce phénomène, la fiabilité de la forme optimale doit elle aussi à son tour être remise en question. Pour remédier à cette situation, une technique peut être de travailler avec deux maillages : le premier, assez grossier, pour lequel les nœuds seront des variables d'optimisation, le second beaucoup plus fin, obtenu par interpolation du maillage grossier servira pour les simulations. Une stratégie, utilisant les lignes de niveau, pour gérer l'évolution du maillage au cours de l'optimisation est proposée dans [Dapogny, 2013].

D'un point de vue industriel, la forme optimale est souvent désirée sous la forme d'un modèle CAO. L'étape de conversion du maillage vers un objet CAO peut paraître compliquée. Grossièrement, comme le montre la figure 1.10, il s'agit de transformer une forme facetisée en une forme continue.



(a) CAO d'une sphère (b) Maillage de la sphère (c) Reconstruction de la sphère

FIGURE 1.10: Reconstruction du modèle CAO d'une sphère à partir d'un maillage.

Affiner la discrétisation semble être une solution mais cela aurait pour conséquence l'augmentation du nombre de variables d'optimisation. Par ailleurs, il faut noter qu'une fois la forme maillée, les paramètres CAO ne sont plus pris en compte. En d'autres termes, il n'est pas possible de suivre leur évolution au cours de l'optimisation.

Nous présentons enfin une dernière méthode entre l'optimisation de formes géométrique et paramétrique. L'idée consiste à prendre des paramètres CAO comme variables d'optimisation mais qui ne sont pas des cotes. Ces variables se nomment les points de contrôle des surfaces définissant les modèles CAO. La variation des points de contrôle a pour effet le changement de la frontière de la structure étudiée. Dans ce cas, les problèmes dus au changement de maillage n'ont plus lieu d'être puisque c'est la forme exacte qui est modifiée. Une méthode de paramétrage pour l'optimisation de formes surfaciques a été mise en place dans [de Nazelle, 2013]. C'est cette approche que nous utiliserons tout au long de cette thèse. Avec cette méthode, l'espace de design est davantage exploré et le retour à un modèle CAO en fin d'optimisation n'est plus un obstacle. En utilisant cette pratique, les simulations sont également plus précises puisque la géométrie exacte est exploitée plutôt qu'un maillage. Ce genre d'approche appartient aux méthodes dites isogéométriques [Cottrell *et al.*, 2009]. L'analyse isogéométrique a pour vocation d'effectuer les calculs non pas sur maillage classique mais directement sur les surfaces du modèle CAO. Une présentation plus complète des objets CAO et de la méthode de paramétrage choisie sera faite dans la chapitre 2.

1.2.3 Méthodes d'optimisation

Une fois le paramétrage de la forme choisi, une procédure d'optimisation peut être mise en place. La formulation générale d'un problème d'optimisation est la suivante :

$$\text{trouver } x \in \mathcal{G}_{ad} \quad \text{qui minimise } J(x) \text{ et tel que} \quad \begin{cases} g_j(x) \leqslant 0, & j = 1, \dots, n \\ h_l(x) = 0, & l = 1, \dots, m \end{cases} \tag{1.2.1}$$

où :

- J est la fonction objectif ;
- $\mathcal{G}_{ad}$ l'ensemble des variables admissibles ;
- g_j les n contraintes d'inégalité ;
- h_l les m contraintes d'égalité.

Après avoir posé le problème d'optimisation, l'étape suivante consiste à adopter une stratégie pour le résoudre. L'optimisation par plans d'expériences et surfaces de réponse ainsi que l'optimisation par des méthodes de gradient sont les approches les plus répandues dans l'industrie. Nous n'entrerons pas dans les détails de la première car c'est une méthodologie que nous n'utiliserons pas dans la suite. Une courte description est seulement proposée ici. Pour plus d'explications, on pourra se référer à [Goupy, 2013]. La méthode de paramétrage retenue dans ces travaux permet de déterminer les gradients de façon immédiate. L'information des gradients étant disponible, le choix des méthodes d'optimisation s'est donc orienté vers des techniques exploitant ces informations.

1.2.3.1 Optimisation par plans d'expériences et surfaces de réponse

Le but des méthodes par plans d'expériences est de déterminer une fonction f qui lie la réponse notée y, aux différents facteurs $x_1, x_2, \dots, x_n$:

$$y = f(x_1, x_2, \dots, x_n)$$

Un plan d'expériences regroupe l'ensemble des expériences (calculs ou essais) à mener pour pouvoir déterminer la fonction f. Il existe plusieurs types de plans d'expériences comme les plans orthogonaux, les plans composites ou encore les plans de Box-Behnken. L'objectif lors de la sélection du plan d'expériences est d'obtenir le maximum d'information avec le moins d'expériences possibles. La surface de réponse est établie à l'aide d'un modèle polynomial ou statistique en se basant sur les résultats du plan. Dans le cadre de l'optimisation, la surface de réponse à reconstruire correspond à la fonction objectif J. Les contraintes du problème sont intégrées à la conception du plan d'expériences. L'optimum de

la fonction objectif est repéré alors sur la surface de réponse. Afin d'améliorer les résultats trouvés, de nouveaux essais peuvent être générés. L'optimisation par plans d'expérience et surfaces de réponse consiste donc à balayer l'ensemble du domaine de conception et à en retirer la meilleure configuration. Le recours aux plans d'expérience est une solution pour l'optimisation peu intrusive vis à vis des codes de calcul. Toutefois, cette méthode peut rapidement souffrir du nombre important de calculs à effectuer afin de trouver un optimum.

1.2.3.2 Optimisation par méthodes de gradient

Il est connu que le comportement de la dérivée d'une fonction permet de déterminer le sens de variation de cette dernière. Elle permet en particulier d'en repérer les optimums. Avant d'évoquer les algorithmes d'optimisation par gradient, rappelons la définition de la différentiabilité que nous considérerons tout au long de ce manuscrit.

Définition 1.2.1 (Différentiabilité au sens de Fréchet)**.** Soient V un espace de Banach réel muni de la norme $||.||_V$ et V' son espace dual. Soit J une fonction continue et définie sur un voisinage de $x \in V$ à valeurs réelles. J est dérivable (ou différentiable) au sens de Fréchet en x s'il existe une forme linéaire continue $L \in V'$ telle que :

$$\forall h \in V \quad J(x+h) = J(x) + L(h) + o(||h||_V) \tag{1.2.2}$$

Le gradient, qui représente les dérivées de la fonction objectif par rapport aux variables, est utilisé comme direction de descente. Les algorithmes consistent donc à « suivre » la direction opposée pour trouver le minimum. Les algorithmes d'optimisation utilisant les gradients peuvent être classés en deux grandes catégories, avec ou sans contraintes. Pour les problèmes sans contraintes, il s'agit de résoudre un problème similaire à l'équation (1.2.1) mais en ignorant les contraintes g_i et h_j.

Soient J une fonction de $\mathbb{R}^n$ dans $\mathbb{R}$ et son gradient $\overrightarrow{\nabla J} = \left[\frac{\partial J}{\partial x_1}, \ldots, \frac{\partial J}{\partial x_n}\right]^T \in \mathbb{R}^n$. L'algorithme 1 donne un algorithme de gradient classique pour un problème sans contraintes.

Algorithme 1 : Algorithme de gradient à pas variable, sans contraintes

Entrée : $\vec{x}^0$ le point initial et $\varepsilon \geq 0$ la tolérance
Sortie : $\vec{x}^*$

1 Initialisation : $k = 0$;
2 **tant que** $||\overrightarrow{\nabla J}(\vec{x}^k)|| > \varepsilon$ **faire**
3 Calcul du pas $\alpha^k > 0$;
4 $\vec{x}^{k+1} = \vec{x}^k - \alpha^k \overrightarrow{\nabla J}(\vec{x}^k)$;
5 $k = k + 1$;
6 **fin**
7 Fin : $\vec{x}^* = \vec{x}^k$

À la ligne 3 de l'algorithme 1, une étape consiste à déterminer le pas de descente. En général, soit une valeur de pas est fixée et, dans ce cas, on parle d'algorithme à pas fixe soit il s'agit d'algorithme à pas variable. Pour les algorithmes à pas variable, ce dernier est estimé à l'aide de recherches linéaires en utilisant des méthodes de Wolfe ou Armijo par exemple. Il existe des variantes de cet algorithme comme le gradient conjugué. Un autre algorithme souvent recommandé est l'algorithme de Newton. Il s'agit d'un algorithme d'optimisation d'ordre 2 exigeant le calcul de la matrice Hessienne. En pratique, une méthode de quasi-Newton, où la Hessienne est approchée, est souvent appliquée. L'algorithme 2, appelé méthode BFGS (Broyden-Fltecher-GoldFarb-Shanno), appartient à cette classe de méthodes.

Algorithme 2 : Algorithme BFGS

Entrée : $\vec{x}^0$ le point initial, $B_0 = Id$ et $\varepsilon \geq 0$ la tolérance
Sortie : $\vec{x}^*$

1 Initialisation : $k = 0$;
2 **tant que** $||\overrightarrow{\nabla J}(\vec{x}^k)|| > \varepsilon$ **faire**
3 Résoudre $B^k \vec{p}^k = -\overrightarrow{\nabla J}(\vec{x}^k)$;
4 Calcul du pas α^k par recherche linéaire ;
5 $\vec{x}^{k+1} = \vec{x}^k + \alpha^k \vec{p}^k = \vec{x}^k + \vec{s}^k$;
6 $\vec{y}^k = \overrightarrow{\nabla J}(\vec{x}^{k+1}) - \overrightarrow{\nabla J}(\vec{x}^k)$;
7 $B^{k+1} = B^k + (\vec{y}^k \vec{y}^{kT})/(\vec{y}^{kT} \vec{s}^k) - (B^k \vec{s}^k \vec{s}^{kT} B^k)/(\vec{s}^{kT} B^k \vec{s}^k)$;
8 $k = k + 1$;
9 **fin**
10 Fin : $\vec{x}^* = \vec{x}^k$

Pour l'optimisation sous contraintes, deux types d'approches sont considérées. La première démarche se base sur l'utilisation d'un Lagrangien. Définissons le Lagrangien du problème (1.2.1) en ne tenant compte que des contraintes d'inégalité notées $\vec{g}$, les contraintes d'égalité pouvant être ramenées à des contraintes d'inégalités. Soit $\vec{p} \in \mathbb{R}^m_+$, le Lagrangien est donné par l'équation (1.2.3) :

$$\mathcal{L}(\vec{x}, \vec{p}) = J(\vec{x}) + \vec{p}^T.\vec{g}(\vec{x}) \tag{1.2.3}$$

Les algorithmes d'optimisation exploitant le Lagrangien comme les algorithmes de Uzawa ou du Lagragien augmenté, recherchent alors un point selle de $\mathcal{L}$ en vertu des théorèmes d'optimalité de celui-ci. L'autre approche possible, qui sera utilisée dans la suite, est la pénalisation des contraintes. Il s'agit d'approcher le problème de minimisation (1.2.1) par une suite de problème d'optimisation sans contraintes. Pour $\varepsilon > 0$, le problème suivant est introduit :

$$\min_{x \in \mathcal{G}_{ad}} \left(J(\vec{x}) + \frac{1}{\varepsilon} \sum_{i=1}^{m} \max[g_i(\vec{x}), 0] \right). \tag{1.2.4}$$

Ce problème sans contraintes est alors résolu avec les algorithmes de gradient.

Les résultats de convergence des différents algorithmes de gradient sont directement liés à la convexité de la fonction objectif et de l'espace des solutions admissibles. En effet, il est établi que ces algorithmes fournissent de bons résultats en cas de convexité. En cas de non convexité, il n'est pas assuré que l'algorithme converge vers un optimum. À noter que les problèmes d'optimisation de formes, au vu de leur espace de définition et des critères étudiés, s'avèrent souvent ne pas être convexes.

1.2.3.3 Outils numériques existants

Comme nous l'avons vu, en automobile, deux types de méthodes d'optimisation sont utilisés. Les méthodes par plans d'expériences ont l'avantage d'être non intrusives vis à vis des codes de calculs mais les délais peuvent être longs. Aujourd'hui, plusieurs codes de calculs comme OptiStruct© d'Altair ou Nastran© de MSC intègrent des modules pour l'optimisation. De fait, ils intègrent le calcul des gradient par rapport aux nœuds du maillage. À ce jour, il n'existe pas de logiciel capable de donner les dérivées par rapport à des paramètres CAO. L'utilisation des plans d'expériences et surfaces de réponse dans ce cas est choisie. À noter que dans la thèse de [Froment, 2014], une technique est développée pour déterminer le gradient de forme par rapport aux paramètres CAO en utilisant le gradient par rapport aux nœuds.

1.3 Critères de dimensionnement en automobile

Lors de la conception de produits automobiles, l'amélioration des prestations est toujours recherchée. Il peut s'agir par exemple de vouloir minimiser la masse pour réduire la consommation de carburant ou simplement limiter les côuts. Les critères à optimiser varient selon les pièces. Dans le cadre des structures surfaciques, on compte trois grandes familles de critères de dimensionemment : la vibro-acoustique, la tenue à la fatigue et les déformations en grands déplacements.

1.3.1 Les critères vibratoires

L'étude des critères vibratoires, souvent appelés NVH (*Noise Vibration and Harshness*), est essentielle en automobile. Elle regroupe à la fois les phénomènes purement vibratoires et leur contribution au niveau du comportement acoustique du véhicule. Les principaux critères que l'on cherche à optimiser sont en général :

- la première fréquence propre que l'on cherche à repousser au-delà d'une fréquence cible (fréquence du groupe motopropulseur par exemple) pour éviter que les pièces n'entrent en résonance ;
- les fonctions de transfert ou les inertances pour maîtriser la réponse vibratoire de la structure pour une excitation donnée ;
- le rayonnement acoustique des structures afin de limiter le bruit engendré par les pièces lorsqu'elles vibrent.

FIGURE 1.11: Test acoustique du groupe motopropulseur - source : www.nordfranceinvest.fr

De nombreux documents comme [Marburg et Hardtke, 2002], [Zimmer *et al.*, 2012] ou encore [Ma *et al.*, 1994] abordent ces différents critères. La plupart des procédures proposées ont recourt à des gradients par rapport aux nœuds d'un maillage. Notre objectif étant de conduire l'optimisation sur la géométrie exacte, il sera nécessaire de s'assurer de la dérivabilité des fonctions objectifs et d'exprimer les gradients par rapport à la forme. En effet, rappelons, que certaines de ces fonctions coût ont la particularité de ne pas être toujours différentiables.

1.3.2 Les critères d'endurance

Pour garantir, la fiabilité des pièces automobiles dans le temps, des simulations de la durée de vie des structures sont réalisées. Aujourd'hui, pour améliorer l'endurance et maîtriser davantage la fatigue des structures, l'optimisation de formes est de plus en plus utilisée comme le rappellent [Schnack et Weikl,

2001] et [Häußler et Albers, 2005]. Les études en fatigue reposent essentiellement sur les calculs de cumul de dommage. La plupart du temps, lors des simulations, le dommage est défini de façon discrète au niveau temporel. Dans [Fourcade, 2015], une approche pour calculer le dommage de façon continue et réaliser des optimisations de formes avec gradient est développée.

FIGURE 1.12: Essais de fatigue sur des pédales - source : Laboratoire Centre Technologie de l'Automobile de Galice.

1.3.3 Les critères de déformation en grands déplacements

La principale préoccupation au sein de l'industrie automobile est la sécurité. Il s'agit aussi bien de la sécurité des occupants d'un véhicule [Sinha, 2007] que celle des piétons qui pourraient être heurtés par une voiture [Henn, 1998]. La résistance au choc, ou crash, occupe donc une place importante au sein des questions d'optimisation de formes.

FIGURE 1.13: Crash test - source : EuroNCAP 2015.

L'un des aspects les plus contraignant vis à vis des études de crash est le temps de simulation. La majorité des optimisations étant effectuée au moyen de plans d'expériences, voir par exemple [Kurtaran *et al.*, 2002], le délai pour obtenir une solution optimale peut s'avérer extrêmement long. Dans [Genest, 2016], il est introduit un processus d'optimisation pour des critères liés au crash en utilisant une estimation du gradient de forme. En raison de leur complexité à mettre en œuvre, ces critères ne seront pas abordés au sein de ce manuscrit.

1.4 Synthèse du chapitre

Dans ce chapitre, nous avons introduit le principe de l'optimisation de formes et comment elle était mise à profit au sein de l'industrie. En automobile, elle est notamment adoptée pour la conception des structures surfaciques minces afin d'apporter les meilleures prestations vis à vis des critères de dimensionnement. Dans cette thèse, nous mettrons en particulier l'accent sur les critères de types vibratoires sachant qu'ils regroupent plusieurs problématiques pour l'automobile. Les méthodes actuelles pour l'optimisation de formes des coques présentent par ailleurs certaines limites. En effet, les industriels désirent pour des raisons pratiques que les pièces issues de l'optimisation soient sous forme de modèle CAO. L'optimisation géométrique qui fournit des résultats sous forme de maillage peut voir ses résultats dégradés par la conversion du maillage vers un modèle CAO. L'optimisation paramétrique, qui exporte des modèles CAO, restreint l'exploration de l'espace de conception en raison de son paramétrage. La méthode que nous allons introduire peut être une solution car elle permet d'explorer davantage l'espace de conception, comme l'optimisation géométrique classique, tout en fournissant un modèle CAO en sortie. De plus, en faisant usage des gradients lors de l'optimisation nous pourrons traiter un très grand nombre de paramètres avec une vitesse de convergence meilleure que pour une méthode de type surface de réponse.

CHAPITRE

2 Analyse isogéométrique et équations de coques

Sommaire

2.1 Introduction **19**
2.2 Les patchs CAO **20**
2.2.1 Les patchs de Bézier 20
2.2.2 Les surfaces B-splines 21
2.2.3 Les patchs NURBS 24
2.3 Le concept de l'analyse isogéométrique **26**
2.3.1 Définition de l'analyse géométrique 26
2.3.2 Maillage des NURBS 26
2.3.3 Méthodes numériques en analyse isogéoémtrique 26
2.3.3.1 La méthode de Galerkin 27
2.3.3.2 La méthode de collocation 28
2.3.4 Bilan sur l'analyse isogéométrique 28
2.4 Les équations de coques **28**
2.4.1 Description de la géométrie d'une coque mince 28
2.4.2 Éléments de géométrie différentielle 29
2.4.3 Le modèle linéaire de Koiter 31
2.4.4 Mise en œuvre éléments finis 32
2.4.5 Stratégie pour l'optimisation de formes 33
2.5 Synthèse du chapitre **33**

2.1 Introduction

Le processus standard de simulation en industrie peut se décomposer en quatre étapes. Au départ, une forme paramétrée doit être réalisée à l'aide d'un modeleur CAO. Ensuite, l'objet de l'étude doit être préparé en vue des calculs. Il faut alors « nettoyer » la CAO (si besoin) puis créer un maillage. À la suite de cela, les simulations sont exécutées au moyen des différents logiciels adaptés au cas d'étude. La dernière étape est le post-traitement des résultats. Comme l'illustre le diagramme de la figure 2.1,

pour mener à bien une simulation, il faut souvent arriver à faire interagir différents logiciels.

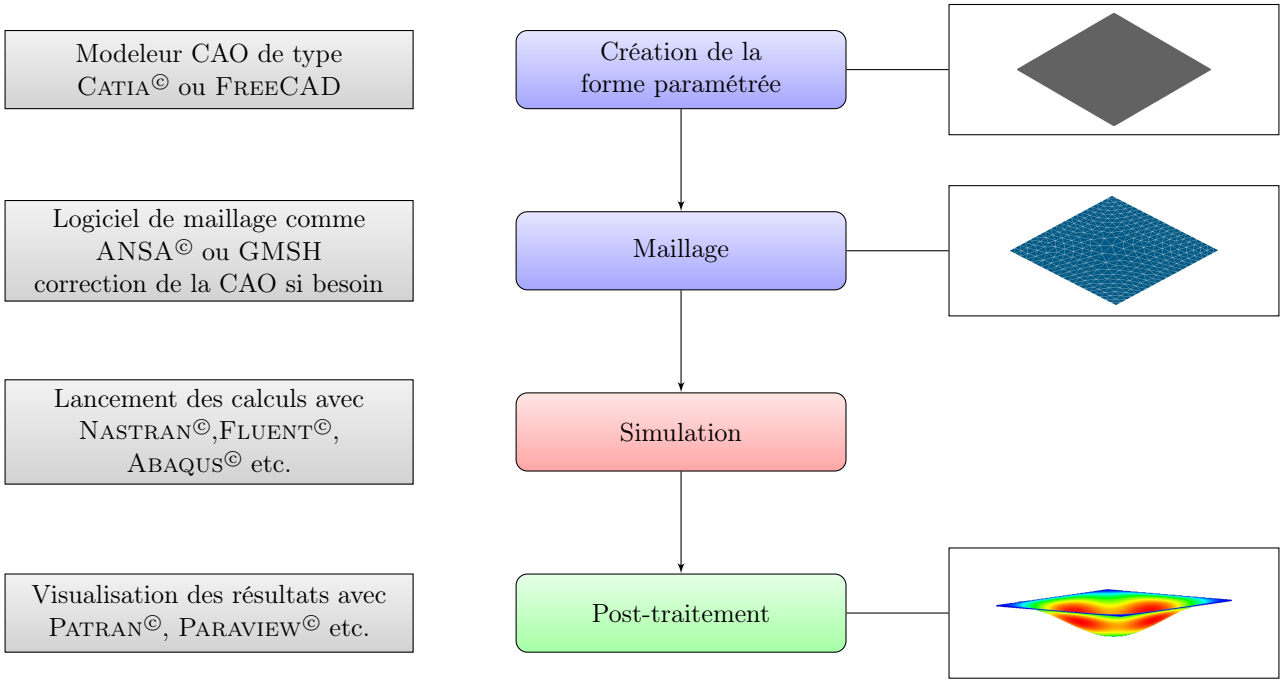


FIGURE 2.1: Déroulement d'une simulation

La mise en place de ce processus peut être complexe et chronophage. De plus, elle requiert pour chacune des phases une certaine expertise des procédés. Dans le but d'améliorer et de simplifier l'élaboration des simulations, Hughes propose la notion d'analyse isogéométrique dans [Hughes *et al.*, 2005]. L'analyse isogéométrique est une méthode qui vise à lier la géométrie (le modèle CAO) à la simulation par éléments finis. L'idée consiste à remplacer les fonctions de forme des éléments finis par les objets constituant un modèle CAO que nous appellerons patchs. Comme nous le verrons, la méthode qui sera utilisée dans ce manuscrit sera un peu différente car le lien avec le patch CAO ne se situe pas au même endroit. Elle appartient néanmoins aux méthodes dites isogéométriques. Avant de décrire davantage la méthode, exposons le concept du patch CAO.

2.2 Les patchs CAO

Les patchs CAO sont les outils élémentaires de la CAO. Il en existe plusieurs types, toujours sous forme de surfaces paramétriques. Nous en présentons les plus communs ici. Pour un exposé complet sur ces questions, nous renvoyons à [Piegl et Tiller, 1995].

2.2.1 Les patchs de Bézier

Afin de créer un outil pour simplifier la modélisation des formes pour le design des voitures, Pierre Bézier, ingénieur chez Renault développa les courbes de Bézier. Ces recherches énoncées dans [Bézier, 1977] s'appuient sur les travaux de Paul de Casteljau [de Casteljau, 1985] et ont permis de développer le code UNISURF© qui est la base de nombreux logiciel de CAO dont CATIA© de Dassault Systèmes.

Définition 2.2.1 (Courbe de Bézier)**.** Une courbe de Bézier $\mathscr{C}$ de degré n est définie par la courbe paramétrée suivante :

$$\mathscr{C}(\xi) = \sum_{i=0}^{n} B_{i,n}(\xi) P_i, \qquad 0 \leq \xi \leq 1, \tag{2.2.1}$$

où

- les fonctions de bases $(B_{i,n}(u))_{1\leq i\leq n}$ sont les polynômes de Bernstein de degré n donnés par :

$$B_{i,n}(\xi) = \frac{n!}{i!(n-i)!}\xi^i(1-\xi)^{n-i}; \tag{2.2.2}$$

- les $n+1$ coefficients P_i sont appelés les *points de contrôle* ou pôles.

Les points de contrôle sont les paramètres qui pilotent la courbe. Leur variation permet de modifier sa forme. Il faut noter qu'à part pour le premier et le dernier point de contrôle ($\mathscr{C}(0) = P_0$ et $\mathscr{C}(1) = P_n$), la courbe ne passe généralement pas par ses pôles. Elle est cependant contenue dans l'enveloppe convexe de ces derniers.

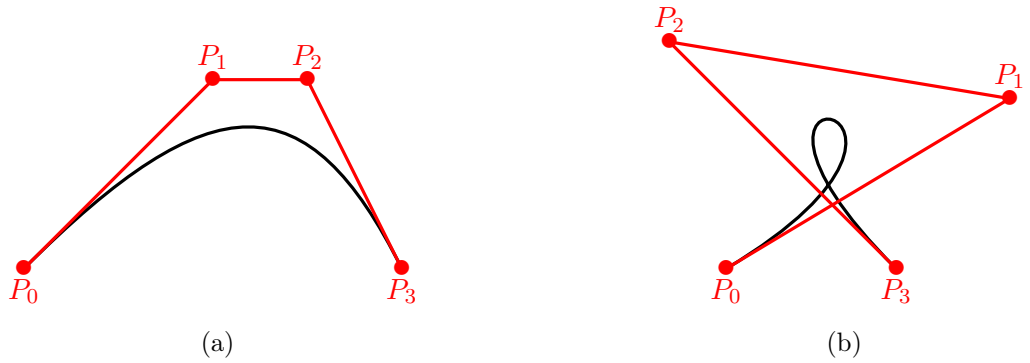


FIGURE 2.2: Exemple de deux courbes de Bézier de degré 3 pilotés par 4 points de contrôle

Les premiers patchs CAO à avoir été utilisés étaient les patchs de Bézier. Ceux-ci sont des surfaces paramétriques qui sont les résultats de produits tensoriels de deux courbes de Bézier.

Définition 2.2.2 (Surface de Bézier)**.** Une surface de Bézier $\mathscr{S}$ de degré (n, m) est définie par :

$$\mathscr{S}(\xi,\eta) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,n}(\xi)B_{j,m}(\eta)P_{ij}, \qquad 0 \leq \xi,\eta \leq 1. \tag{2.2.3}$$

Les surfaces de Bézier interpolent les quatre points de contrôle aux coins de la surface ($\mathscr{S}(0,0) = P_{0,0}$, $\mathscr{S}(0,1) = P_{0,m}$, $\mathscr{S}(1,0) = P_{n,0}$ et $\mathscr{S}(1,1) = P_{n,m}$) et sont également contenues dans l'enveloppe convexe de leurs points de contrôle (voir figure 2.3). Chaque arête de la surface est une courbe de Bézier.

Les courbes et surfaces de Bézier sont des fonctions polynomiales, ce qui présente plusieurs avantages vis à vis de la régularité des fonctions ou encore de l'implémentation dans des programmes informatiques. Néanmoins, cela limite le champ des formes possibles. Certaines courbes et surfaces telles que les cercles, les ellipses, les cylindres ou les sphères ne peuvent pas être parfaitement représentées par des polynômes et donc par des courbes ou surfaces de Bézier.

2.2.2 Les surfaces B-splines

Comme évoquée précédemment, la modélisation par des courbes et des surfaces de Bézier ne permet pas d'obtenir certaines formes. D'autre part, il faut souvent recourir à des polynômes d'ordre élevé pour satisfaire un nombre de contraintes conséquent. Le choix de courbes d'ordre élevé peut provoquer des instabilités numériques. À ceci s'ajoute le fait que le contrôle des courbes et surfaces de Bézier par les points de contrôle peut ne pas être suffisant d'un point de vue local. C'est entre autres pour

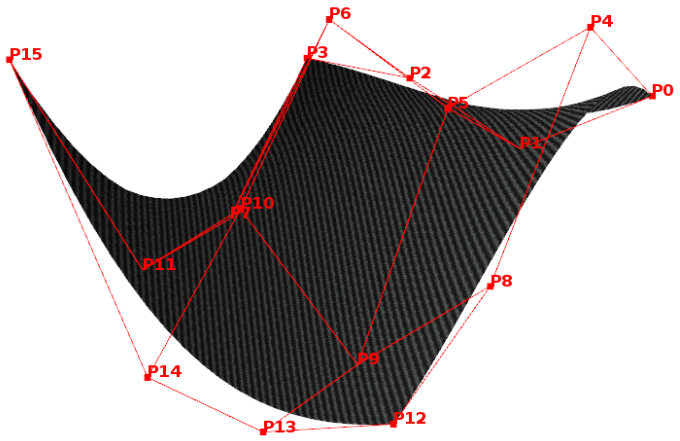


FIGURE 2.3: Surface de Bézier de degré 3 maîtrisée par 16 points de contrôle.

ces diverses raisons que les courbes et surfaces B-splines leurs sont préférées. De façon similaire aux courbes de Bézier, les courbes B-splines sont le résultat d'une combinaison de points de contrôle et de fonctions de base. Dans le cas présent, les fonctions de base ne sont pas des polynômes de Bernstein mais sont obtenues à l'aide de la formule de Cox-de Boor [de Boor, 1962].

Définition 2.2.3 (Fonctions de base B-splines)**.** Soit $\Xi = \{\xi_0, \ldots, \xi_m\}$ une séquence croissante de nombres réels. Les ξ_i sont appelés des *nœuds* (ou *knots* en anglais) et Ξ est le *vecteur de nœuds*.
La $i^{\text{ème}}$ fonction de base B-spline de degré p (ou d'ordre $p+1$) notée $N_{i,p}$ s'écrit :

$$\begin{cases} N_{i,0}(\xi) = \begin{cases} 1 & \text{si} \quad \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{sinon} \end{cases}, \\ N_{i,p}(\xi) = \dfrac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \dfrac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi). \end{cases} \tag{2.2.4}$$

Les fonctions de bases B-splines sont polynomiales. Aux nœuds ξ_i les fonctions sont $p-k$ continûment différentiables où k est la multiplicité du nœud, c'est-à-dire le nombre de fois où il est répété dans le vecteur de nœuds.

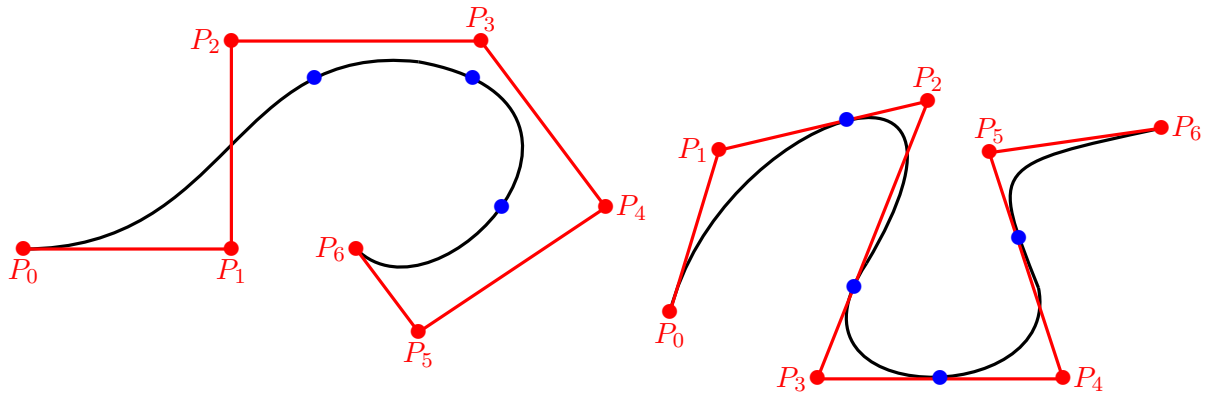
Définition 2.2.4 (courbe B-spline)**.** Une courbe B-spline $\mathscr{C}$ de degré p est définie par :

$$\mathscr{C}(\xi) = \sum_{i=0}^{n} N_{i,p}(\xi) P_i, \qquad \xi_0 = \alpha \leq \xi \leq \xi_m = \beta, \tag{2.2.5}$$

où

- les P_i sont les points de contrôle qui forment le polygone de contrôle ;
- les $N_{i,p}$ sont les fonctions de bases B-splines de la définition 2.2.3 définies sur le vecteur de nœuds non périodique et non uniforme $\Xi = \{\underbrace{\alpha, \ldots, \alpha}_{p+1}, \xi_{p+1}, \ldots, \xi_{m-p-1}, \underbrace{\beta, \ldots, \beta}_{p+1}\}$ avec $m = n+p+1$.

Remarque 2.2.1. On dit que la fonction B-spline est uniforme, lorsque le vecteur de nœuds est uniforme c'est-à-dire lorsque les nœuds sont espacés de façon régulière. Dans le cas contraire, on parle de vecteur de nœuds non uniforme. En particulier, si $n = p$ et $\Xi = \{0, \ldots, 0, 1, \ldots, 1\}$ alors la courbe $\mathscr{C}$ est une courbe de Bézier.



(a) (b)

FIGURE 2.4: Exemple de deux courbes B-splines avec 7 points de contrôle - les • indiquent les points de rupture du vecteur de nœuds (c'est-à-dire lorsque $\xi_{i+1} \neq \xi_i$).

Tout comme la courbe de Bézier, la courbe B-spline interpole ses extrémités (voir figure 2.4) et est contenue dans l'enveloppe convexe de son polygone de contrôle. À noter que la modification d'un point de contrôle P_i change la courbe $\mathscr{C}$ uniquement sur l'intervalle $[\xi_i, \xi_{i+p+1}]$. Une propriété essentielle des courbes B-splines est que le polygone de contrôle représente une approximation linéaire par morceaux de la courbe. Cette approximation peut être améliorée par l'ajout de nœuds dans le vecteur de nœuds ou bien en augmentant le degré de la courbe.

Une surface B-spline est obtenue en prenant un ensemble bidimensionnel de points de contrôle, deux vecteurs de nœuds et les produits invariants des fonctions de base B-splines.

Définition 2.2.5 (surface B-spline)**.** Une surface B-spline $\mathscr{S}$ a pour équation

$$\mathscr{S}(\xi,\eta) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(\xi)N_{j,q}(\eta)P_{i,j}, \quad \alpha \leq \xi \leq \beta, \quad \alpha' \leq \eta \leq \beta', \tag{2.2.6}$$

avec

- $P_{i,j}$ l'ensemble des points de contrôle ;
- $N_{i,p}$ et $N_{j,q}$ données par l'équation (2.2.4) définies respectivement avec les vecteurs :
  - $\Xi = \{\underbrace{\alpha,\ldots,\alpha}_{p+1}, \xi_{p+1},\ldots,\xi_{r-p-1}, \underbrace{\beta,\ldots,\beta}_{p+1}\}$ où $r = n+p+1$;
  - $H = \{\underbrace{\alpha',\ldots,\alpha'}_{q+1}, \eta_{q+1},\ldots,\eta_{s-q-1}, \underbrace{\beta',\ldots,\beta'}_{q+1}\}$ où $s = m+q+1$.

Remarque 2.2.2. Si $n = p$, $m = q$, $\Xi = \{0,\ldots,0,1,\ldots,1\}$ et $H = \{0,\ldots,0,1,\ldots,1\}$ alors la surface $\mathscr{S}$ est une surface de Bézier.

Les surfaces B-splines jouissent de propriétés similaires à celles des courbes B-spline : la surface interpole les quatre coins du polygone de contrôle, elle est contenue dans l'enveloppe convexe de ce dernier et l'approximation par le polygone peut être améliorée avec les mêmes méthodes.

La courbe ou la surface peut être modifiée par les points de contrôles ou par les nœuds. Le fait d'utiliser des polynômes par morceaux apporte davantage de contrôle au niveau local. Malgré ce meilleur contrôle, certaines formes restent inaccessibles. Pour pallier ce problème, il faut avoir recours à un troisième type de courbe et de surface plus général : les NURBS.

2.2.3 Les patchs NURBS

Les courbes et surfaces NURBS (*Non Uniform Rational B-Splines*) sont la généralisation des courbes et surfaces introduites précédemment. Elles sont déterminées à partir des fonctions de bases B-splines. La différence entre les B-splines classiques et les NURBS est que ces dernières sont définies de manière rationnelle.

Définition 2.2.6 (Courbe NURBS)**.** Une courbe NURBS $\mathscr{C}$ de degré p a pour équation

$$\mathscr{C}(\xi) = \frac{\displaystyle\sum_{i=0}^{n} N_{i,p}(\xi) w_i P_i}{\displaystyle\sum_{i=0}^{n} N_{i,p}(\xi) w_i}, \qquad \alpha \leq \xi \leq \beta \tag{2.2.7}$$

où

- les P_i sont les points de contrôle formant le polygone de contrôle ;
- les $N_{i,p}$ sont les fonctions de base B-splines définies avec le vecteur de nœuds : $\Xi = \{\underbrace{\alpha, \ldots, \alpha}_{p+1}, \xi_{p+1}, \ldots, \xi_{m-p-1}, \underbrace{\beta, \ldots, \beta}_{p+1}\}$ où $m = n + p + 1$.
- les w_i sont les poids associés aux points de contrôle.

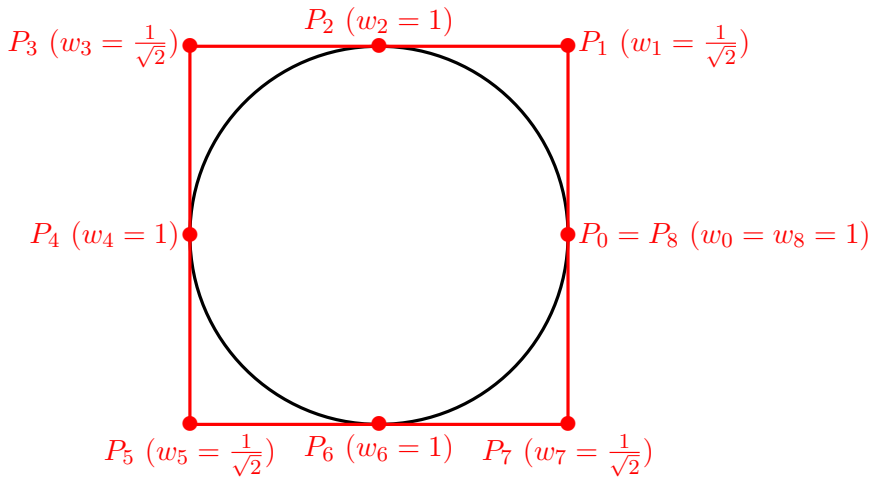


FIGURE 2.5: Modélisation d'un cercle parfait à l'aide une courbe NURBS pilotée par 9 points de contrôles et poids associés.

Contrairement aux surfaces de Bézier et B-splines, les surfaces NURBS ne sont pas obtenues par un produit tensoriel de deux courbes.

Définition 2.2.7 (Surface NURBS)**.** Une surface NURBS $\mathscr{S}$ s'exprime de la façon suivante :

$$\mathscr{S}(\xi, \eta) = \frac{\displaystyle\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(\xi) N_{j,q}(\eta) w_{i,j} P_{i,j}}{\displaystyle\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(\xi) N_{j,q}(\eta) w_{i,j}}, \quad \alpha \leq \xi \leq \beta, \quad \alpha' \leq \eta \leq \beta' \tag{2.2.8}$$

avec :

- $P_{i,j}$ l'ensemble des points de contrôle ;
- w_i les poids associés aux points de contrôle ;
- $N_{i,p}$ et $N_{j,q}$ données par l'équation (2.2.4) définie respectivement avec les vecteurs :
  - $\Xi = \{\underbrace{\alpha,\ldots,\alpha}_{p+1},\xi_{p+1},\ldots,\xi_{r-p-1},\underbrace{\beta,\ldots,\beta}_{p+1}\}$ où $r = n+p+1$;
  - $H = \{\underbrace{\alpha',\ldots,\alpha'}_{q+1},\eta_{q+1},\ldots,\eta_{s-q-1},\underbrace{\beta',\ldots,\beta'}_{q+1}\}$ où $s = m+q+1$.

Les surfaces NURBS disposent des mêmes propriétés que les surfaces B-splines c'est-à-dire qu'elles sont incluses dans les enveloppes convexes de leur polygone de contrôle et sont invariantes par transformation affine. De plus, une modification locale des points de contrôle implique une modification locale de la surface et les surfaces passent par les quatre points de contrôle aux extrémités. Il est également possible de représenter la plupart des objets coniques tels que les cercles ou les sphères de façon parfaite (voir figures 2.5 et 2.6.)

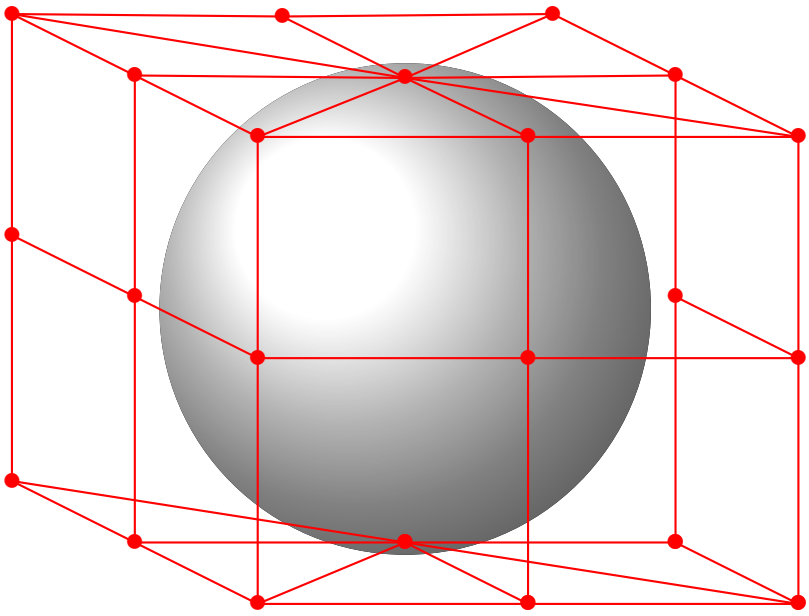


FIGURE 2.6: Répresentation d'une sphère exacte à l'aide d'une surface NURBS

Remarque 2.2.3. L'une des propriétés des fonctions de base B-splines (et également des polynômes de Bernstein) est la partition de l'unité, en d'autres termes :

$$\sum_{i=1}^{n} N_{i,p}(\xi) = 1. \tag{2.2.9}$$

Ainsi, si tous les poids sont égaux à 1, alors la surface (ou la courbe) NURBS devient une surface (ou une courbe) B-spline voire de Bézier.

Les patchs de Bézier, B-splines et NURBS ne sont pas les seuls patchs CAO existants. Un autre type de patch très répandu, sur lequel nous reviendrons dans le chapitre 3, est le patch trimmé. Nous nous sommes restreints à l'introduction des Bézier, B-splines et NURBS car ils sont les plus adaptés pour l'analyse isogéométrique dont il est maintenant question.

2.3 Le concept de l'analyse isogéométrique

2.3.1 Définition de l'analyse géométrique

L'analyse isogéométrique, introduite par Hughes en 2005 et largement développée dans [Cottrell *et al.*, 2009], regroupe l'ensemble de méthodes numériques pour lesquelles les calculs sont faits sur la géométrie exacte en exploitant les patchs CAO. L'idée principale de l'analyse isogéométrique est d'utiliser les mêmes éléments pour modéliser à la fois la géométrie et l'espace des solutions des méthodes numériques. L'analyse isogéométrique a un concept semblable à celui des éléments finis isoparamétriques. Avec les éléments finis isoparamétriques, les fonctions de bases choisies pour l'approximation de la solution inconnue servent ensuite à déterminer une approximation de la géométrie qui elle est connue. L'analyse isogéométrique fonctionne cependant de façon différente : les fonctions de base pour l'approximation de la solution sont choisies de sorte qu'elles représentent exactement la géométrie. En pratique, ce sont les NURBS qui sont préférées. Pour mener à bien une analyse isogéométrique, une discrétisation du patch est nécessaire et la résolution du problème étudié (élasticité, fluide etc.) est effectuée au moyen de méthodes numériques *ad hoc*.

2.3.2 Maillage des NURBS

Pour l'analyse isogéométrique, un maillage des surfaces NURBS est réalisé. Ce maillage est basé sur la partition de l'espace des paramètres (voir figure 2.7). Les frontières des éléments dans l'espace physique sont les images des lignes de nœuds par les fonctions de base B-splines.
Il existe plusieurs façons de raffiner ce maillage si besoin :

- l'insertion de nœuds, qui s'apparente au h-raffinement des éléments finis, consiste à ajouter un nœud au sein du vecteur de nœuds partitionnant davantage le patch ;
- l'élévation du degré qui ressemble au $p-$raffinement classique où l'on augmente le degré des fonctions de base B-splines ;
- l'augmentation du degré et de la continuité aussi appelé k-raffinement où des nœuds sont ajoutés et le degré des fonctions de base est également augmenté.

Ces différents types de raffinement laissent dans tous les cas la géométrie inchangée. Le vecteur de nœuds des NURBS contrôle à la fois le nombre d'éléments utilisé pour l'analyse isogéométrique ainsi que la continuité entre les différents éléments.

2.3.3 Méthodes numériques en analyse isogéoémtrique

Pour résoudre les problèmes étudiés à l'aide de l'analyse isogéométrique, deux types de méthodes numériques sont généralement utilisés :

- la méthode de Galerkin ;
- la méthode des collocations.

Nous illustrons ces deux méthodes à l'aide d'un exemple simple. Considérons l'équation de Laplace sur un domaine défini par une surface NURBS. L'objectif est de trouver la solution $u \ : \Omega \to \mathbb{R}$ telle que :

$$\begin{cases} \Delta u + f & = 0 \quad \text{dans } \Omega \\ u & = 0 \quad \text{sur } \partial\Omega \end{cases} \tag{2.3.1}$$

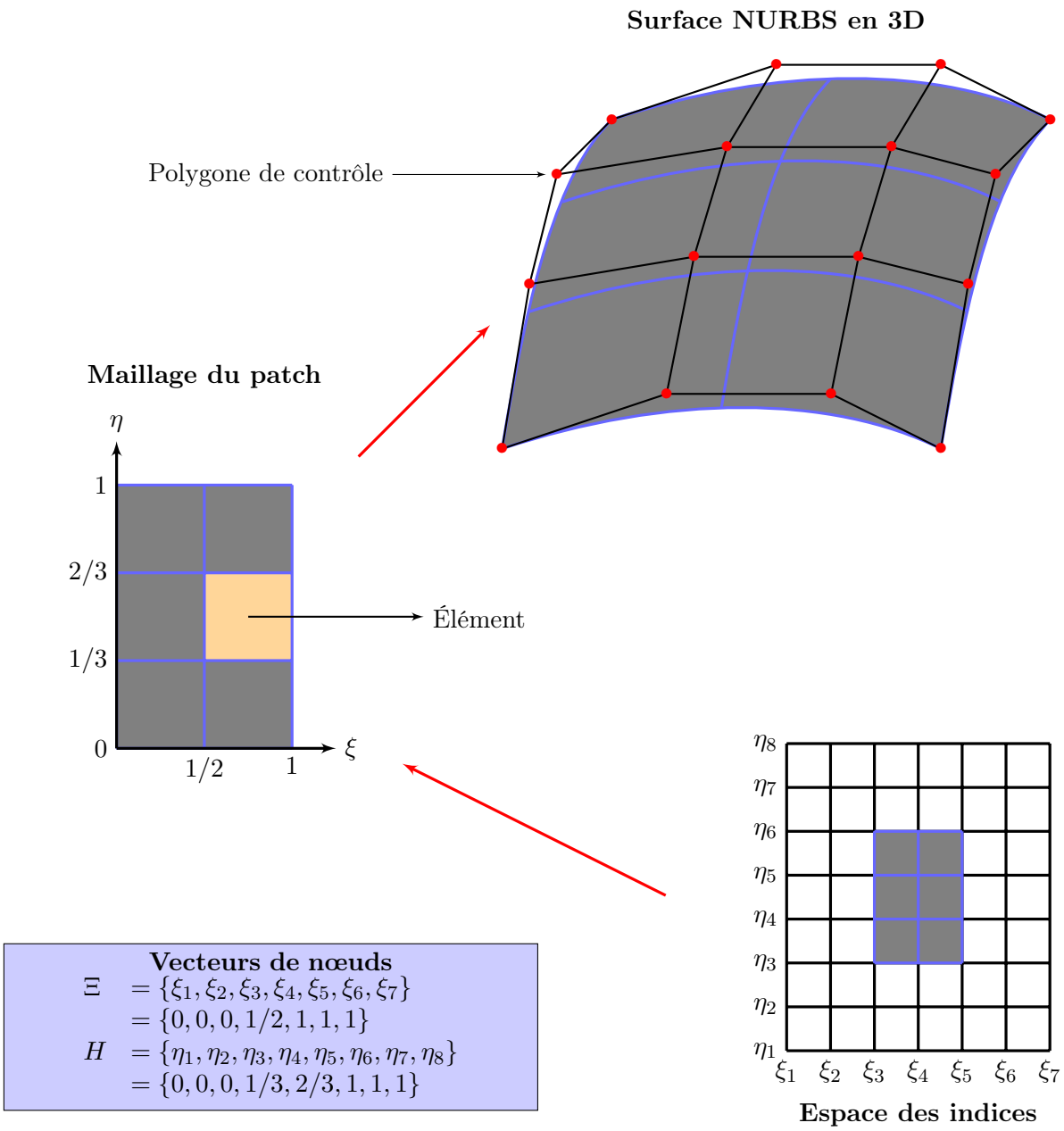


FIGURE 2.7: Exemple de maillage d'une surface NURBS à partir de ses vecteurs de nœuds.

2.3.3.1 La méthode de Galerkin

La méthode de Galerkin en analyse isogéometrique est la même que celle pour les éléments finis. Elle est basée sur la formulation faible du problème (2.3.1) :

$$\forall v \in \mathcal{V} \quad \int_\Omega \nabla u \cdot \nabla v \, d\Omega = \int_\Omega f v \, d\Omega \tag{2.3.2}$$

avec

$$\mathcal{V} = \{v \mid v \in H^1(\Omega), v|_{\partial\Omega} = 0\} \tag{2.3.3}$$

où $H^1(\Omega)$ est l'espace de Sobolev défini par $H^1(\Omega) = \{u \in L^2(\Omega) \mid \nabla u \in L^2(\Omega)\}$. La méthode de Galerkin consiste à construire une approximation $\mathcal{V}^h$ de dimension finie de l'espace $\mathcal{V}$. On cherche alors u^h solution de :

$$\forall v^h \in \mathcal{V}^h, \quad \int_\Omega \nabla u^h \cdot \nabla v^h \, d\Omega = \int_\Omega f v^h \, d\Omega \tag{2.3.4}$$

L'espace des solutions est composé de toutes les combinaisons linéaires d'un ensemble donné de fonctions NURBS $N_i : \Omega \to \mathbb{R}$ avec $i = 1, \ldots, q$. Ainsi, $\forall v^h \in \mathcal{V}^h$, il existe des constantes v_i avec $i = 1, \ldots, q$ telles que :

$$v^h = \sum_{i=1}^{q} N_i v_i. \tag{2.3.5}$$

L'assemblage des matrices et vecteurs, semblable aux éléments finis, est détaillé dans [Cottrell *et al.*, 2009]. Plusieurs études, dont celles de [Bazilevs *et al.*, 2006a] et [Hughes *et al.*, 2010], montrent que les résultats de convergence de cette méthode sont aussi bons voire meilleurs que ceux de la méthode traditionnelle des éléments finis.

2.3.3.2 La méthode de collocation

La seconde méthode, de plus en plus répandue, est la méthode de collocation [Auricchio *et al.*, 2010]. Souvent utilisée dans la méthode des éléments finis de frontière, elle est plus simple à mettre en œuvre que la méthode de Galerkin. Le but de cette méthode est de générer une solution vérifiant l'équation aux dérivées partielles (2.3.1) en un certain nombre de points appelés points de collocation.

Soient $\zeta_i \in \Omega$ $(i = 1, \ldots, q)$ l'ensemble des points de collocation. Le problème à résoudre est :

$$\text{trouver } u^h = \sum_{i=1}^{q} N_i u_i \quad \text{tel que} \quad \Delta u^h(\zeta_i) = f(\zeta_i) \tag{2.3.6}$$

L'existence d'une solution à un tel système dépend directement du choix des points de collocation. Les points de collocation doivent cependant être sélectionnés avec soin car la méthode peut se révéler instable.

2.3.4 Bilan sur l'analyse isogéométrique

L'analyse isogéométrique propose une nouvelle approche de la simulation. Avec des calculs réalisés sur la géométrie exacte, les résultats sont tout aussi satisfaisants qu'avec des éléments finis mais avec une mise en oeuvre plus simple et une meilleure intégration dans le processus de conception. Plusieurs logiciels sont déjà disponibles pour l'analyse isogéométrique : le logiciel Rhino 5© permet de créer des modèles CAO composés exclusivement de NURBS et des codes de calcul comme GeoPDEs ou PetIGA assurent l'analyse isogéométrique.

L'analyse isogéométrique a fait ses preuves dans de nombreux domaines. Des résultats satisfaisants ont été obtenus aussi bien pour des problèmes en élasticité, en mécanique des fluides [Bazilevs *et al.*, 2006b] ou encore liés à la dynamique [Cottrell *et al.*, 2006]. L'un des principaux enjeux en analyse isogéométrique est l'étude des coques minces, connue pour être complexe [Kiendl *et al.*, 2009]. Nous rappelons dans la partie suivante la définition d'un problème de coques et comment le résoudre. Nous introduirons également la méthodologie retenue pour faire l'optimisation de formes associée.

2.4 Les équations de coques

2.4.1 Description de la géométrie d'une coque mince

Soit $\mathcal{E}^3$ l'espace euclidien classique muni du repère orthonormé fixe $(0, \vec{e}_1, \vec{e}_2, \vec{e}_3)$ et soit Ω un sous-ensemble ouvert borné du plan $\mathcal{E}^2$. Une coque $\mathcal{C}$ est définie par sa surface moyenne S qui est l'image d'une application $\vec{\varphi}$ bidimensionnelle du domaine Ω (figure 2.8) telle que :

$$\vec{\varphi} : (\xi_1, \xi_2) \in \bar{\Omega} \subset \mathcal{E}^2 \to \vec{\varphi}(\xi_1, \xi_2) \in S \subset \mathcal{E}^3. \tag{2.4.1}$$

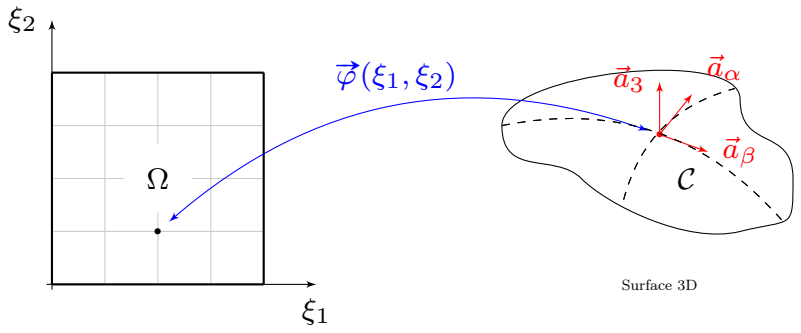
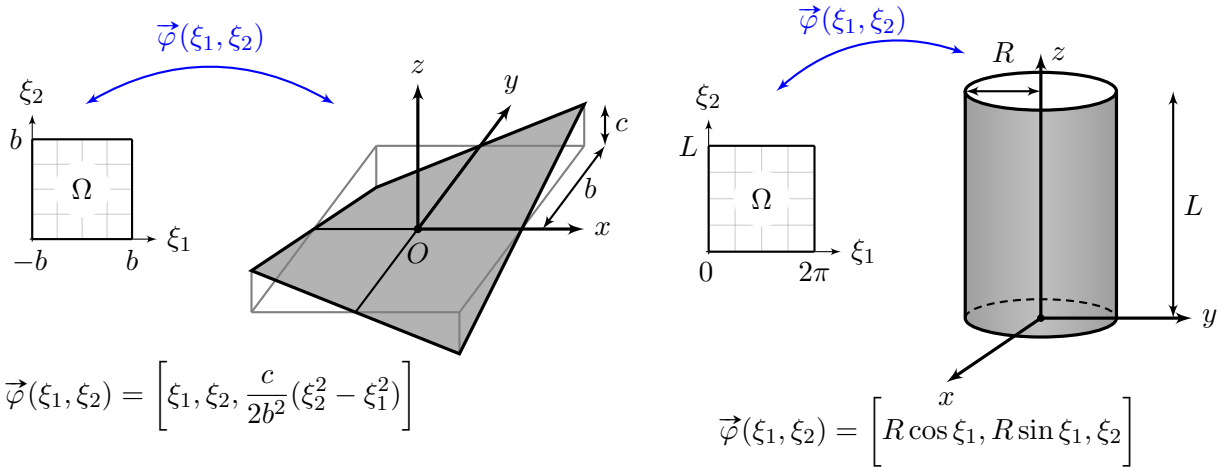


FIGURE 2.8: Géométrie d'une coque mince et fonction de forme

Dans le cadre des problèmes de coques, les équations ne sont pas exprimées sur la surface en dimension 3 mais sont ramenées sur le domaine de référence Ω. La fonction $\vec{\varphi}$ appelée, fonction de forme, contient le paramétrage permettant de faire le lien entre le domaine de référence et la coque en dimension 3. Les équations sont écrites sur le domaine de référence à l'aide de plusieurs entités de géométrie différentielle estimées à partir des coordonnées curvilignes (ξ_1, ξ_2) (voir les exemples des figures 2.9a et 2.9b).



(a) Un paraboloïde hyperbolique

(b) Un cylindre

FIGURE 2.9: Deux exemples de paramétrage de la fonction de forme.

La coque $\mathcal{C}$ d'épaisseur e est donc le sous-ensemble fermé de $\mathcal{E}^3$ défini par :

$$\mathcal{C} = \left\{ M \in \mathcal{E}^3; \quad \overrightarrow{OM} = \vec{\varphi}(\xi_1, \xi_2) + \xi_3 \vec{a}_3, \quad (\xi_1, \xi_2) \in \bar{\Omega}, \quad -\frac{1}{2}e \leq \xi_3 \leq \frac{1}{2}e \right\}. \tag{2.4.2}$$

2.4.2 Éléments de géométrie différentielle

Pour être en mesure de transférer les phénomènes appliqués à la coque en dimension 3 sur le domaine de référence Ω, des outils issus de la géométrie différentielle sont nécessaires. Nous listons ici ceux principalement utilisés pour établir les modèles de coque. Les lettres grecques en indice prennent leurs valeurs dans $\{1, 2\}$, les lettres romaines dans $\{1, 2, 3\}$ et nous utilisons la sommation d'Einstein.

La base covariante $(\vec{a}_i)$

À chaque point $\vec{\varphi}(\xi)$ est attaché un repère local $(\vec{a}_1, \vec{a}_2, \vec{a}_3)$ appelé base covariante. Cette base est composée de deux vecteurs définissant le plan tangent à la surface moyenne en ce point :

$$\vec{a}_\alpha = \vec{\varphi}_{,\alpha} = \frac{\partial \vec{\varphi}}{\partial \xi^\alpha}, \quad \alpha = 1, 2,$$

et d'un vecteur normal au plan tangent :

$$\vec{a}_3 = \frac{\vec{a_1} \wedge \vec{a_2}}{||\vec{a_1} \wedge \vec{a_2}||}, \qquad \text{avec } ||.|| \text{ la norme euclidienne dans } \mathcal{E}^3.$$

Précisons que pour alléger les notations, on note $\vec{a}_\alpha$ au lieu de $\vec{a}_\alpha(\xi)$.

La base contravariante ($\vec{a}^i$)

Aux vecteurs tangents $\vec{a}_\alpha$ de la base covariante sont associés les vecteurs $\vec{a}^\beta$ de la base contravariante tels que :

$$\vec{a}_\alpha \cdot \vec{a}^\beta = \delta_\alpha^\beta = \begin{cases} 1 & \text{si } \alpha = \beta \\ 0 & \text{si } \alpha \neq \beta \end{cases}$$

et

$$\vec{a}_3 = \vec{a}^3$$

Les première et seconde formes fondamentales ($a_{\alpha\beta}, b_{\alpha\beta}$)

Les première et seconde formes fondamentales s'écrivent respectivement :

$$a_{\alpha\beta} = a_{\beta\alpha} = \vec{a}_\alpha \cdot \vec{a}_\beta$$
$$b_{\alpha\beta} = b_{\beta\alpha} = -\vec{a}_\alpha \cdot \vec{a}_{3,\beta} = \vec{a}_3 \cdot \vec{a}_{\alpha,\beta} = \vec{a}_3 \cdot \vec{a}_{\beta,\alpha}$$

Relations entre les différents tenseurs

Le tenseur ($a^{\alpha\beta}$), l'inverse du tenseur métrique ($a_{\alpha\beta}$), est donné par :

$$a^{\alpha\beta} = \vec{a}^\alpha \cdot \vec{a}^\beta.$$

Les composantes mixtes (b_α^β) sont définies par :

$$b_\alpha^\beta = a^{\beta\lambda} b_{\lambda\alpha}.$$

Il existe de nombreuses relations reliant les bases covariantes, contravariantes et les différents tenseurs. Elles sont détaillées notamment dans [Ciarlet, 2005].

Les symboles de Christoffel ($\Gamma_{\beta\gamma}^\alpha$)

Pour pouvoir calculer les dérivées des vecteurs de base, les symboles de Christoffel $\Gamma_{\beta\gamma}^\alpha$ sont nécessaires :

$$\Gamma_{\beta\gamma}^\alpha = \Gamma_{\gamma\beta}^\alpha = \vec{a}^\alpha \cdot \vec{a}_{\gamma,\beta} = \vec{a}^\alpha \cdot \vec{a}_{\beta,\gamma}.$$

Ainsi :

$$\vec{a}_{\alpha,\beta} = \Gamma_{\alpha\beta}^\gamma \vec{a}_\gamma + b_{\alpha\beta}\vec{a}_3; \quad \vec{a}_{,\beta}^\alpha = -\Gamma_{\beta\lambda}^\alpha \vec{a}^\lambda + b_\beta^\alpha \vec{a}_3; \quad \vec{a}_{3,\alpha} = \vec{a}_{,\alpha}^3 = -b_\alpha^\gamma \vec{a}_\gamma$$

Les symboles de Christoffel sont utilisés pour calculer la dérivée covariante $(.)_{\alpha|\gamma}$ pour les tenseurs de surface T :

$$T_{\alpha|\gamma} = T_{\alpha,\gamma} - \Gamma_{\alpha\gamma}^\lambda T^\lambda; \quad T^\alpha|_\gamma = T_{,\gamma}^\alpha + \Gamma_{\lambda\gamma}^\lambda T^\lambda$$

$$\begin{cases} T_{\alpha\beta|\gamma} = & T_{\alpha\beta,\gamma} - \Gamma_{\alpha\gamma}^\lambda T_{\lambda\beta} - \Gamma_{\beta\gamma}^\lambda T_{\alpha\lambda}, \\ T_{.\beta|\gamma}^\alpha = & T_{.\beta,\gamma}^\alpha + \Gamma_{\gamma\lambda}^\alpha T_{.\beta}^\lambda - \Gamma_{\beta\gamma}^\lambda T_{.\lambda}^\alpha, \\ T^{\alpha\beta}|_\gamma = & T_{,\gamma}^{\alpha\beta} + \Gamma_{\lambda\gamma}^\alpha T^{\lambda\beta} + \Gamma_{\lambda\gamma}^\beta T^{\alpha\lambda}. \end{cases}$$

2.4.3 Le modèle linéaire de Koiter

Il existe plusieurs modèles de coques tels que les modèles de Koiter, Naghdi ou Maguerre-von Karman [Bischoff *et al.*, 2004]. Le modèle de Naghdi est, par exemple, réservé aux coques épaisses et celui de von Karman aux coques peu profondes. Dans le secteur automobile, les structures surfaciques sont modélisées par des coques minces profondes autorisant de forte variation de courbure, ce qui permet de conserver la validité du modèle au cours d'une optimisation de formes. Le modèle de Koiter est donc celui qui a été retenu. Pour respecter au mieux le comportement d'une coque mince, le modèle, introduit par W. T Koiter [Koiter, 1966], vérifie les conditions de Kirchhoff Love qui sont les suivantes :

- les contraintes restent planes et parallèles au plan tangent à la surface moyenne pendant la déformation ;
- la normale à la surface moyenne non déformée reste normale à la surface moyenne déformée.

La seconde hypothèse implique que les effets du cisaillement sont négligés. La coque sera donc uniquement sollicitée en flexion et en membrane. Si l'on considère une coque $\mathcal{C}$ de surface moyenne S image du domaine de référence Ω par la fonction de forme $\vec{\varphi}$, encastrée sur une partie $\partial\mathcal{S}_0$ de sa frontière $\partial\mathcal{S}$ et chargée sur la partie complémentaire de sa frontière $\partial\mathcal{S}_1$, alors la formulation variationnelle du modèle de Koiter est la suivante :

$$\forall \vec{v} \in \vec{\mathcal{V}} \quad \text{trouver } \vec{u} \in \vec{V} \text{ tel que } a(\vec{u}, \vec{v}) = f(\vec{v}) \tag{2.4.3}$$

avec :

$$\vec{\mathcal{V}} = \left\{ \vec{v} = (v_\alpha, v_3) \in ((H^1(\Omega))^2 \times H^2(\Omega));\ \vec{v}|_{\partial\mathcal{S}_0} = \vec{0}, \quad \frac{\partial v_3}{\partial n}\Big|_{\partial\mathcal{S}_0} = 0 \right\} \tag{2.4.4}$$

$$a(\vec{u}, \vec{v}) = \int_\Omega e E^{\alpha\beta\lambda\mu} \left[\gamma_{\alpha\beta}(\vec{u})\gamma_{\lambda\mu}(\vec{v}) + \frac{e^2}{12}\varrho_{\alpha\beta}(\vec{u})\varrho_{\lambda\mu}(\vec{v}) \right] \sqrt{a} d\xi_1 d\xi_2 \tag{2.4.5}$$

où :

- $\vec{u} = (u_1, u_2, u_3)$ est le déplacement ;
- e est l'épaisseur de la coque ;
- $E^{\alpha\beta\lambda\mu}$ est le tenseur d'élasticité ;
- $\gamma_{\alpha\beta}(\vec{u})$ est le tenseur de déformation de la surface moyenne ;
- $\varrho_{\alpha\beta}(\vec{u})$ est le tenseur de changement de courbure de la surface moyenne.
- $\sqrt{a}$ est l'élément d'aire.

Les différents termes dans l'équation (2.4.5) s'écrivent :

- $E^{\alpha\beta\lambda\mu} = \dfrac{E}{2(1+\nu)}\left[a^{\lambda\mu}a^{\beta\mu} + a^{\alpha\mu}a^{\beta\lambda} + \dfrac{2\nu}{1-\nu}a^{\alpha\beta}a^{\lambda\mu}\right]$ où E est le module de Young et ν le coefficient de poisson ;
- $\gamma_{\alpha\beta}(\vec{u}) = \dfrac{1}{2}\Big(u_{\alpha|\beta} + u_{\beta|\alpha}\Big) - b_{\alpha\beta}u_3$;
- $\varrho_{\alpha\beta}(\vec{u}) = -(u_{3|\alpha\beta} - b^\lambda_\alpha b_{\lambda\beta}u_3 + b^\lambda_{\alpha|\beta}; u_\lambda + b^\lambda_\alpha u_{\lambda|\beta} + b^\lambda_\beta u_{\lambda|\alpha})$;
- $\sqrt{a} = ||\vec{a_1} \wedge \vec{a_2}||$.

tandis que :

$$f(\vec{v}) = \int_\Omega \vec{p}\,\vec{v}\sqrt{a}d\xi_1 d\xi_2 + \int_{\partial\mathcal{S}_1} \vec{N}\vec{v} + \vec{M}\vec{\psi}(\vec{v})ds \tag{2.4.6}$$

où :

- $\vec{p}$ est la résultante sur $\mathcal{S}$ des charges extérieures ;
- $\vec{N}$ est la résultante sur $\partial\mathcal{S}$ des charges extérieures ;
- $\vec{M}$ est le moment résultant sur $\partial\mathcal{S}$ des charges appliquées au bord latéral de la coque ;
- $\vec{\psi}(\vec{v})$ est le vecteur de rotation infinitésimale.

La formulation variationnelle du modèle de Koiter est en fait l'écriture de la formulation variationnelle de l'élasticité linéaire en coordonnées curvilignes tout en respectant les hypothèses de Kirchhoff-Love. Dans [Ciarlet, 2005], la marche à suivre est détaillée pour déterminer ces équations. Afin de pouvoir satisfaire les hypothèses de Kirchhoff-Love, le modèle de Koiter fait intervenir des opérateurs d'ordre élevé et nécessite le recours à des éléments finis de type $\mathscr{C}^1$. Ces éléments étant coûteux, les modèles de coques épaisses respectant les hypothèses de Reissner-Midlin (prise en compte du cisaillement), comme le modèle de Naghdi, sont généralement préférés. Leur implémentation requiert seulement des éléments finis de type $\mathscr{C}^0$. Cependant, ce type de modèle est sujet à des phénomènes de verrouillage numérique, même en faisant appel à l'analyse isogéométrique [Adam *et al.*, 2015]. Dans notre cas, les éléments finis de Argyris, des éléments $\mathscr{C}^1$, ont été implémentés pour résoudre le problème (2.4.3).

2.4.4 Mise en œuvre éléments finis

La résolution du problème de coque pour un modèle de Koiter nécessite des éléments finis $\mathscr{C}^1$. Dans [Bernadou, 1994], il est montré que les éléments de Argyris sont tout à fait adaptés. Ils sont basés sur des polynômes d'interpolation de degré 5 et chaque triangle compte 21 degrés de liberté (figure 2.10) :

- la valeur du polynôme à chaque sommet ;
- la valeur des premières dérivées à chaque sommet ;
- la valeur des dérivées secondes à chaque sommet ;
- La valeur de la dérivée normale au milieu de chaque arête .

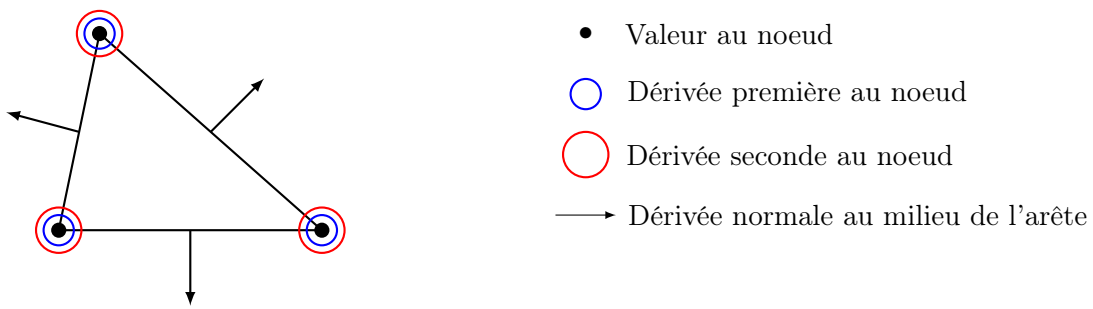


FIGURE 2.10: Dégrés de liberté d'un triangle d'Argyris.

L'approximation du déplacement $\vec{u} = (u_1, u_2, u_3)$ est implémentée à l'aide de trois triangles d'Argyris (un triangle par composante). On compte donc 63 degrés de liberté par élément. L'une des particularité de l'élément d'Argyris est que la définition des degrés de liberté est locale à chaque élément. Un soin particulier doit donc être accordé à l'assemblage. L'implémentation complète de ces éléments pour la résolution du problème de Koiter est entièrement détaillée dans [Bernadou, 1994] et rappelée dans [de Nazelle, 2013]. L'avantage d'utiliser des éléments aussi « riches » est que la convergence est assurée avec moins d'éléments qu'avec les méthodes utilisant des éléments finis classiques.

2.4.5 Stratégie pour l'optimisation de formes

Rappelons que l'objectif final de ce travail est l'optimisation de formes de structures minces. Comme nous l'avons vu dans le chapitre 1, l'une des questions cruciales en optimisation de formes est le choix du paramétrage de la forme. Ici, nous reprendrons l'approche proposée dans [de Nazelle, 2013] qui consistait à paramétrer la forme en utilisant les patchs CAO. Dans ce travail, il s'agissait de résoudre les équations de la mécanique à l'aide du modèle de Koiter pour lequel la fonction de forme $\vec{\varphi}$ de la coque, définie à l'équation (2.4.1), correspondait à un patch de Bézier de degré 3. D'après la définition 2.2.2, sous forme matricielle, la fonction de forme $\vec{\varphi}$ a pour expression :

$$\begin{cases} \vec{\varphi} : & [0,1]^2 \longrightarrow & \mathbb{R}^3 \\ & (\xi_1, \xi_2) \longmapsto & \begin{cases} T_{\xi_1} \cdot B \cdot P_X \cdot B^T \cdot T_{\xi_2}^T \\ T_{\xi_1} \cdot B \cdot P_Y \cdot B^T \cdot T_{\xi_2}^T \\ T_{\xi_1} \cdot B \cdot P_Z \cdot B^T \cdot T_{\xi_2}^T \end{cases} \end{cases} \tag{2.4.7}$$

où

- T_t est le vecteur donné par $T_t = [t^3, t^2, t, 1]$;
- P_X, P_Y, P_Z des matrices de taille 4×4 contenant respectivement les coordonnées en x, y et z des 16 points de contrôle du patch ;
- B est la matrice donnée par les polynômes de Bernstein : $B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & 6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$

Les variables d'optimisation sont ici les coordonnées des points de contrôle. Ces derniers apparaissent de façon explicite dans le problème de coque et permettent ainsi de déterminer un gradient par rapport à la forme. Un autre intérêt à choisir les points de contrôle comme variables d'optimisation est qu'ils constituent un petit ensemble de paramètres pouvant engendrer de fortes variations de la forme. Cette méthodologie se situe entre l'optimisation paramétrique et géométrique. En effet, la variable d'optimisation est bien la frontière de la forme mais celle-ci est contrôlée par des paramètres intrinsèques.

2.5 Synthèse du chapitre

Dans ce chapitre, nous avons présenté le concept de l'analyse isogéométrique. Cette approche regroupe les méthodes qui utilisent pour leurs simulations la géométrie exacte de la structure étudiée. Les fonctions de forme des éléments finis classiques peuvent alors être remplacées par des fonctions de forme issues de la CAO (patchs de Bézier ou NURBS). Nous avons également présenté la stratégie d'optimisation que nous utiliserons. Elle est basée sur la formulation du problème de coque de Koiter. La fonction de forme liant un domaine en dimension deux à la coque en trois dimensions est paramétrée par des patchs de Bézier ou des NURBS et les points de contrôle de ces derniers sont les variables d'optimisation. Dans [de Nazelle, 2013], des résultats d'optimisation pour des structures modélisées avec un seul patch de Bézier et pour un critère de première fréquence propre ont prouvé l'intérêt de la méthode. Dans le chapitre suivant, nous verrons dans un premier temps comment étendre le paramétrage à des modélisations plus complexes c'est-à-dire comportant plusieurs patchs CAO. Nous traiterons aussi de l'implémentation nécessaire à la fois pour ce raccordement entre patchs et la prise en compte d'autres type de patchs comme les NURBS. Les chapitres ultérieurs aborderont l'optimisation de la forme ainsi paramétrée pour différents critères de dimensionnement.

CHAPITRE

3 Mise en place des simulations numériques de coques minces

Sommaire

3.1 Introduction **36**

3.2 Le raccordement entre patchs **36**

3.2.1 Le raccordement entre patchs CAO en isogéométrie 36

3.2.2 Mise en œuvre pratique 39

3.2.2.1 Mise en équations du problème de jonction de coques 39

3.2.2.2 Comportement des charnières 42

3.2.2.3 Formulations variationnelles des problèmes de jonction 43

3.2.2.4 Choix du type de charnière 45

3.3 Méthode de résolution numérique **45**

3.3.1 Mise en œuvre de la méthode des éléments finis 45

3.3.1.1 Écriture du problème sous forme matricielle 45

3.3.1.2 Discrétisation des conditions de charnière 47

3.4 Présentation du code de calcul OptSurf **49**

3.4.1 Étape 1 : lecture du modèle CAO 50

3.4.2 Étape 2 : maillage des domaines de référence 50

3.4.3 Étape 3 : assemblage des matrices 50

3.4.4 Étape 4 : résolution du problème de coques 51

3.4.5 Étape 5 : le post-traitement 55

3.4.6 Bilan sur le code OptSurf 55

3.5 Résultats de quelques benchmarks **55**

3.5.1 Identification des modes de corps rigide 56

3.5.1.1 Le cas de la plaque carrée 56

3.5.1.2 Deux plaques perpendiculaires 57

3.5.1.3 Cylindre 58

3.5.2 Cas test classiques de déformation de coques 58

3.5.2.1 Plaque carrée soumise à une pression uniforme 58

3.5.2.2 Le paraboloïde hyperbolique sous pression 59

3.5.2.3 La toiture de Scordelis-Lo 60
3.5.3 Conclusion sur les résultats des benchmarks 60
3.6 Extension de la méthode à des configurations industrielles 61
3.7 Synthèse du chapitre 63

3.1 Introduction

Dans le chapitre 2, nous avons introduit notre choix de paramétrage des formes pour l'optimisation. Il consiste à utiliser la définition des patchs CAO composant un modèle comme fonction de forme du problème de coque. Dans le but de mener des études sur des cas industriels, la méthode que nous proposons doit être adaptée à des modèles CAO riches comme celui de la figure 3.1. La majorité des géométries à étudier sont constituées de plusieurs patchs CAO afin de décrire les pièces avec le plus de détail possible. Pour mener une analyse isogéometrique de qualité, il faut alors une description précise de chacun de ces patchs mais également de leurs jonctions.

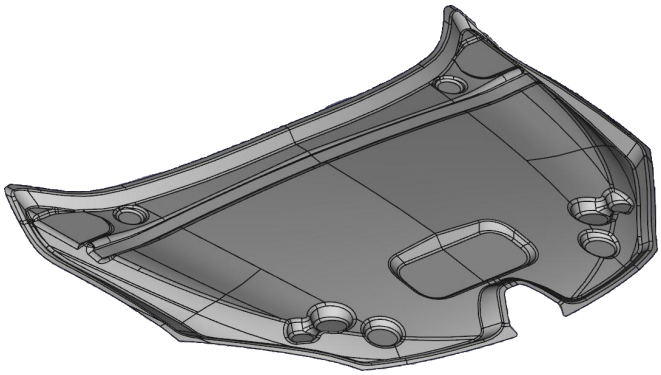


FIGURE 3.1: Modèle CAO d'une doublure de capot Renault

Dans ce chapitre, nous proposons une méthodologie pour la gestion des raccordements entre les patchs CAO. Nous détaillerons également la mise en place de la méthode de résolution numérique que nous illustrerons avec quelques applications.

3.2 Le raccordement entre patchs

Les méthodes classiques de calcul, dans le cas de raccordement entre surfaces, consistent à imposer l'égalité des déplacements de part et d'autre de la frontière commune aux deux surfaces. Cependant, cette hypothèse est suffisante uniquement dans le cas où le raccord est de type $\mathcal{C}^1$. En effet, dans cette configuration, l'égalité des déplacements entraîne aussi la transmission des moments d'un patch vers l'autre. En automobile, et plus généralement dans l'industrie, des pièces complexes comme des membrures de caisse présentent parfois des raccordement de type $\mathcal{C}^0$ se traduisant par des discontinuités de courbure. Pour ce type de géométrie, imposer des égalités de déplacement ne garantit plus la transmission des moments et, de ce fait, la cohésion de la pièce étudiée. Il est donc nécessaire de mettre en œuvre des conditions de jonction entre patchs CAO pouvant tenir compte des discontinuités de courbure.

3.2.1 Le raccordement entre patchs CAO en isogéométrie

La question du raccordement entre les patchs CAO dans le cadre d'une analyse isogéométrique pour des coques a déjà été discutée. Il est proposé dans [Kiendl, 2010] des méthodologies pour traiter différents types de raccord. Dans le cas d'un raccordement entre surfaces NURBS suffisamment régulières,

la connexion entre les patchs repose sur la continuité $\mathcal{G}^1$. La continuité $\mathcal{G}^1$ est une continuité dite géométrique. Elle se définit de la façon suivante :

Définition 3.2.1. Soient $\mathcal{C}^1(\xi)$ et $\mathcal{C}^2(\xi)$ avec $0 \leq \xi \leq 1$ deux courbes paramétriques partageant un point commun telles que

$$\mathcal{C}^1(1) = \mathcal{C}^2(0). \tag{3.2.1}$$

La jonction entre les deux courbes est de continuité $\mathcal{G}^1$ si les vecteurs tangents au point commun sont parallèles. Cette condition se traduit par l'équation (3.2.2).

$$\frac{\partial \mathcal{C}^1(1)}{\partial \xi} = c\frac{\partial \mathcal{C}^2(0)}{\partial \xi} \tag{3.2.2}$$

où c est un scalaire.

La continuité $\mathcal{G}^1$ est moins forte que la continuité $\mathcal{C}^1$ au sens où la continuité $\mathcal{C}^1$ impose l'égalité entre les dérivées du premier ordre alors que la continuité $\mathcal{G}^1$ ne nécessite qu'une relation de proportionnalité. Pour des courbes B-splines, cette égalité peut être explicitée et exprimée en fonction des points de contrôle. En effet, si P_n^1 est le dernier point de contrôle de la première courbe et P_1^2 le premier point de contrôle de la seconde alors :

$$(P_2^2 - P_1^2) = c(P_n^1 - P_{n-1}^1) \tag{3.2.3}$$

De plus, puisque $P_n^1 = P_1^2$, il vient que

$$(P_2^2 - P_n^1) = c(P_n^1 - P_{n-1}^1) \tag{3.2.4}$$

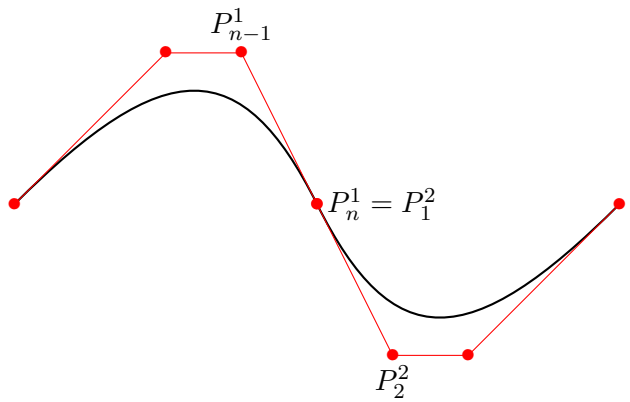


FIGURE 3.2: Continuité $\mathcal{G}^1$ entre deux courbes B-splines

La définition 3.2.1 se généralise aux surfaces et l'égalité (3.2.4) peut être écrite pour des NURBS à l'aide de points de contrôle pondérés le long des frontières communes aux surfaces. En posant $P_i = (x_i, y_i, z_i)$ et $P_i^w = \{w_i x_i, w_i y_i, w_i z_i, w_i\}$, la condition de jonction entre deux patchs est alors définie par l'équation suivante :

$$(P_{2,j}^{w,2} - P_{n,j}^{w,1}) = c(P_{n,j}^{w,1} - P_{n-1,j}^{w,1}) \quad j = 1, ..., m \tag{3.2.5}$$

Lors des simulations, la condition (3.2.5) doit toujours être vérifiée pour assurer l'intégrité du résultat. Cependant, cette condition peut être insuffisante face à des jonctions entre patchs présentant des discontinuités de courbure. Dans ce cas, l'angle entre les deux patchs dans la configuration initiale doit être conservé dans la configuration déformée. Pour des surfaces NURBS, l'équation (3.2.6) traduit cette condition à l'aide des points de contrôle, l'angle α entre les deux patchs devant rester constant :

$$\alpha = cos^{-1}\left(\frac{(P_{n,j}^1 - P_{n-1,j}^1).(P_{2,j}^2 - P_{n,j}^1)}{|P_{n,j}^1 - P_{n-1,j}^1|.|P_{2,j}^2 - P_{n,j}^1|}\right). \tag{3.2.6}$$

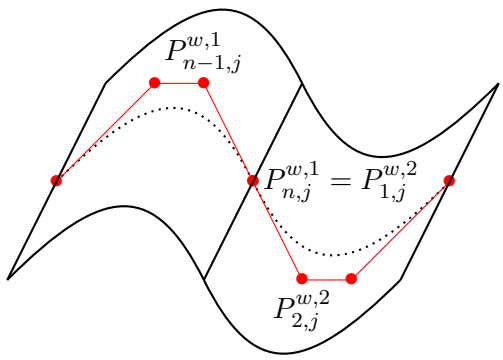


FIGURE 3.3: Continuité $\mathcal{G}^1$ entre deux surfaces NURBS

La relation de l'équation (3.2.6) s'avère être non linéaire et donc difficile à mettre en œuvre dans les simulations. Afin de pallier au caractère non linéaire de cette contrainte, une solution est de considérer une condition faible, c'est-à-dire de supposer que l'angle reste seulement approximativement constant au cours de la déformation. Il faut alors créer un nouveau patch NURBS, appelé *bande de flexion* aux propriétés particulières et formé à partir des points de contrôle à l'interface de chacun des patchs (voir figure 3.4). C'est au travers de cette *bande de flexion*, qui modifie la raideur associée à chaque patch, que la contrainte sera appliquée au cours des calculs. L'implémentation d'une *bande de flexion* est détaillée dans [Kiendl, 2010].

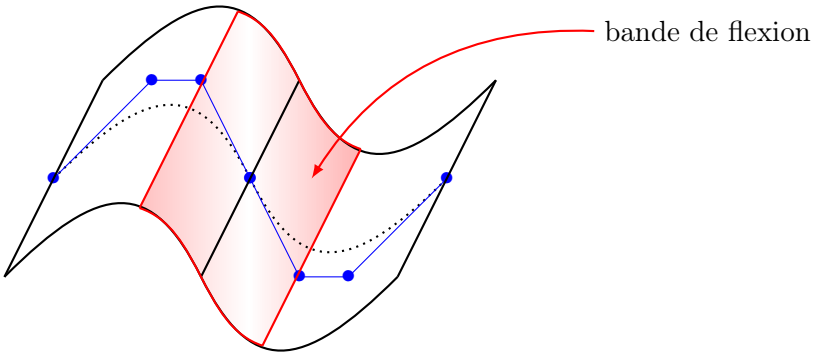


FIGURE 3.4: Représentation de la bande de flexion pour le raccordement entre patchs.

Ces méthodes de raccordement fournissent des résultats satisfaisants et ont l'avantage de pouvoir être explicitées en fonction des points de contrôle. Cependant, dans le cadre de notre étude, ce type de raccordement peut s'avérer délicat à mettre en œuvre. En effet, les calculs étant effectués sur un domaine plan indépendant de la forme, il faudrait alors inscrire ces conditions de jonction dans la fonction de forme. Ces opérations alourdiraient alors la fonction de forme qui ne serait plus simplement l'équation de la surface. L'alternative que nous proposons est de définir des conditions de raccord qui seront exprimées sur le domaine de référence, en utilisant les équations de jonction de coques développées dans [Bernadou, 1994], [Bernadou et Cubier, 1996a] et [Bernadou et Cubier, 1996b].

3.2.2 Mise en œuvre pratique

L'un des principaux intérêts de l'approche que nous proposons ici est que tous les calculs sont effectués sur un domaine qui est indépendant de la forme. Dans cette configuration, les équations de jonction

entre les différents patchs qui composent la forme doivent être adaptées. Nous développons une démarche qui consiste à considérer chaque patch CAO comme une coque et les raccords entre les patchs sont intégrés sous forme de jonctions entre les coques. Les problèmes de jonctions sont établis à partir des équations du modèle de Koiter mais également en fonction du comportement de la jonction. En effet, selon le type d'élément de liaison, le comportement mécanique de la structure varie. Nous considérerons dans la suite deux sortes de raccord : les *charnières rigides* et les *charnières élastiques*. Cette approche permet de décrire plus précisément les phénomènes mécaniques qui apparaissent au niveau de la jonction et ceci, sans tenir compte de la régularité des surfaces et du raccord. Cette technique permet ainsi de traiter des formes présentant des discontinuités de courbure.

3.2.2.1 Mise en équations du problème de jonction de coques

Afin de pouvoir effectuer des simulations sur des pièces comprenant plusieurs patchs, il est nécessaire dans un premier temps de déterminer la formulation variationnelle associée aux problèmes de jonctions de coques. Cette dernière repose sur l'expression des équations du modèle du Koiter appliquées à chacune des coques tout en tenant compte du principe d'action-réaction pour assurer la transmission des efforts d'une coque vers une autre.

Proposition 3.2.1. Considérons deux coques $\mathcal{C}$ et $\tilde{\mathcal{C}}$ de surface moyenne respective S et $\tilde{S}$ images des domaines plans Ω et $\tilde{\Omega}$ par les applications $\vec{\varphi}$ et $\vec{\tilde{\varphi}}$. Supposons que :

- S (respectivement $\tilde{S}$) est encastrée sur une partie ∂S_0 (respectivement $\partial \tilde{S}_0$) de sa frontière ∂S (respectivement $\partial \tilde{S}$) ;
- S (respectivement $\tilde{S}$) est chargée sur une partie ∂S_1 (respectivement $\partial \tilde{S}_1$) de sa frontière ;
- S et $\tilde{S}$ partagent une frontière commune Γ.

Ainsi :

- $\partial S = \partial S_0 \cup \partial S_1 \cup \Gamma$ soit $\partial \Omega = \partial \Omega_0 \cup \partial \Omega_1 \cup \gamma$ où $\gamma = (\vec{\varphi})^{-1}(\Gamma)$
- $\partial \tilde{S} = \partial \tilde{S}_0 \cup \partial \tilde{S}_1 \cup \Gamma$ soit $\partial \tilde{\Omega} = \partial \tilde{\Omega}_0 \cup \partial \tilde{\Omega}_1 \cup \tilde{\gamma}$ où $\tilde{\gamma} = (\vec{\tilde{\varphi}})^{-1}(\Gamma)$

La figure 3.5 représente la configuration décrite.

Soient :

- $\vec{p}$ (resp. $\vec{\tilde{p}}$) la résultante sur S (resp. $\tilde{S}$) des charges extérieures ;
- $\vec{N}$ (resp. $\vec{\tilde{N}}$) la résultante sur ∂S (resp. $\partial \tilde{S}$) des charges extérieures ;
- $\vec{M}$ (resp. $\vec{\tilde{M}}$) le moment résultant sur ∂S (resp. $\partial \tilde{S}$) des charges appliquées au bord latéral de la coque ;
- $\vec{\psi}(\vec{v})$ et $\vec{\tilde{\psi}}(\vec{\tilde{v}})$ les vecteurs de rotation infinitésimale.

Alors, la formulation variationnelle du problème avec une jonction est de la forme :

$$\text{trouver } (\vec{u}, \vec{\tilde{u}}) \in W \text{ tel que } \quad \forall (\vec{v}, \vec{\tilde{v}}) \in W, \quad a[(\vec{u}, \vec{\tilde{u}}), (\vec{v}, \vec{\tilde{v}})] + b[(\vec{v}, \vec{\tilde{v}})] = l(\vec{v}, \vec{\tilde{v}}) \tag{3.2.7}$$

avec :

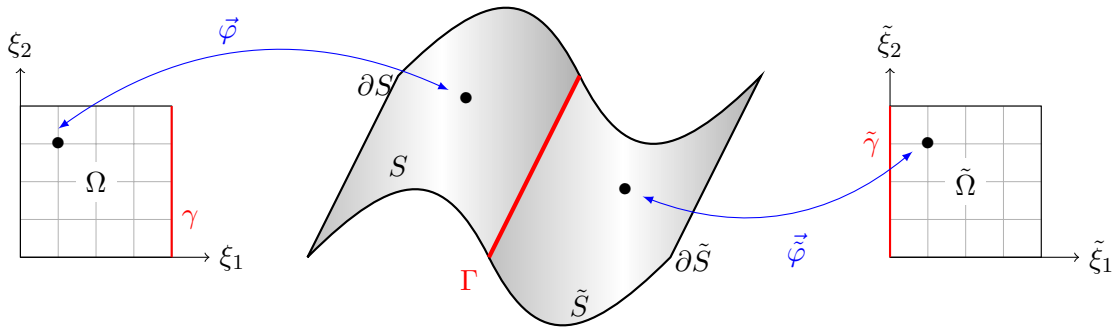


FIGURE 3.5: Jonction entre deux coques minces.

$$\begin{cases} a[(\vec{u},\vec{\tilde{u}}),(\vec{v},\vec{\tilde{v}})] &= \displaystyle\int_\Omega eE^{\alpha\beta\lambda\mu}\left[\gamma_{\alpha\beta}(\vec{u})\gamma_{\lambda\mu}(\vec{v})+\frac{e^2}{12}\varrho_{\alpha\beta}(\vec{u})\varrho_{\lambda\mu}(\vec{v})\right]\sqrt{a}\,d\xi_1 d\xi_2 \\ &+\displaystyle\int_{\tilde{\Omega}}\tilde{e}\tilde{E}^{\alpha\beta\lambda\mu}\left[\tilde{\gamma}_{\alpha\beta}(\vec{\tilde{u}})\tilde{\gamma}_{\lambda\mu}(\vec{\tilde{v}})+\frac{\tilde{e}^2}{12}\tilde{\varrho}_{\alpha\beta}(\vec{\tilde{u}})\tilde{\varrho}_{\lambda\mu}(\vec{\tilde{v}})\right]\sqrt{\tilde{a}}\,d\tilde{\xi}_1 d\tilde{\xi}_2 \end{cases} \tag{3.2.8}$$

$$b[(\vec{v},\vec{\tilde{v}})] = \int_\gamma M_n[n^\beta(v_{3,\beta}+b^\alpha_\beta v_\alpha)-\tilde{n}^\beta(\vec{t}\,.\,\vec{\tilde{t}})(\tilde{v}_{3,\beta}+\tilde{b}^\alpha_\beta\tilde{v}_\alpha)]\,d\gamma \tag{3.2.9}$$

$$\begin{cases} l(\vec{v},\vec{\tilde{v}}) &= \displaystyle\int_\Omega \vec{p}\,\vec{v}\sqrt{a}\,d\xi_1 d\xi_2+\int_{\partial\Omega_1}\vec{N}\vec{v}+\vec{M}\vec{\psi}(\vec{v})ds+\int_{\tilde{\Omega}}\vec{\tilde{p}}\,\vec{\tilde{v}}\sqrt{\tilde{a}}\,d\tilde{\xi}_1 d\tilde{\xi}_2 \\ &+\displaystyle\int_{\partial\tilde{\Omega}_1}\vec{\tilde{N}}\vec{\tilde{v}}+\vec{\tilde{M}}\vec{\tilde{\psi}}(\vec{\tilde{v}})d\tilde{s}+\int_\gamma \vec{N}.(\vec{v}-\vec{\tilde{v}})\,d\gamma \end{cases} \tag{3.2.10}$$

et W l'espace des champs de déplacement cinématiquement admissibles.

Démonstration. Dans un premier temps, considérons les deux coques indépendemment l'une de l'autre. Les équations du modèles de Koiter donnent alors :

$$\begin{cases} \forall\vec{v}\in\left\{(v_\alpha,v_3)\in((H^1(\Omega))^2\times H^2(\Omega));\ \vec{v}|_{\partial\Omega_0}=\vec{0},\ \left.\dfrac{\partial v_3}{\partial n}\right|_{\partial\Omega_0}=0\right\} \\ \displaystyle\int_\Omega eE^{\alpha\beta\lambda\mu}\left[\gamma_{\alpha\beta}(\vec{u})\gamma_{\lambda\mu}(\vec{v})+\frac{e^2}{12}\varrho_{\alpha\beta}(\vec{u})\varrho_{\lambda\mu}(\vec{v})\right]\sqrt{a}\,d\xi_1 d\xi_2 \\ \displaystyle=\int_\Omega\vec{p}\,\vec{v}\sqrt{a}\,d\xi_1 d\xi_2+\int_{\partial\Omega_1}\vec{N}\vec{v}+\vec{M}\vec{\psi}(\vec{v})\,ds+\int_\gamma\vec{N}\vec{v}+\vec{M}\vec{\psi}(\vec{v})\,d\gamma \end{cases} \tag{3.2.11}$$

et

$$\begin{cases} \forall\vec{\tilde{v}}\in\left\{(\tilde{v}_\alpha,\tilde{v}_3)\in((H^1(\tilde{\Omega}))^2\times H^2(\tilde{\Omega}));\ \vec{\tilde{v}}|_{\partial\tilde{\Omega}_0}=\vec{0},\ \left.\dfrac{\partial\tilde{v}_3}{\partial\tilde{n}}\right|_{\partial\tilde{\Omega}_0}=0\right\} \\ \displaystyle\int_{\tilde{\Omega}}\tilde{e}\tilde{E}^{\alpha\beta\lambda\mu}\left[\tilde{\gamma}_{\alpha\beta}(\vec{\tilde{u}})\tilde{\gamma}_{\lambda\mu}(\vec{\tilde{v}})+\frac{\tilde{e}^2}{12}\tilde{\varrho}_{\alpha\beta}(\vec{\tilde{u}})\tilde{\varrho}_{\lambda\mu}(\vec{\tilde{v}})\right]\sqrt{\tilde{a}}\,d\tilde{\xi}_1 d\tilde{\xi}_2 \\ \displaystyle=\int_{\tilde{\Omega}}\vec{\tilde{p}}\,\vec{\tilde{v}}\sqrt{\tilde{a}}d\tilde{\xi}_1 d\tilde{\xi}_2+\int_{\partial\tilde{\Omega}_1}\vec{\tilde{N}}\vec{\tilde{v}}+\vec{\tilde{M}}\vec{\tilde{\psi}}(\vec{\tilde{v}})\,d\tilde{s}+\int_\gamma\vec{\tilde{N}}\vec{\tilde{v}}+\vec{\tilde{M}}\vec{\tilde{\psi}}(\vec{\tilde{v}})\,d\tilde{\gamma} \end{cases} \tag{3.2.12}$$

En sommant membre à membre les équations (3.2.11) et (3.2.12), il vient alors :

$$\begin{cases} \displaystyle\int_\Omega eE^{\alpha\beta\lambda\mu}\left[\gamma_{\alpha\beta}(\vec{u})\gamma_{\lambda\mu}(\vec{v}) + \frac{e^2}{12}\varrho_{\alpha\beta}(\vec{u})\varrho_{\lambda\mu}(\vec{v})\right]\sqrt{a}\, d\xi_1 d\xi_2 \\ \displaystyle + \int_{\tilde{\Omega}} \tilde{e}\tilde{E}^{\alpha\beta\lambda\mu}\left[\tilde{\gamma}_{\alpha\beta}(\vec{\tilde{u}})\tilde{\gamma}_{\lambda\mu}(\vec{\tilde{v}}) + \frac{\tilde{e}^2}{12}\tilde{\varrho}_{\alpha\beta}(\vec{\tilde{u}})\tilde{\varrho}_{\lambda\mu}(\vec{\tilde{v}})\right]\sqrt{\tilde{a}}\, d\tilde{\xi}_1 d\tilde{\xi}_2 \\ \displaystyle = \int_\Omega \vec{p}\,\vec{v}\sqrt{a}\, d\xi_1 d\xi_2 + \int_{\partial\Omega_1} \vec{N}\vec{v} + \vec{M}\vec{\psi}(\vec{v})ds + \int_{\tilde{\Omega}} \vec{\tilde{p}}\,\vec{\tilde{v}}\sqrt{\tilde{a}}\, d\tilde{\xi}_1 d\tilde{\xi}_2 \\ \displaystyle + \int_{\partial\tilde{\Omega}_1} \vec{\tilde{N}}\vec{\tilde{v}} + \vec{\tilde{M}}\vec{\tilde{\psi}}(\vec{\tilde{v}})d\tilde{s} + \int_\gamma \vec{N}\vec{v} + \vec{M}\vec{\psi}(\vec{v})\, d\gamma + \int_\gamma \vec{\tilde{N}}\vec{\tilde{v}} + \vec{\tilde{M}}\vec{\tilde{\psi}}(\vec{\tilde{v}})\, d\tilde{\gamma} \end{cases} \tag{3.2.13}$$

D'autre part, le principe d'action-réaction en tout point P de la frontière Γ implique la transmission des efforts. Ainsi :

$$\vec{N}(P) = \vec{\tilde{N}}(P) \quad \text{et} \quad \vec{M}(P) = \vec{\tilde{M}}(P), \quad \forall P \in \Gamma$$

Par ailleurs :

$$\vec{M} = \varepsilon_{\alpha\beta}M^\alpha \vec{a}_\beta \quad \text{et} \quad \vec{\psi}(\vec{v}) = \varepsilon^{\lambda\beta}(v_{3,\beta} + b^\alpha_\beta v_\alpha)\vec{a}_\lambda + \frac{1}{2}\varepsilon^{\lambda\beta}v_{\beta|\lambda}\vec{a}_3$$

avec :

- $\varepsilon_{11} = \varepsilon_{22} = 0$
- $\varepsilon_{12} = \sqrt{a},\ \varepsilon_{21} = -\sqrt{a}$

Par conséquent :

$$\begin{aligned} \vec{M}\cdot\vec{\psi}(\vec{v}) &= (\varepsilon_{\lambda\mu}M^\lambda\vec{a}_\mu)\cdot\left(\varepsilon^{\lambda\beta}(v_{3,\beta} + b^\alpha_\beta v_\alpha)\vec{a}_\lambda + \frac{1}{2}\varepsilon^{\lambda\beta}v_{\beta|\lambda}\vec{a}_3\right) \\ &= \varepsilon_{\lambda\mu}\varepsilon^{\lambda\beta}M^\lambda\vec{a}_\mu\cdot(v_{3,\beta} + b^\alpha_\beta v_\alpha)\vec{a}_\lambda \end{aligned}$$

Soient $\vec{n}$ le vecteur normal à la frontière ∂S dans le plan tangent et $\vec{t}$ le vecteur tangent à la frontière. Ainsi :

$$\begin{aligned} \vec{M}\cdot\vec{\psi}(\vec{v}) &= \varepsilon_{\lambda\mu}\varepsilon^{\lambda\beta}(M_n\vec{n} + M_t\vec{t})\cdot(v_{3,\beta} + b^\alpha_\beta v_\alpha)\vec{a}_\lambda \\ &= \varepsilon_{\lambda\mu}\varepsilon^{\lambda\beta}(M_n n_\alpha\vec{a}^\alpha + M_t t_\alpha\vec{a}^\alpha)\cdot(v_{3,\beta} + b^\alpha_\beta v_\alpha)\vec{a}_\lambda \\ &= (t^\beta M_t + n^\beta M_n)\cdot(v_{3,\beta} + b^\alpha_\beta v_\alpha) \end{aligned} \tag{3.2.14}$$

Revenons à l'équation (3.2.13). D'après [Bernadou, 1994], en observant que :

$$d\tilde{\gamma} = -d\gamma, \quad \vec{N} = \vec{\tilde{N}} \quad \text{et} \quad \vec{\tilde{M}} = \vec{M} = -M_n\vec{t}$$

Alors ,

$$\begin{aligned} &\int_\gamma \vec{N}\vec{v} + \vec{M}\vec{\psi}(\vec{v})d\gamma + \int_\gamma \vec{\tilde{N}}\vec{\tilde{v}} + \vec{\tilde{M}}\vec{\tilde{\psi}}(\vec{\tilde{v}})d\tilde{\gamma} \\ &\qquad = \int_\gamma \vec{N}\cdot(\vec{v} - \vec{\tilde{v}}) + M_n[n^\beta(v_{3,\beta} + b^\alpha_\beta v_\alpha) - \tilde{n}^\beta(\vec{t}\cdot\vec{\tilde{t}})(\tilde{v}_{3,\beta} + \tilde{b}^\alpha_\beta\tilde{v}_\alpha)]\, d\gamma \end{aligned}$$

□

Le problème décrit par l'équation (3.2.7) est incomplet puisqu'il reste à définir W l'espace auquel les solutions du problème variationnel appartiennent. Dans le cas d'une coque seule, l'espace des déplacements cinématiquement admissibles ne dépend que de la coque en question et s'écrit :

$$W = \left\{ (v_\alpha, v_3) \in ((H^1(\Omega))^2 \times H^2(\Omega)); \quad \vec{v}|_{\partial\Omega_0} = \vec{0}, \quad \frac{\partial v_3}{\partial n}\Big|_{\partial\Omega_0} = 0. \right\} \tag{3.2.15}$$

En présence d'une jonction, il faut tenir compte des déplacements au niveau de la frontière qui sont influencés par la présence d'une autre coque et, par conséquent, par l'existence d'une charnière. Le comportement de la charnière choisie joue donc un rôle majeur. Nous présentons dans la section suivante le comportement des deux charnières étudiées, ce qui constitue la seconde étape dans l'écriture de la formulation variationnelle du problème de jonction.

3.2.2.2 Comportement des charnières

Les deux types d'éléments de liaison entre patchs que nous considérons ici sont les charnières dites rigides et les charnières dites élastiques. Ces deux charnières possèdent des propriétés mécaniques différentes.

Les charnières rigides

Par définition, une charnière rigide est une liaison entre plusieurs pièces qui ne peut pas être déformée. Ce type de liaison comporte un organe de liaison rigide (vis, écrou) ou les surfaces des pièces qu'elle lie ont des formes complémentaires. Dans des conditions normales d'utilisation d'une pièce, une charnière rigide impose aux structures assemblées une position relative fixe qui ne peut pas être modifiée par des sollicitations extérieures (voir figure 3.6).

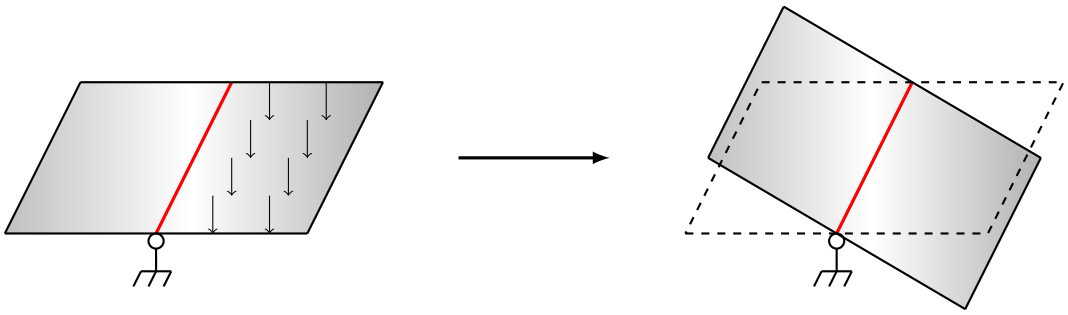


FIGURE 3.6: Exemple d'une charnière rigide.

D'un point de vue mécanique, le comportement d'une charnière rigide se traduit par la continuité des déplacements de part et d'autre de la charnière ($\vec{u}$ et $\vec{\tilde{u}}$) et de la composante tangentielle de rotation en tout point P de la charnière Γ. Ainsi,

$$\vec{u}(P) = \vec{\tilde{u}}(P), \quad (\vec{\psi} \cdot \vec{t})(P) = (\vec{\tilde{\psi}} \cdot \vec{t})(P) = (\vec{t} \cdot \vec{\tilde{t}})(\vec{\tilde{\psi}} \cdot \vec{\tilde{t}})(P), \quad \forall P \in \Gamma \tag{3.2.16}$$

Les charnières élastiques

Contrairement à une charnière rigide, une charnière élastique est une liaison déformable. Elle lie les pièces par une force de pression ou de compression comme à la manière d'un ressort. Elle assure aux pièces un déplacement relatif limité. La position relative des pièces est fonction de l'intensité des sollicitations extérieures (voir figure 3.7).

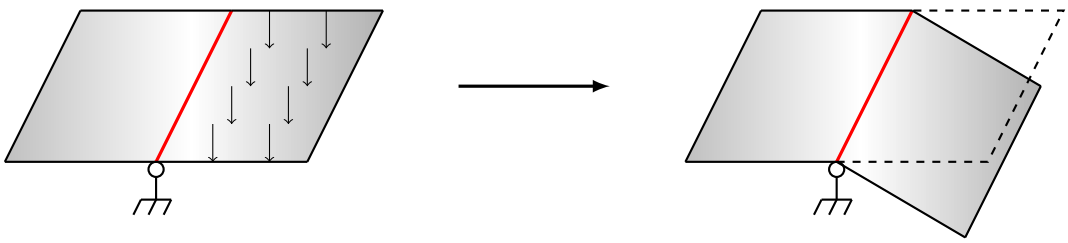


FIGURE 3.7: Exemple d'une charnière élastique.

Mécaniquement, le comportement d'une charnière élastique se traduit par la continuité des déplacements de part et d'autre de la charnière ($\vec{u}$ et $\vec{\tilde{u}}$) et la proportionnalité de la composante tangentielle du moment $\vec{M}$ par rapport au saut des composantes tangentielles de rotation le long de la charnière Γ. Ainsi, la condition s'écrit :

$$\vec{u}(P) = \vec{\tilde{u}}(P) \qquad M_n(P) = k[(\vec{\psi} - \vec{\tilde{\psi}}) \cdot \vec{t}](P), \quad \forall P \in \Gamma \tag{3.2.17}$$

où k désigne le coefficient de rigidité élastique de la charnière et M_n la composante normale du moment. Le coefficient k peut être déterminé expérimentalement ou par développement asymptotique.

3.2.2.3 Formulations variationnelles des problèmes de jonction

Pour compléter l'écriture de la formulation variationnelle de l'équation (3.2.7), il reste à déterminer W, l'espace des déplacements cinématiquement admissibles. Il faut donc intégrer les propriétés associées à la charnière. Commençons par le cas d'une jonction de coque avec une charnière rigide.

Proposition 3.2.2. La formulation variationnelle du problème de jonction entre deux coques avec une charnière rigide s'écrit :

$$\text{trouver } (\vec{u}, \vec{\tilde{u}}) \in W_{rig} \text{ tel que } \quad \forall (\vec{v}, \vec{\tilde{v}}) \in W_{rig}, \quad a[(\vec{u}, \vec{\tilde{u}}), (\vec{v}, \vec{\tilde{v}})] = l(\vec{v}, \vec{\tilde{v}}) \tag{3.2.18}$$

avec :

$$W_{rig} = \{(\vec{v}, \vec{\tilde{v}}) \in \vec{V} \times \vec{\tilde{V}},\ \vec{v} = \vec{\tilde{v}} \text{ sur } \gamma\ ;\ n_\beta(v_{3,\beta} + b^\alpha_\beta v_\alpha) - (\vec{t}.\vec{\tilde{t}})\tilde{n}_\beta(\tilde{v}_{3,\beta} + \tilde{b}^\alpha_\beta \tilde{v}_\alpha) = 0 \text{ sur } \gamma\} \tag{3.2.19}$$

où a et l sont données par (3.2.8) et (3.2.10).

Démonstration. Rappelons que le comportement d'une charnière rigide Γ est défini par :

$$\forall P \in \Gamma, \quad \vec{u}(P) = \vec{\tilde{u}}(P) \qquad (\vec{\psi}.\vec{t})(P) = (\vec{\tilde{\psi}}.\vec{t})(P) = (\vec{t}.\vec{\tilde{t}})(\vec{\tilde{\psi}}.\vec{\tilde{t}})(P)$$

L'égalité des déplacements impose :

$$\vec{v} = \vec{\tilde{v}} \text{ sur } \gamma$$

D'autre part,

$$\begin{aligned}
\vec{\psi}(\vec{v}).\vec{t} &= (\varepsilon^{\lambda\beta}(v_{3,\beta} + b^\alpha_\beta v_\alpha)\vec{a}_\lambda + \frac{1}{2}(\varepsilon^{\lambda\beta} v_{\beta|\lambda}\vec{a}_3)).\vec{t} \\
&= \varepsilon^{\lambda\beta}(v_{3,\beta} + b^\alpha_\beta v_\alpha)\vec{a}_\lambda.\vec{t} \\
&= \varepsilon^{\lambda\beta}(v_{3,\beta} + b^\alpha_\beta v_\alpha)\vec{a}_\lambda.(\vec{a}_3 \wedge \vec{n}) \\
&= \varepsilon^{\lambda\beta}(v_{3,\beta} + b^\alpha_\beta v_\alpha)\vec{a}_\lambda.(\vec{a}_3 \wedge n_a\vec{a}^\alpha) \\
&= \varepsilon^{\lambda\beta}\varepsilon^{\alpha\lambda}(v_{3,\beta} + b^\alpha_\beta v_\alpha)\vec{a}_\lambda.(n_\alpha\vec{a}_\alpha) \\
&= n_\beta(v_{3,\beta} + b^\alpha_\beta v_\alpha)
\end{aligned}$$

Alors,

$$\vec{\psi}(\vec{v}).\vec{t} - (\vec{t}.\vec{\tilde{t}})\vec{\tilde{\psi}}(\vec{\tilde{v}}).\vec{\tilde{t}} = n_\beta(v_{3,\beta} + b^\alpha_\beta v_\alpha) - (\vec{t}.\vec{\tilde{t}})\tilde{n}_\beta(\tilde{v}_{3,\beta} + \tilde{b}^\alpha_\beta \tilde{v}_\alpha) = 0$$

Par conséquent, la forme b dans (3.2.9) devient :

$$b[(\vec{v}, \vec{\tilde{v}})] = 0$$

□

Dans cette configuration, la forme bilinéaire b disparaît. En plus de l'égalité des déplacements de part et d'autre de la frontière, l'espace des déplacements cinématiques intègre une condition supplémentaire liée à la transmission des moments.

Étant donné que les hypothèses sont différentes, la formulation variationnelle pour les jonctions de coques au moyen de charnières élastiques est différente de la formulation pour les charnières rigides.

Proposition 3.2.3. La formulation variationnelle du problème de jonction entre deux coques avec une charnière élastique est donnée par :

$$\text{trouver } (\vec{u}, \vec{\tilde{u}}) \in W_{elas} \text{ tel que } \quad \forall (\vec{v}, \vec{\tilde{v}}) \in W_{elas}, \quad a[(\vec{u}, \vec{\tilde{u}}), (\vec{v}, \vec{\tilde{v}})] + b[(\vec{u}, \vec{\tilde{u}}), (\vec{v}, \vec{\tilde{v}})] = l(\vec{v}, \vec{\tilde{v}}) \quad (3.2.20)$$

avec :

$$W_{elas} = \{(\vec{v}, \vec{\tilde{v}}) \in \vec{V} \times \vec{\tilde{V}},\ \vec{v} = \vec{\tilde{v}} \text{ sur } \gamma\ \} \quad (3.2.21)$$

et

$$b[(\vec{u}, \vec{\tilde{u}}), (\vec{v}, \vec{\tilde{v}})] = \int_\gamma k[n^\beta(u_{3,\beta} + b^\alpha_\beta u_\alpha) - \tilde{n}^\beta(\vec{t}.\vec{\tilde{t}})(\tilde{u}_{3,\beta} + \tilde{b}^\alpha_\beta \tilde{u}_\alpha)][n^\beta(v_{3,\beta} + b^\alpha_\beta v_\alpha) - \tilde{n}^\beta(\vec{t}.\vec{\tilde{t}})(\tilde{v}_{3,\beta} + \tilde{b}^\alpha_\beta \tilde{v}_\alpha)]d\gamma \quad (3.2.22)$$

où a et l sont données par (3.2.8) et (3.2.10).

Démonstration. Rappelons que le comportement d'une charnière élastique est définie par les conditions suivantes :

$$\forall P \in \Gamma, \quad \vec{u}(P) = \vec{\tilde{u}}(P) \quad M_n(P) = k[(\vec{\psi} - \vec{\tilde{\psi}}).\vec{t}](P),$$

où k désigne le coefficient de rigidité élastique de la charnière.
L'égalité des déplacements impose :

$$\vec{v} = \vec{\tilde{v}} \ \text{ sur } \gamma$$

D'autre part, sachant que :

$$(\vec{\psi} - \vec{\tilde{\psi}}).\vec{t} = n_\beta(v_{3,\beta} + b^\alpha_\beta v_\alpha) - (\vec{t}.\vec{\tilde{t}})\tilde{n}_\beta(\tilde{v}_{3,\beta} + \tilde{b}^\alpha_\beta \tilde{v}_\alpha)$$

Il vient :

$$M_n = k[n_\beta(v_{3,\beta} + b^\alpha_\beta v_\alpha) - (\vec{t}.\vec{\tilde{t}})\tilde{n}_\beta(\tilde{v}_{3,\beta} + \tilde{b}^\alpha_\beta \tilde{v}_\alpha)]$$

Par conséquent, la forme b dans (3.2.9) devient :

$$b[(\vec{u}, \vec{\tilde{u}}), (\vec{v}, \vec{\tilde{v}})] = \int_\gamma k[n^\beta(u_{3,\beta} + b^\alpha_\beta u_\alpha) - \tilde{n}^\beta(\vec{t}.\vec{\tilde{t}})(\tilde{u}_{3,\beta} + \tilde{b}^\alpha_\beta \tilde{u}_\alpha)][n^\beta(v_{3,\beta} + b^\alpha_\beta v_\alpha) - \tilde{n}^\beta(\vec{t}.\vec{\tilde{t}})(\tilde{v}_{3,\beta} + \tilde{b}^\alpha_\beta \tilde{v}_\alpha)]d\gamma$$

□

À l'inverse de la formulation pour les charnières rigides, les problèmes pour les charnières élastiques conservent la forme bilinéaire b et l'espace W a pour seule condition l'égalité des déplacements de part et d'autre de la frontière commune.

Le théorème suivant assure que les problèmes variationnels décrits plus haut pour ces deux types de charnières admettent chacun une solution unique.

Théorème 3.2.1 (Théorème 3.2.1 p.226 dans [Bernadou, 1994]). Pour des données suffisamment régulières et pour $\vec{p} \in (L^2(\partial\Omega_1))^3, \vec{N} \in (L^2(\partial\Omega_1))^3, \vec{M} \in (L^2(\partial\Omega_1))^3, \vec{\tilde{p}} \in (L^2(\partial\tilde{\Omega}_1))^3, \vec{\tilde{N}} \in (L^2(\partial\Omega_1))^3, \vec{\tilde{M}} \in (L^2(\partial\Omega_1))^3, k > 0, E > 0, \tilde{E} > 0, 0 < \nu < 1/2$ *et* $0 < \tilde{\nu} < 1/2$, alors les problèmes (3.2.18) et (3.2.20) ont chacun une solution unique.

3.2.2.4 Choix du type de charnière

Il existe une relation entre la solution d'un problème à charnière rigide et celle d'un problème à charnière élastique. En effet, la solution du problème à charnière élastique tend fortement vers la solution d'un problème à charnière rigide lorsque le coefficient de rigidité de la charnière tend vers $+\infty$.

Théorème 3.2.2 (Théorème 3.2.3 p.227 dans [Bernadou, 1994]). Soit $(\vec{u}^k, \vec{\tilde{u}}^k)$ la solution du problème à charnière élastique (3.2.20) avec un coefficient $k > 0$ et soit $(\vec{u}, \vec{\tilde{u}})$ la solution du problème à charnière rigide (3.2.18). Alors,

$$\lim_{k\to\infty} ||(\vec{u}^k, \vec{\tilde{u}}^k) - (\vec{u}, \vec{\tilde{u}})||_{W_{elas}} = 0.$$

Étant donné que les patchs CAO sont liés par la complémentarité de leur forme, les jonctions associées aux raccordements de patchs sont des charnières rigides. Cependant, l'implémentation du problème à charnière rigide est plus complexe que celle du problème à charnière élastique. Ainsi, nous avons choisi d'implémenter et de résoudre le problème à charnière élastique et de déterminer la solution du problème à charnière rigide à l'aide du théorème précédent (théorème 3.2.2).

Remarque 3.2.1. En pratique, le coefficient de rigidité de la charnière k sera choisi égal au module de Young du matériau considéré.

3.3 Méthode de résolution numérique

Dans cette partie, nous détaillons l'implémentation de la résolution numérique par éléments finis. Nous présentons ensuite comment la méthode de paramétrage, le raccordement entre patchs et la résolution par éléments finis ont été intégrés dans le code de calcul.

3.3.1 Mise en œuvre de la méthode des éléments finis

3.3.1.1 Écriture du problème sous forme matricielle

Lors de l'implémentation d'une méthode éléments finis, il est souvent judicieux d'exprimer la formulation variationnelle sous forme matricielle. Pour un modèle de Koiter, la forme bilinéaire :

$$a(\vec{u}, \vec{v}) = \int_\Omega eE^{\alpha\beta\lambda\mu}\left[\gamma_{\alpha\beta}(\vec{u})\gamma_{\lambda\mu}(\vec{v}) + \frac{e^2}{12}\varrho_{\alpha\beta}(\vec{u})\varrho_{\lambda\mu}(\vec{v})\right]\sqrt{a}\, d\xi_1 d\xi_2 \tag{3.3.1}$$

peut être mise sous la forme :

$$a(\vec{u}, \vec{v}) = \int_\Omega U^T K V\, d\xi_1 d\xi_2 \tag{3.3.2}$$

où le vecteur colonne V (et de façon similaire U) de dimension (12×1) est donné par :

$$V = [v_1\ v_{1,1}\ v_{1,2}\ v_2\ v_{2,1}\ v_{2,2}\ v_3\ v_{3,1}\ v_{3,2}\ v_{3,11}\ v_{3,12}\ v_{3,22}]. \tag{3.3.3}$$

La matrice $[K]$ de dimension (12×12) vérifie :

$$K = \frac{Ee}{1-\nu^2}\sqrt{a}\left\{(1-\nu)\Lambda^{\alpha T}_\beta \Lambda^\beta_\alpha + \nu\Lambda^\alpha_\alpha\Lambda^\beta_\beta + \frac{e^2}{12}(1-\nu)M^{\alpha T}_\beta M^\beta_\alpha + \frac{e^2}{12}\nu M^\alpha_\alpha M^\beta_\beta\right\}. \tag{3.3.4}$$

avec :

- Λ_α^β quatre matrices de taille (1×12) telles que $\Lambda_\alpha^\beta V = \gamma_{\alpha\beta}(\vec{v})$ (le tenseur des déformations) ;
- M_α^β quatre matrices de taille (1×12) telles que $M_\alpha^\beta V = \varrho_{\alpha\beta}(\vec{v})$ (le tenseur de changement de courbure).

L'expression matricielle de la forme linéaire :

$$l(\vec{v}) = \int_\Omega \vec{p}\,\vec{v}\sqrt{a}d\xi_1 d\xi_2 + \int_{\partial\Omega_1} \vec{N}\,\vec{v} + \vec{M}\vec{\psi}(\vec{v})ds + \int_\gamma \vec{N}\,\vec{v} + \vec{M}\vec{\psi}(\vec{v})\, d\gamma \tag{3.3.5}$$

varie en fonction du type de chargement considéré. Elle prend la forme :

$$l(\vec{v}) = \int_\Omega F^T V\, d\xi_1 d\xi_2. \tag{3.3.6}$$

avec

- $F = p\sqrt{a}[0\ 0\ 0\ 0\ 0\ 0\ (-1)\ 0\ 0\ 0\ 0\ 0]$ si le chargement est une pression d'intensité p ;
- $F = \rho_1 g e\sqrt{a}[(\vec{e}_3 \cdot \vec{a}^1)\ 0\ 0\ (\vec{e}_3 \cdot \vec{a}^2)\ 0\ 0\ (\vec{e}_3 \cdot \vec{a}^3)\ 0\ 0\ 0\ 0\ 0]$ pour les charges de types gravitationnelles (ρ_1 la densité surfacique, g l'accélération de la pesanteur, $\vec{e}_3$ le vecteur de la composante z dans le repère cartésien) ;
- $F = p_j\sqrt{a}[(\vec{e}_j \cdot \vec{a}^1)\ 0\ 0\ (\vec{e}_j \cdot \vec{a}^2)\ 0\ 0\ (\vec{e}_j \cdot \vec{a}^3)\ 0\ 0\ 0\ 0\ 0]$ pour des charges surfaciques $\vec{p} = p_j\vec{e}^j$;
- $F = [0\ 0\ 0\ 0\ 0\ 0\ f\ 0\ 0\ 0\ 0\ 0]$ pour une force ponctuelle d'intensité f.

Il existe également d'autres types de forces, notamment des forces linéiques, que nous ne détaillerons pas ici mais qui sont explicitées dans [Bernadou, 1994]. Dans le reste de l'étude, nous nous intéresserons surtout aux chargements surfaciques et ponctuels.

À partir de cela, nous pouvons écrire le problème de jonction de coques sous forme matricielle. Introduisons les vecteurs $\tilde{U}$ et $\tilde{V}$ définis de la même façon que U et V pour l'autre coque. La matrice $\tilde{K}$ est l'homologue de la matrice K pour la deuxième coque. D'après l'équation (3.2.8), la forme bilinéaire a s'écrit sous forme matricielle de la façon suivante :

$$a[(\vec{u}, \vec{\tilde{u}}), (\vec{v}, \vec{\tilde{v}})] = \int_\Omega U^T K V d\xi_1 d\xi_2 + \int_{\tilde{\Omega}} \tilde{U}^T \tilde{K} \tilde{V} d\tilde{\xi}_1 d\tilde{\xi}_2. \tag{3.3.7}$$

Nous avons fait le choix d'implémenter le problème de charnière élastique. Nous allons donc exprimer sous forme de matrice la forme bilinéaire b donnée par l'équation (3.2.22) :

$$b[(u, \tilde{u}), (v, \tilde{v})] = \int_\gamma [U\ \tilde{U}]^T B[V\ \tilde{V}]ds. \tag{3.3.8}$$

La matrice B est une matrice creuse de dimension (24×24) dont la définition dépend uniquement de la géométrie de la charnière. Elle peut être écrite sous la forme réduite suivante :

$$\int_\gamma [U\ \tilde{U}]^T B[V\ \tilde{V}]ds = \int_\gamma [U_r\ \tilde{U}_r]^T B_r[V_r\ \tilde{V}_r]ds, \quad \text{avec} \quad B_r = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

où :

$$B_{11} = \begin{bmatrix} n^\alpha n^\beta b_\alpha^1 b_\beta^1 & n^\alpha n^\beta b_\alpha^1 b_\beta^2 & n^\alpha n^1 b_\alpha^1 & n^\alpha n^2 b_\alpha^1 \\ n^\alpha n^\beta b_\alpha^2 b_\beta^1 & n^\alpha n^\beta b_\alpha^2 b_\beta^2 & n^\alpha n^1 b_\alpha^2 & n^\alpha n^2 b_\alpha^2 \\ n^1 n^\beta b_\beta^1 & n^1 n^\beta b_\beta^2 & n^1 n^1 & n^1 n^2 \\ n^2 n^\beta b_\beta^1 & n^2 n^\beta b_\beta^2 & n^2 n^1 & n^2 n^2 \end{bmatrix}$$

$$B_{12} = B_{21}^T = \begin{bmatrix} -n^\alpha \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}}) b_\alpha^1 \tilde{b}_\beta^1 & -n^\alpha \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}}) b_\alpha^1 \tilde{b}_\beta^2 & -n^\alpha \tilde{n}^1 b_\alpha^1 (\vec{t}.\vec{\tilde{t}}) & -n^\alpha \tilde{n}^2 b_\alpha^1 (\vec{t}.\vec{\tilde{t}}) \\ -n^\alpha \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}}) b_\alpha^2 \tilde{b}_\beta^1 & -n^\alpha \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}}) b_\alpha^2 \tilde{b}_\beta^2 & -n^\alpha \tilde{n}^1 b_\alpha^2 (\vec{t}.\vec{\tilde{t}}) & -n^\alpha \tilde{n}^2 b_\alpha^2 (\vec{t}.\vec{\tilde{t}}) \\ -n^1 \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}}) \tilde{b}_\beta^1 & -n^1 \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}}) \tilde{b}_\beta^2 & -n^1 \tilde{n}^1 (\vec{t}.\vec{\tilde{t}}) & -n^1 \tilde{n}^2 (\vec{t}.\vec{\tilde{t}}) \\ -n^2 \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}}) \tilde{b}_\beta^1 & -n^2 \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}}) \tilde{b}_\beta^2 & -n^2 \tilde{n}^1 (\vec{t}.\vec{\tilde{t}}) & -n^2 \tilde{n}^2 (\vec{t}.\vec{\tilde{t}}) \end{bmatrix}$$

$$B_{22} = \begin{bmatrix} \tilde{n}^\alpha \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) \tilde{b}_\alpha^1 \tilde{b}_\beta^1 & \tilde{n}^\alpha \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) \tilde{b}_\alpha^1 \tilde{b}_\beta^2 & \tilde{n}^\alpha \tilde{n}^1 (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) \tilde{b}_\alpha^1 & \tilde{n}^\alpha \tilde{n}^2 (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) \tilde{b}_\alpha^1 \\ \tilde{n}^\alpha \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) \tilde{b}_\alpha^2 \tilde{b}_\beta^1 & \tilde{n}^\alpha \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) \tilde{b}_\alpha^2 \tilde{b}_\beta^2 & \tilde{n}^\alpha \tilde{n}^1 (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) \tilde{b}_\alpha^2 & \tilde{n}^\alpha \tilde{n}^2 (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) \tilde{b}_\alpha^2 \\ \tilde{n}^1 \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) \tilde{b}_\beta^1 & \tilde{n}^1 \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) \tilde{b}_\beta^2 & \tilde{n}^1 \tilde{n}^1 (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) & \tilde{n}^1 \tilde{n}^2 (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) \\ \tilde{n}^2 \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) \tilde{b}_\beta^1 & \tilde{n}^2 \tilde{n}^\beta (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) \tilde{b}_\beta^2 & \tilde{n}^2 \tilde{n}^1 (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) & \tilde{n}^2 \tilde{n}^2 (\vec{t}.\vec{\tilde{t}})(\vec{t}.\vec{\tilde{t}}) \end{bmatrix}$$

$$U_r = [u_1, u_2, u_{3,1}, u_{3,2}], \quad \tilde{U}_r = [\tilde{u}_1, \tilde{u}_2, \tilde{u}_{3,1}, \tilde{u}_{3,2}], \quad V_r = [v_1, v_2, v_{3,1}, v_{3,2}], \quad \tilde{V}_r = [\tilde{v}_1, \tilde{v}_2, \tilde{v}_{3,1}, \tilde{v}_{3,2}]$$

Enfin, la forme linéaire peut être mise sous la forme :

$$l([\vec{v}, \vec{\tilde{v}}]) = \int_\Omega F^T V d\xi_1 d\xi_2 + \int_{\tilde{\Omega}} \tilde{F}^T \tilde{V} d\tilde{\xi}_1 d\tilde{\xi}_2. \tag{3.3.9}$$

Il reste maintenant une dernière matrice à définir. Il s'agit de la matrice C qui contient les contraintes linéaires qui caractérisent l'égalité des déplacements de part et d'autre de la charnière.

$$\int_\gamma (\vec{u} - \vec{\tilde{u}})\, d\gamma = \int_\gamma C[U\ \tilde{U}]\, d\gamma = 0. \tag{3.3.10}$$

Rappelons que, pour résoudre le problème de coque de Koiter, nous devons employer des éléments d'Argyris. La matrice C sera déterminée à partir des degrés de liberté des triangles d'Argyris.

3.3.1.2 Discrétisation des conditions de charnière

Afin d'écrire des égalités de déplacements le long de la charnière, les triangulations des domaines de référence doivent être compatibles. Les égalités sont ensuite écrites sur chacun des nœuds des maillages appartenant à la frontière commune.

Dans le but d'assurer la correspondance des maillages des domaines Ω et $\tilde{\Omega}$, deux fonctions affines par morceaux F et $\tilde{F}$ sont introduites de telle sorte que l'on puisse se ramener à une discrétisation commune (figure 3.8).

Remarque 3.3.1. Pour pouvoir définir les fonctions F et $\tilde{F}$, il est nécessaire que les domaines de référence comptent exactement le même nombre de triangles de part et d'autre de la frontière.

Ces égalités sont traduites sur les degrés de liberté des triangles d'Argyris (voir figure 3.9) :

Proposition 3.3.1. Soient deux fonctions affines F et $\tilde{F}$ définies comme précédemment où $[0,1]$ est divisé en $n+1$ segments $[s_p, s_{p+1}]$, pour $p = 0, ..., n$, avec $s_0 = 0$ et $s_{n+1} = 1$. Ainsi $F(s_p) = a_p$ et $\tilde{F}(s_p) = \tilde{a}_p$ où a_p et $\tilde{a}_p$ sont des sommets de triangles des maillages des domaines de références Ω et $\tilde{\Omega}$ situés sur les frontières γ et $\tilde{\gamma}$. Soient $\{\vec{a}_1, \vec{a}_2, \vec{a}_3\}$ et $\{\vec{\tilde{a}}_1, \vec{\tilde{a}}_2, \vec{\tilde{a}}_3\}$ les bases covariantes pour chaque coque et $\{\vec{a}^1, \vec{a}^2, \vec{a}^3\}$ et $\{\vec{\tilde{a}}^1, \vec{\tilde{a}}^2, \vec{\tilde{a}}^3\}$ les bases contravariantes. Alors, pour $l = 0, ..., n+1$ les conditions de jonction discrètes sur les triangles d'Argyris sont de la forme :

$$\begin{cases} \tilde{u}_i(\tilde{a}_l) = & \vec{\tilde{a}}_i(\tilde{a}_l) \cdot \vec{a}^j(a_l) u_j(a_l), \\ \tilde{u}_{i,\tau}(\tilde{a}_l) = & \dfrac{1}{|D\tilde{F}(s_l)|} \left(\dfrac{d}{ds}(\vec{\tilde{a}}_i(\tilde{a}_l) \cdot \vec{a}^j(a_l)) u_j(a_l) + |DF(s_l)| \vec{\tilde{a}}_i(\tilde{a}_l) \cdot \vec{a}^j(a_l) u_{j,\tau}(a_l) \right), \\ \tilde{u}_{i,\tau\tau}(\tilde{a}_l) = & \dfrac{1}{|D\tilde{F}(s_l)|} \left(\dfrac{d^2}{ds^2}(\vec{\tilde{a}}_i(\tilde{a}_l) \cdot \vec{a}^j(a_l)) u_j(a_l) + 2|DF(s_l)| \dfrac{d}{ds}(\vec{\tilde{a}}_i(\tilde{a}_l) \cdot \vec{a}^j(a_l)) u_{j,\tau}(a_l) \right. \\ & \left. + |DF(s_l)|^2 \vec{\tilde{a}}_i(\tilde{a}_l) \cdot \vec{a}^j(a_l) u_{j,\tau\tau}(a_l) \right). \end{cases} \tag{3.3.11}$$

où $u_{,\tau}$ et $u_{,\tau\tau}$ sont les dérivées premières et secondes tangentielles.

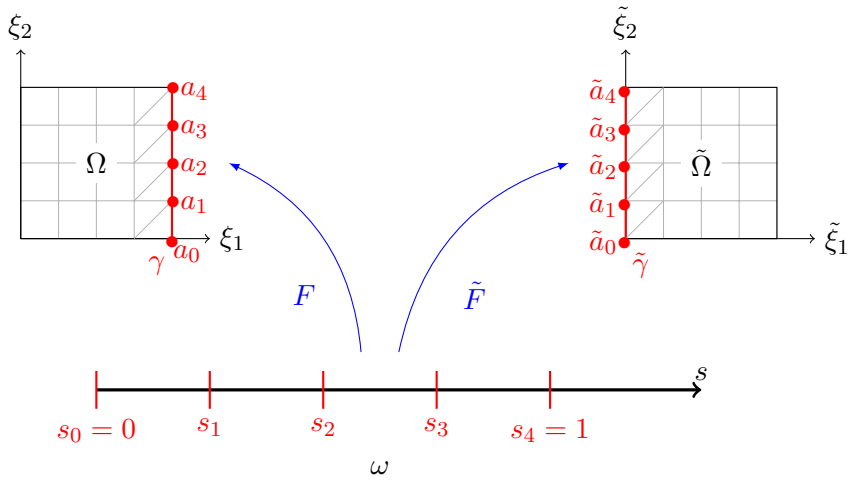


FIGURE 3.8: Discrétisation des domaines de référence : définition des fonctions affines par morceaux F et $\tilde{F}$.

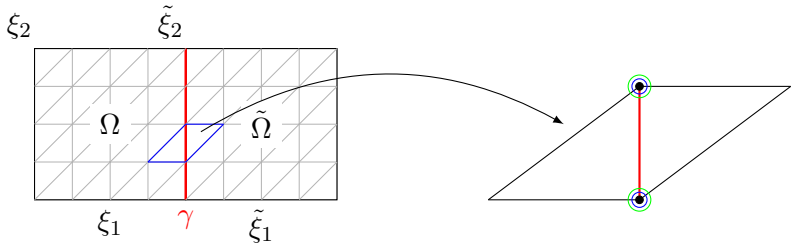


FIGURE 3.9: Conditions de jonctions sur un triangle d'Argyris (valeurs aux nœuds, premières dérivées et dérivées secondes aux sommets du triangle).

Démonstration. La condition de continuité des déplacements le long de la charnière est donnée par

$$\vec{u}(\xi) = \tilde{\vec{u}}(\tilde{\xi}), \quad \forall(\xi, \tilde{\xi}) \in \gamma \times \tilde{\gamma} \quad \text{tel que} \quad \vec{\varphi}(\xi) = \tilde{\vec{\varphi}}(\tilde{\xi}).$$

En écrivant cette égalité à l'aide des vecteurs des bases contravariantes, il vient :

$$\tilde{u}_i(\tilde{\xi})\tilde{\vec{a}}^i(\tilde{\xi}) = u_j(\xi)\vec{a}^j(\xi) \Rightarrow \tilde{u}_i(\tilde{\xi}) = \tilde{\vec{a}}_i(\tilde{\xi}) \cdot \vec{a}^j(\xi)u_j(\xi).$$

Puis en introduisant la discrétisation commune et les fonctions affines F et $\tilde{F}$, on a :

$$\tilde{u}_i \circ \tilde{F}(s_l) = \tilde{\vec{a}}_i(\tilde{F}(s_l)) \cdot \vec{a}^j(F(s_l))u_j \circ F(s_l) \quad \text{avec} \quad F(s_l) = a_l \text{ et } \tilde{F}(s_l) = \tilde{a}_l.$$

Compte tenu des degrés de liberté pour un triangle d'Argyris, il faut imposer les conditions suivantes :

$$\begin{cases} \tilde{u}_i \circ \tilde{F}(s_l) = \tilde{\vec{a}}_i(\tilde{a}_l) \cdot \vec{a}^j(a_l)u_j \circ F(s_l) \\ \dfrac{d}{ds}\left(\tilde{u}_i \circ \tilde{F}\right)(s_l) = \dfrac{d}{ds}\left(\tilde{\vec{a}}_i(\tilde{a}_l) \cdot \vec{a}^j(a_l)u_j \circ F\right)(s_l) \\ \dfrac{d^2}{ds^2}\left(\tilde{u}_i \circ \tilde{F}\right)(s_l) = \dfrac{d^2}{ds^2}\left(\tilde{\vec{a}}_i(\tilde{a}_l) \cdot \vec{a}^j(a_l)u_j \circ F\right)(s_l) \end{cases}$$

Soit $\vec{\tau}$ le vecteur tangent unitaire à γ donné par :

$$DF(s_l) = |DF(s_l)|\vec{\tau}(s_l).$$

Sachant que,

$$\frac{d}{ds}\Big(u_i \circ F\Big)(s_l) = Du_i(F(s_l)) \cdot DF(s_l)$$

on a alors :

$$\frac{d}{ds}\Big(u_i \circ F\Big)(s_l) = Du_i \cdot DF(s_l) = Du_i \cdot |DF(s_l)|\vec{\tau}(s_l) = |DF(s_l)|u_{i,\tau}(a_l).$$

Avec un raisonnement similaire, on obtient :

$$\frac{d^2}{ds^2}\Big(u_i \circ F\Big)(s_l) = |DF(s_l)|^2 u_{i,\tau\tau}(a_l).$$

En développant les produits de dérivées, on retrouve alors les équations (3.3.11). □

Remarque 3.3.2. Dans notre cas, les domaines de référence étant des carrés, les dérivées tangentielles correspondent aux dérivées selon ξ_1 ou ξ_2 en fonction du côté considéré.

En pratique, pour gérer la compatibilité des triangulations, le maillage des domaines de référence est construit en imposant une discrétisation par arête. En d'autres termes, à chaque arête un nombre de nœuds est imposé et de cette façon le nombre d'éléments de part et d'autre des arêtes représentant une charnière, est le même. Avec cette technique, les patchs dit non conformes, qui ne partagent pas une arête entière, peuvent être traités sans travail supplémentaire. Il suffit de repérer la longueur de l'arête sur les domaines référence et d'imposer la discrétisation en conséquence.

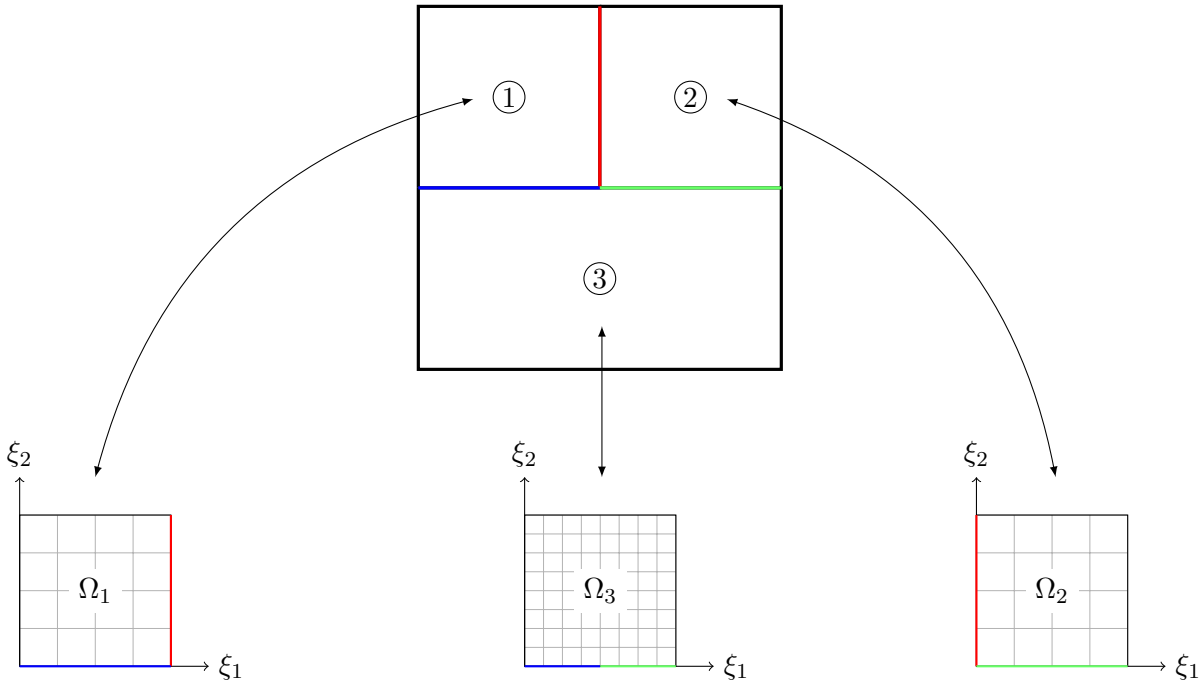


Figure 3.10: Exemple de discrétisation de patchs non conformes.

3.4 Présentation du code de calcul OptSurf

L'analyse isogéométrique est apparue il y a environ une dizaine d'années. Quelques codes sont déjà disponibles et permettent de réaliser des simulations. Dans le cadre de notre problème, ces codes de calcul ne sont pas adaptés. Pour répondre aux besoins de notre méthode, un nouveau code de calcul, nommé OptSurf, a été développé. Ce programme a été implémenté à partir de plusieurs librairies open source. Afin d'opérer une simulation, cinq étapes sont essentielles. Nous détaillons chacune d'entre elles dans les paragraphes qui suivent.

3.4.1 Étape 1 : lecture du modèle CAO

L'intérêt de l'analyse isogéométrique est d'exploiter la géométrie de la forme pour faire les calculs. La première étape consiste donc en la lecture du modèle CAO et en l'extraction des informations nécessaires. En terme de format d'échange de modèle CAO, le format STEP a été retenu. Il s'agit d'un format universel lu par la majorité des modeleurs. La gestion complète du modèle CAO (repérage des arêtes, des faces etc.) est assuré par la libraire C++ OpenCascade Technology (OCCT) [OpenCascade, 2015]. La librairie OCCT développée par la société OpenCascade, est une librairie permettant la création et l'exploitation des modèles CAO. Elle est par exemple utilisée au sein de logiciels comme le mailleur Gmsh ou le modeleur Salome. Elle intègre un interpréteur STEP permettant de lire la CAO pour OptSurf. Une fois le modèle CAO chargé, les différentes faces qui le composent sont transformées en objets `Geom_BSplineSurface` qui correspondent à des surfaces NURBS. Après avoir créé ces objets, toutes les informations liées aux surfaces sont disponibles. On peut alors déterminer l'entité géométrique de base pour l'écriture du problème de coque en coordonnées curvilignes : la base covariante. La fonction membre `D1` de la classe `Geom_BSplineSurface` permet de calculer la dérivée de la surface aux points (ξ_1, ξ_2) par rapport aux coordonnées curvilignes. Une fois la base covariante déterminée, les autres entités nécessaires (base contravariante, tenseur covariant, première forme fondamentale etc.) sont déterminées à l'aide des relations rappelées au chapitre 2. Les définitions des différentes surfaces sont stockées au sein d'un objet de type `Shape`. Les objets `Shape` constituent la base de OptSurf. Les appels à OCCT n'interviennent pas uniquement au début du processus de simulation. En effet, chaque fois qu'une information liée à la surface est nécessaire, c'est au travers de cette librairie qu'elle est obtenue. L'un des principaux avantages à utiliser OCCT est qu'elle permet de considérer n'importe quel type de surfaces (Bézier, B-spline ou NURBS) et offre donc une certaine liberté pour la modélisation des CAO.

3.4.2 Étape 2 : maillage des domaines de référence

Avec notre approche, les calculs sont effectués sur un domaine de référence Ω. À chaque patch CAO est attribué un domaine de référence propre. Ces domaines sont dans tous les cas le carré $[0,1]^2$. À l'aide de OCCT, les numéros attribués aux arêtes du modèle CAO sont reportés sur les domaines de référence. En pratique, OCCT est capable de déterminer les coordonnées curvilignes associées aux extrémités des arêtes. Ces coordonnées sont ensuite ramenées sur le domaine de référence (comme sur la figure 3.10). Une discrétisation par arête pourra être ainsi imposée et facilitera la gestion des patchs non conformes. Une fois les extrémités de chaque arête connues, un fichier de commande pour Gmsh est généré [Guezaine et Remacle, 2014]. Dans ce fichier, le carré $[0,1]^2$, avec le même nombre et la même position des arêtes du patch associé, est défini. Chaque arête sera ensuite discrétisée avec le nombre de points choisi. Le logiciel Gmsh est alors lancé en ligne de commande et génère les différents maillages. La figure 3.11 décrit la procédure.
Les maillages générés sont ensuite relus, stockés dans un tableau d'objets de type `Mesh` qui est un attribut de la classe `Shape`.

3.4.3 Étape 3 : assemblage des matrices

À partir du maillage crée, les différentes matrices peuvent être exprimées. Étant donné qu'il s'agit de matrices creuses, elles seront stockées dans un format sparse. Les fonctions de formes de l'élément d'Argyris sont obtenues à l'aide du logiciel de calcul formel Maxima. Celui-ci offre la possibilité de générer du code fortran. Les matrices élémentaires sont alors construites en Fortran 90 en suivant la procédure donnée dans [Bernadou, 1994]. Les numéros de ligne et de colonne dans la matrice globale sont ensuite déterminés et les matrices sont stockées dans trois tableaux : indices de la ligne, indices de la colonne et valeurs. Le raccordement est ensuite traité. Toutes les jonctions sont parcourues de la même manière. Les nœuds de chacun des patchs de part et d'autre d'une jonction sont repérés et associés deux à deux. Après ceci, les deux matrices, l'une pour la forme bilinéaire de la charnière et l'autre pour les contraintes linéaires sur les déplacements, sont assemblées. Elles sont ensuite ajoutées

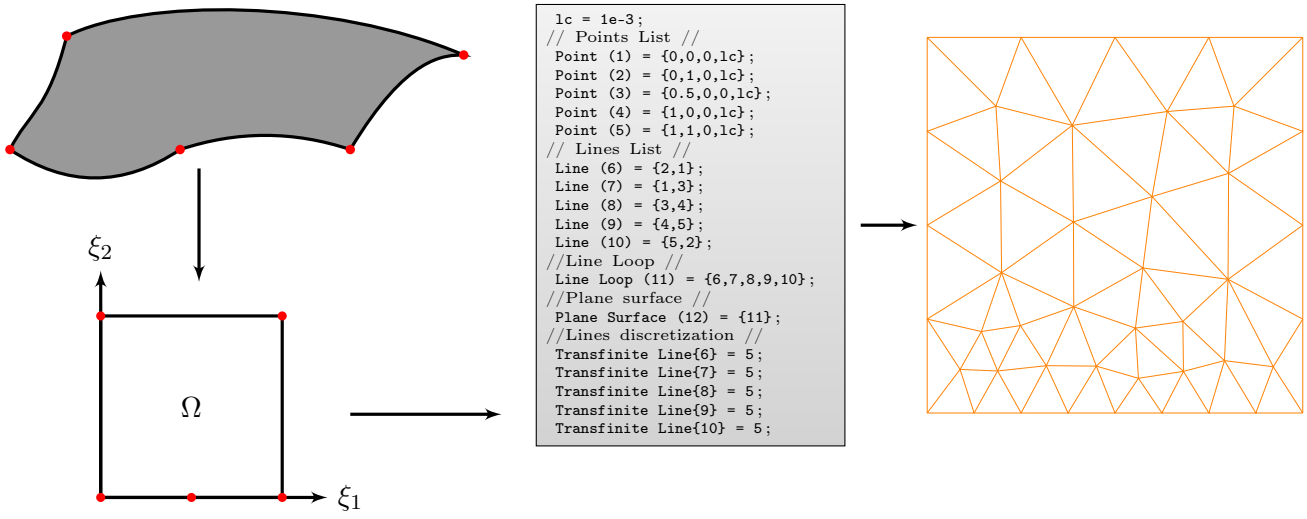


Figure 3.11: Procédure de maillage des domaines de référence.

aux trois tableaux. Avec l'aide de la libraire Csparse, les trois tableaux sont « compressés » c'est-à-dire que les valeurs des indices répétés sont sommées pour diminuer la taille des tableaux. Cette opération terminée, les matrices sont enregistrées au format sparse COO dans un objet `COOMatrix`. Les matrices de masse et de raideur sont des matrices symétriques définies positives.

Les conditions aux limites sont représentées par l'objet `CondLim`. Ces objets ont pour attributs le numéro de l'arête où les conditions aux limites doivent être appliquées et un tableau de 21 entrées. Le tableau représente les degrés de liberté du triangle d'Argyris. On associe 0 si le degré n'est pas concerné par la condition aux limites et 1 s'il l'est. Trois types de conditions sont pré-implémentés : libre, simplement posé et encastré. La fonction `ApplyCondLim` génère un tableau contenant l'ensemble des degrés de liberté de la matrice globale concernés par les conditions aux limites. Les conditions aux limites sont appliquées en déplaçant les lignes et les colonnes en fin de matrice. Les calculs sont alors faits sur une matrice réduite (voir figure 3.12).

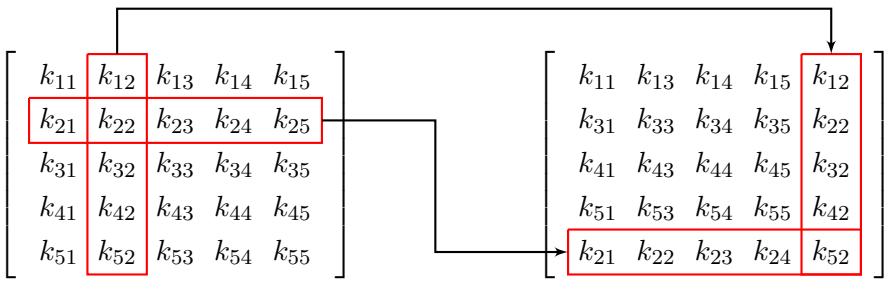


Figure 3.12: Exemple d'application des conditions aux limites.

Le second membre du problème suit une procédure identique aussi bien pour l'assemblage que pour les conditions aux limites.

3.4.4 Étape 4 : résolution du problème de coques

Nous allons voir maintenant comment deux types de problèmes sont résolus dans OptSurf : le problème en mécanique statique et celui en analyse modale. Ces deux problèmes feront l'objet des cha-

pitres 4 et 5 respectivement.

Cas 1 : problème en mécanique statique

Le problème en mécanique statique correspond au problème (3.2.11) pour un seul patch et à l'équation (3.2.7) dans le cas de plusieurs patchs. Prenons l'exemple d'une structure composée de deux patchs (la généralisation à plus de deux patchs étant immédiate). En mécanique statique, on considère le problème :

$$KU = F \quad \text{avec} \quad CU = 0. \tag{3.4.1}$$

où :

- K est la matrice de rigidité telle que :

$$K = \begin{bmatrix} K_1 + B_1 & 0 \\ 0 & K_2 + B_2 \end{bmatrix} \tag{3.4.2}$$

  où K_1 et K_2 correspondent aux matrices de rigidité associées à chaque coque, B_1 et B_2 sont les contributions issues de la charnière ;

- $U = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$ les déplacements sur les coques ;
- $F = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}$ les forces appliquées ;
- $C = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}$ les contraintes linéaires entre les deux coques.

En règle générale, on résout ce type de système en introduisant la notion de multiplicateurs de Lagrange. Le système à résoudre est alors de la forme :

$$\begin{bmatrix} K_1 + B_1 & 0 & C_1 \\ 0 & K_2 + B_2 & C_2 \\ C_1 & C_2 & 0 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ 0 \end{bmatrix} \tag{3.4.3}$$

Le système de l'équation (3.4.3) est résolu avec le solveur MUMPS [Mumps, 2016]. MUMPS est un solveur direct qui résout les problèmes en MPI en format sparse à l'aide de décompositions LU ou de Cholesky.

Il faut noter que ce système est mal conditionné. Il faut parfois raffiner davantage un maillage afin d'obtenir une convergence acceptable du système. Pour faire face à ce genre de situation, il est souvent recommandé de faire appel à une méthode d'élimination des multiplicateurs de Lagrange comme celle présentée dans [Pellet, 2011] :

Méthode numérique 3.4.1 (méthode d'élimination des multiplicateurs de Lagrange)**.**
Soient :

- K la matrice de rigidité du système mécanique sans contrainte ;
- U les déplacements nodaux ;
- F le vecteur des forces appliquées ;
- C, la matrice $(p \times n)$, des contraintes linéaires.

On considère le problème :

$$KU = F \quad \text{avec} \quad CU = 0$$

Si C est de rang p alors, il existe deux sous-matrices C_1 et C_2 et une partition de U telles que :

$$CU = \begin{bmatrix} C1 & C_2 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = C_1U_1 + C_2U_2 = 0$$

avec : $U_1 \in \mathbb{R}^p$, $U_2 \in \mathbb{R}^{n-p}$, C_1 de taille $p \times p$ inversible.
Ainsi, il vient :

$$U_1 = -C_1^{-1}C_2U_2$$

En partionnant K et F de façon similaire à U :

$$K = \begin{bmatrix} K_{11} & K_{12} \\ K_{12}^T & K_{22} \end{bmatrix} \quad F = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}$$

On note $\overline{K} = K_2 + C_2^TC_1^{-T}K_1C_1^{-1}C_2 - K_{12}C_1^{-1}C_2 - C_2^TC_1^{-T}K_{12}^T$
et $\overline{F} = F_2 - C_1^{-1}C_2F_1$

Résoudre le problème $\overline{K}U_2 = \overline{F}$ est alors équivalent à résoudre le problème initial.

La méthode d'élimination de Lagrange permet donc de déterminer la solution du problème à partir d'un système réduit. Cette méthode est équivalente à choisir ce que l'on appelle des nœuds « maîtres » (les nœuds U_2) et des nœuds « esclaves » (les nœuds U_1). Les « esclaves » sont éliminés du système et leur valeur est retrouvée en fonction des « maîtres ». Cette approche n'a pas été implémentée dans OptSurf. Lorsqu'un modèle CAO est composé de nombreux patchs, il existe très souvent des *crosspoints*, c'est-à-dire des points qui appartiennent à au moins trois patchs différents (voir figure 3.13).

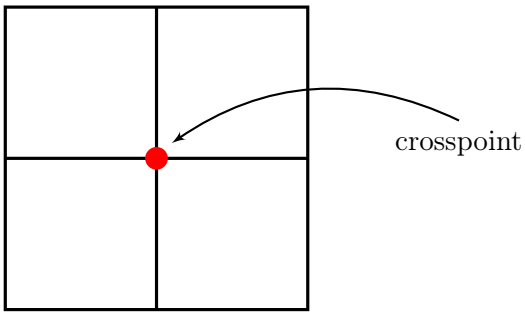


Figure 3.13: Exemple d'un crosspoint commun à quatre patchs.

Dans cette configuration, le choix des nœuds « maîtres » et « esclaves » n'est pas évident et il a donc été choisi de conserver la résolution directe du système.

Cas 2 : problème en analyse modale

L'autre type de problème qui occupe une place importante au sein de cette étude est l'analyse modale (voir chapitre 5). L'objectif est de déterminer les fréquences propres et modes de déformation associés. À cette fin, il faut résoudre les équations de vibration libre qui découlent du problème dynamique. Ces équations décrivent le comportement de la coque dû à des vibrations libres de tout effet de charge extérieure. Il s'agit alors de résoudre le problème aux valeurs propres suivant :

$$\text{trouver } (\lambda, \vec{u}) \in \mathbb{R}^+ \times W, \quad \forall \vec{v} \in W, \quad \text{tel que } a(\vec{u}, \vec{v}) = \lambda b(\vec{u}, \vec{v}), \tag{3.4.4}$$

où

- W est donnée par l'équation (3.2.15) ;
- $a(\vec{u}, \vec{v})$ est donnée par l'équation (3.2.8) ;

et

$$\begin{aligned} b(\vec{u}, \vec{v}) &= \int_\Omega \rho e\{[1 + \frac{e^2}{12}(b_1^1 b_2^2 - b_1^2 b_2^1)][a^{\alpha\beta} u_\alpha v_\beta + u_3 v_3] \\ &+ \frac{e^2}{12} a^{\alpha\beta}[(u_{3|\alpha} + b_\alpha^\lambda u_\lambda)(v_{3|\beta} + b_\beta^\mu v_\mu) \\ &+ (u_\alpha v_{3|\beta} + u_{3|\alpha} v_\beta + 2 b_\alpha^\lambda u_\lambda v_\beta) b_\eta^\eta]\} \sqrt{a}\, d\xi_1 d\xi_2. \end{aligned} \tag{3.4.5}$$

Le problème est écrit ici pour une seule coque mais s'exprime pour plusieurs coques de la même façon que pour le problème statique. Sous forme matricielle, on cherche à résoudre :

$$KU = \lambda MU \quad \text{tel que} \quad CU = 0 \tag{3.4.6}$$

avec :

- K la matrice de rigidité : $K = \begin{bmatrix} K_1 + B_1 & 0 \\ 0 & K_2 + B_2 \end{bmatrix}$;
- M la matrice de masse : $M = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix}$;
- U le mode propre associés à la valeur propre λ.

L'équation (3.4.6) ne peut pas s'écrire de façon similaire à l'équation (3.4.3) car sous cette forme, le problème n'admettrait aucune solution. Dans cette situation, la méthode des multiplicateurs de Lagrange est souvent employée. Pour la raison citée plus haut, à savoir le problème du choix des nœuds « maîtres » et « esclaves », une autre alternative a été choisie. Il a été implémenté dans OptSurf une méthode qui consiste à projeter les matrices de rigidité et masse dans le noyau de la matrice des contraintes linéaires. La procédure avancée dans [Golub, 1973] est détaillée ici.

Méthode numérique 3.4.2 (méthode de projection dans le noyau pour le calcul de valeurs propres)**.** Soient :

- K une matrice de dimension $n \times n$ réelle symétrique ;
- M une matrice de dimension $n \times n$ réelle symétrique définie positive ;
- $U \in \mathbb{R}^n$ un vecteur propre associé à la valeur propre λ ;
- C une matrice de dimension $n \times p$ et de rang r.

On considère le problème :

$$KU = \lambda MU \quad \text{tel que} \quad CU = 0.$$

On effectue une factorisation RRQR (*Rank Revealing QR*) de C :

$$C = Q^T \begin{bmatrix} R & S \\ 0 & 0 \end{bmatrix} \Pi$$

où R est une matrice triangulaire supérieure de dimension r, S est de dimension $r \times (p - r)$, Q est une matrice orthogonale de dimension n telle que $Q^T Q = Id_n$ (avec Id l'identité) et Π est une matrice de permutation.

Soient G et H deux matrices de taille $n \times n$ telles que :

$$G = QKQ^T = \begin{bmatrix} G_{11} & G_{12} \\ G_{12}^T & G_{22} \end{bmatrix}, \qquad H = QMQ^T = \begin{bmatrix} H_{11} & H_{12} \\ H_{12}^T & H_{22} \end{bmatrix}.$$

où G_{11} et H_{11} sont des matrices de dimension $r \times r$ et les matrices G_{22} et H_{22} sont de dimension $(n-r) \times (n-r)$.

Les valeurs propres du problème réduit :

$$\boxed{G_{22}Z = \lambda H_{22}Z}$$

sont les mêmes que celles du problème original. Les vecteurs propres sont ensuite donnés par :

$$\boxed{U = Q^T \begin{bmatrix} 0 \\ \vdots \\ Id_{n-r} \end{bmatrix} Z.}$$

Le problème aux valeurs propres réduit est résolu avec l'algorithme d'Arnoldi de la librairie ARPACK [Lehoucq *et al.*, 1997].

3.4.5 Étape 5 : le post-traitement

L'étape finale est le post-traitement des résultats de simulation. Rappelons que les calculs sont effectués sur le domaine de référence Ω qui est indépendant de la forme. La solution trouvée est donc exprimée dans les bases locales et il faut transférer les informations sur la surface en dimension 3. Une approximation du déplacement de la coque $\overrightarrow{\mathscr{U}}$ en fonction du déplacement de la surface moyenne $\vec{u}$ est donnée par l'équation (3.4.7) :

$$\overrightarrow{\mathscr{U}} = u_i \vec{a}^i - \xi_3 (u_{3|\alpha} + b_\alpha^\lambda u_\lambda) \vec{a}^\alpha. \tag{3.4.7}$$

Les résultats sont ensuite visualisés par le biais d'une routine basée sur la librairie OpenGL. Les déformations, résultant d'un calcul statique ou modal, peuvent être affichées ainsi que le déplacement pour les points désirés par l'utilisateur.

3.4.6 Bilan sur le code OptSurf

Le code OptSurf développé pour l'analyse isogéométrique est basé sur plusieurs librairies et a donc été implémenté à partir de différents langages (C,C++ et Fortran 90). La figure 3.14 résume son organisation.

Des simulations aussi bien en calcul statique que modal peuvent être réalisées. Pour faciliter l'utilisation d'OptSurf, les simulations sont lancées à partir de fichiers de commande. Des exemples de ces fichiers et des extraits du code peuvent être consultés en annexe A.
La section suivante présente des résultats de simulation obtenus avec OptSurf. Ces calculs sont des benchmarks ayant pour but de valider le code.

3.5 Résultats de quelques benchmarks

Pour la validation d'OptSurf, deux types de tests ont été effectués. Le premier concerne la vérification des modes de corps rigide. La seconde batterie de tests consiste en des comparaisons avec des problèmes pour lesquels une solution théorique ou numérique est connue.

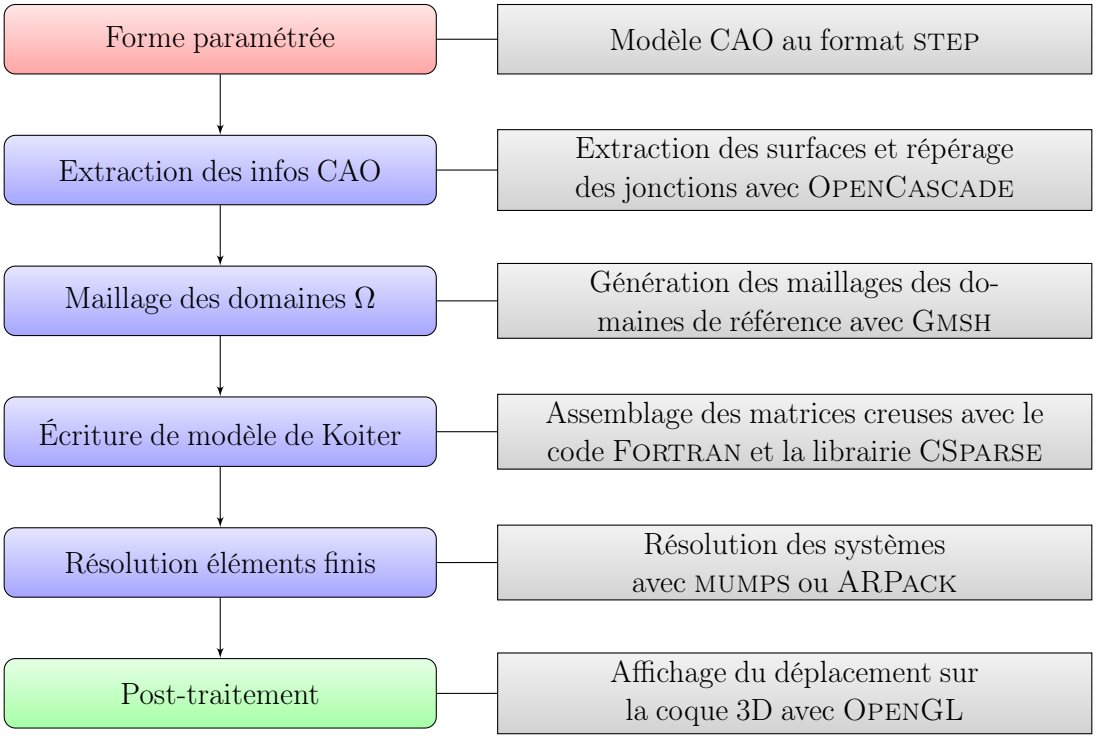


Figure 3.14: Organisation du code de calcul OptSurf.

3.5.1 Identification des modes de corps rigide

En mécanique, les modes de corps rigide correspondent à des modes propres pour lesquels la structure se déplace sans déformation. On compte six modes de corps rigide : trois translations (une dans chaque direction) et trois rotations (une autour de chaque axe du repère cartésien). Lors d'un calcul modal sans condition aux limites ni excitation extérieure, les modes de corps rigide peuvent être repérés car ils sont des modes à énergie nulle et de fait, les valeurs propres associées sont nulles. Leur présence permet de se rendre compte de la cohérence de la structure. En effet, lors de calculs sur des structures comportant plusieurs patchs, il faut s'assurer que les jonctions soient bien prises en compte et que l'ensemble de la structure soit bien considéré comme un seul solide.

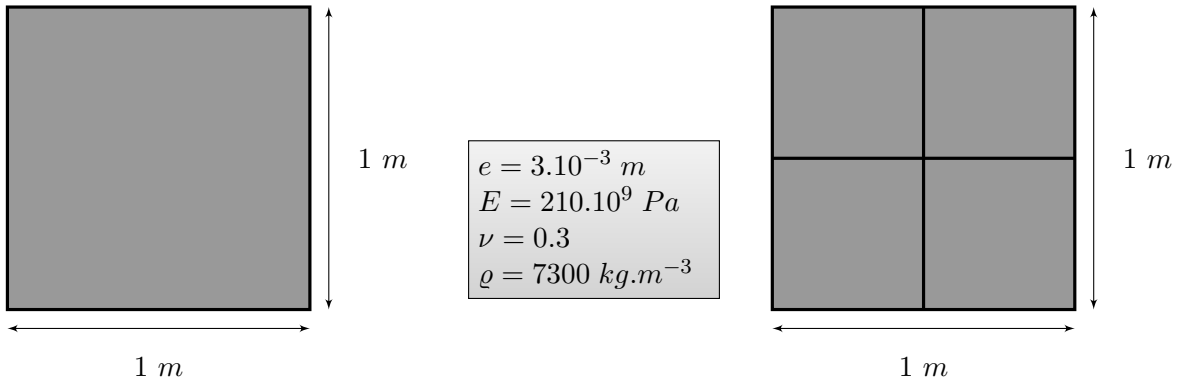
3.5.1.1 Le cas de la plaque carrée

Le premier cas testé est une plaque carré de côté $a = 1\ m$ d'épaisseur $e = 3\ mm$, de masse volumique $\rho = 7300\ kg.m^{-3}$, dont le matériau a pour module de Young $E = 210.10^9\ Pa$ et un coefficient de Poisson égal à $\nu = 0.3$. En plus de vérifier la présence des modes de corps rigide, nous comparons les fréquences propres calculées avec la solution analytique donnée par l'équation (3.5.1).

$$f_i = \frac{1}{2\pi a^2}\lambda_i^2\sqrt{\frac{Ee^2}{12\varrho(1-\nu^2)}}, \quad \text{avec} \quad \begin{array}{|c|c|} \hline i & \lambda_i^2 \\ \hline 1,\ldots,6 & 0 \\ 7 & 13.49 \\ 8 & 19.79 \\ 9 & 24.43 \\ 10 & 35.02 \\ 11 & 35.02 \\ \hline \end{array} \tag{3.5.1}$$

Deux modélisations sont considérées. Dans la première, la plaque est représentée avec un seul patch NURBS alors que la seconde en compte quatre disposés selon la figure 3.15. Le tableau 3.1 présente les résultats obtenus. Les figures 3.16 et 3.17 montrent les trois premières déformées modales pour les

deux modélisations.



(a) Configuration avec 1 patch (b) Configuration avec 4 patchs

FIGURE 3.15: Géométrie de la plaque avec un et quatre patchs.

Mode	Théorique	Modélisation 1 patch	Erreur relative	Modélisation 4 patchs	Erreur relative
1	0	2.5690E-006	<1E-3%	1.6551E-005	<1E-3%
2	0	4.1178E-006	<1E-3%	1.0357E-004	<1E-3%
3	0	5.3717E-006	<1E-3%	1.3851E-004	<1E-3%
4	0	6.3025E-005	<1E-3%	1.6262E-004	<1E-3%
5	0	8.4530E-005	<1E-3%	2.4007E-004	<1E-3%
6	0	1.3840E-004	<1E-3%	3.5281E-004	<1E-3%
7	10.4541	10.4372	0.16%	10.4372	0.16%
8	15.3364	15.1859	0.98%	15.1859	0.98%
9	18.9322	18.8081	0.65%	18.8080	0.65%
10	27.1390	26.9686	0.63%	26.9686	0.63%
11	27.1390	26.9686	0.63%	26.9886	0.63%

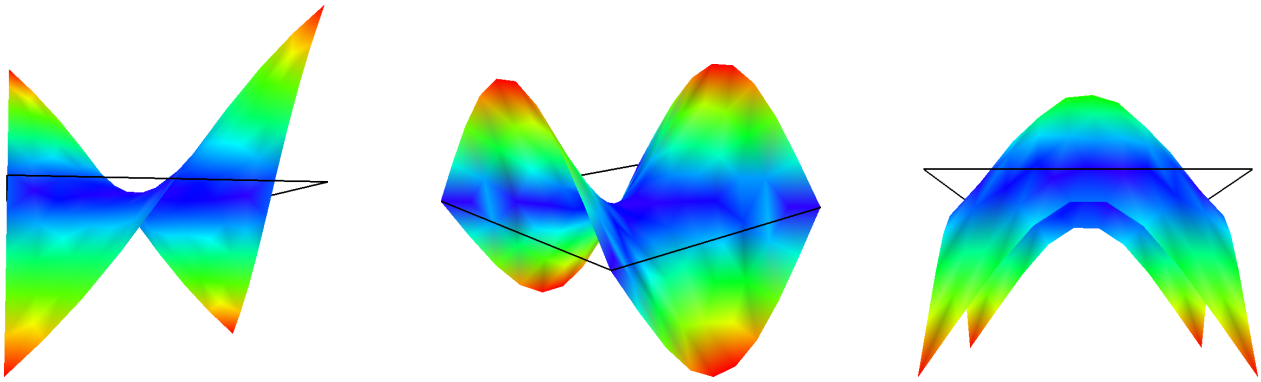
TABLEAU 3.1: Comparaison des fréquences propres (en Hz) entre les valeurs théoriques et les résultats de OPTSURF.

Les calculs ont été faits avec 198 éléments de discrétisation soit 3069 degrés de liberté pour la modélisation avec un seul patch et 792 éléments de discrétisation soit 11919 degrés de liberté pour les quatre patchs. Dans les deux cas, les six modes de corps rigide sont présents. Les déformées modales pour le cas avec un patch et celui avec quatre patchs sont semblables. Comme indiqué dans le tableau 3.1, les fréquences sont également les mêmes. Les résultats ne sont pas exactement identiques aux valeurs théoriques mais ils montrent une certaine cohérence vis à vis des géométries.

3.5.1.2 Deux plaques perpendiculaires

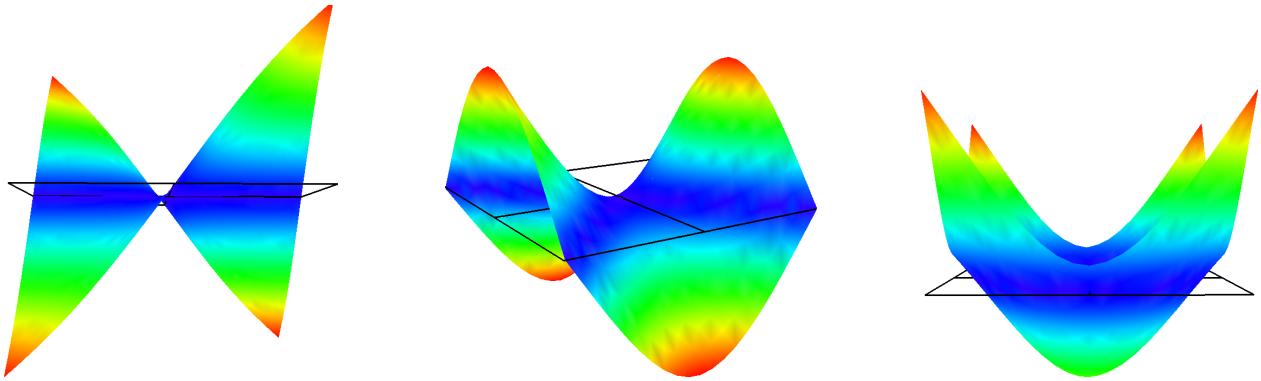
L'une des raisons majeures d'avoir implémenté le raccordement entre patchs CAO sous la forme de jonction entre coques est la prise en compte des structures avec discontinuité de courbure. Le cas test suivant a pour but de valider cette approche. On considère deux plaques perpendiculaires, chacune de dimension 1 $m \times 1\ m$. La figure 3.18a donne la configuration ainsi que les paramètres du problème étudié.
Les fréquences propres calculées avec OPTSUF sont comparées à celles obtenues avec le logiciel CODE_ASTER dans le tableau 3.3.

Dans ce cas test, les six modes de corps rigide sont également retrouvés. Ici aussi, le nombre de degrés de liberté utilisé est important. 396 éléments discrétisation ont été utilisés soit 6048 degrés de liberté.



(a) Mode 7 (b) Mode 8 (c) Mode 9

FIGURE 3.16: Déformées modales de la plaque avec un patch pour les modes 7,8 et 9.



(a) Mode 7 (b) Mode 8 (c) Mode 9

FIGURE 3.17: Déformées modales de la plaque avec quatre patchs pour les modes 7,8 et 9.

Pour s'assurer d'avoir un résultat convergé, le calcul avec CODE_ASTER compte 79056 degrés de liberté. Le résultat est ici plus mitigé. Une partie des modes semble bien déterminée alors que, pour d'autres, l'écart par rapport aux résultats avec CODE_ASTER est important.

3.5.1.3 Cylindre

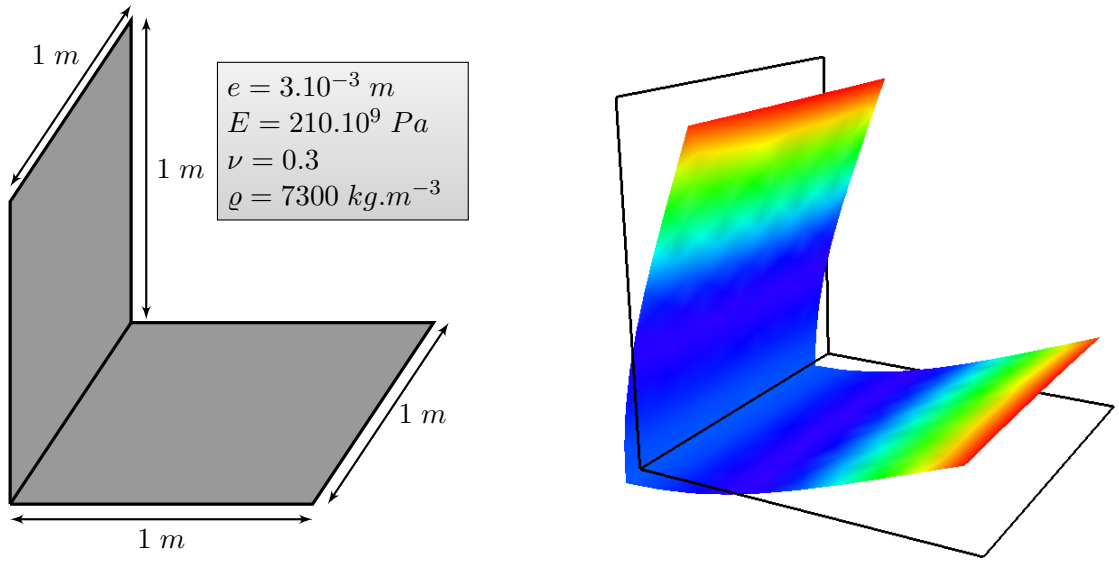
Pour ce cas, nous considérons un cylindre dont la géométrie et les paramètres sont donnés par la figure 3.19a. Les fréquences propres sont comparées avec les résultats issus d'un calcul réalisé avec CODE_ASTER pour 176568 degrés de liberté. Le tableau 3.4 présente les résultats obtenus, quasiment confondus entre les deux codes :

3.5.2 Cas test classiques de déformation de coques

Il existe plusieurs benchmarks ou cas test pour valider l'implémentation des problèmes de coques. Nous en présentons trois exemples ici. Dans la plupart des cas, deux modélisations sont proposées : l'une avec un seul patch et l'autre avec plusieurs patchs.

3.5.2.1 Plaque carrée soumise à une pression uniforme

On considère ici le cas d'une plaque carrée simplement posée sur ses quatre côtés de longueur $a = 1\ m$ soumise à une pression uniformément répartie de valeur $P = 800\ Pa$. Les deux géométries utilisées sont composées d'un seul patch pour la première et de trois patchs pour la seconde (figure 3.20). Le



(a) Géométrie du problème (b) Mode 7 de déformation

FIGURE 3.18: Géométrie des deux plaques perpendiculaires et 7ième déformée modale.

Mode	CODE_ASTER	OPTSURF	Erreur relative
7	3.22483	3.225051	<1E-3%
8	5.52678	6.901850	24.9%
9	7.50372	7.503457	0.27%
10	12.2167	12.27095	0.44%
11	17.6148	17.61515	<1E-3%
12	19.8538	20.74649	4.50%
13	20.1845	21.24528	5.26%

TABLEAU 3.2: Comparaison des fréquences propres (en Hz) des deux plaques perpendiculaires.

deuxième cas d'étude permet de mettre à l'épreuve la gestion des patchs non conformes.

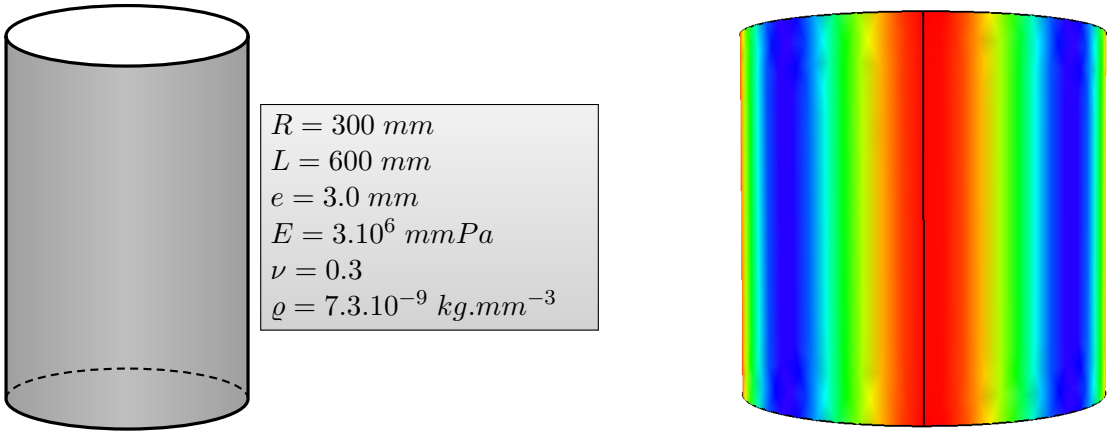
La valeur cible est le déplacement maximal au centre de la plaque. Il est donné par l'équation suivante [Timošenko et Woinowsky-Krieger, 1993] :

$$w_{max} = 0.004062 \times \frac{Pa^4}{D} \quad \text{où} \quad D = \frac{Ee^3}{12(1-\nu^2)} \tag{3.5.2}$$

Les figures 3.21a et 3.21b représentent les déplacements respectifs obtenus. La figure 3.22 donne l'évolution de la valeur cible en fonction de la discrétisation. La discrétisation est donnée en nombre de nœuds par arête. Le tableau 3.4 donne une correspondance entre le nombre de nœuds par arête, le nombre d'éléments du maillage et le nombre de degrés de liberté.
La déformation de la plaque est similaire dans le cas avec un seul patch et celui avec trois patchs. À noter cependant que la convergence du résultat vers la valeur de référence est plus rapide pour la modélisation avec un seul patch. Elle est un peu plus lente pour le cas avec trois patchs. Les deux modélisations atteignent toutefois la même valeur cible (voir figure 3.22).

3.5.2.2 Le paraboloïde hyperbolique sous pression

Le cas test suivant est classique pour la validation des calculs de déformations de coques. On considère un paraboiloïde hyperbolique encastré sur son bord et soumis à une pression uniforme q. La valeur de référence est le déplacement au centre O suivant l'axe z qui vaut, en théorie, $-0.024\ cm$ [Blouza *et al.*, 2006]. Deux géométries sont testées, avec un seul ou avec quatre patchs voir (figure 3.23).



(a) Géométrie du problème (b) Mode 7 de déformation

FIGURE 3.19: Géométrie du cylindre et 7ème déformée modale.

Mode	CODE_ASTER	OPTSURF	Erreur relative
7	2.75227	2.751822	0.12%
8	2.75232	2.751823	0.02%
9	3.68129	3.674580	0.18%
10	3.68157	3.674813	0.18%
11	7.78455	7.783209	0.02%
12	7.78458	7.783209	0.02%
13	9.26512	9.261658	0.04%

TABLEAU 3.3: Comparaison des fréquences propres (en Hz) du cylindre.

La figure 3.24 offre une comparaison des déplacements entre les deux configurations pour ce cas test tandis que la figure 3.25 donne la convergence de la valeur cible. Dans les deux cas, le résultat converge vers une valeur légèrement inférieure à la valeur théorique. À noter que le problème est traité ici en entier alors que dans la littérature, seul un quart de la géométrie est considéré et des conditions de symétrie sont appliquées. L'absence de ces conditions de symétrie peut être une explication à l'écart constaté entre le résultat fourni par OPTSURF et celui de référence.

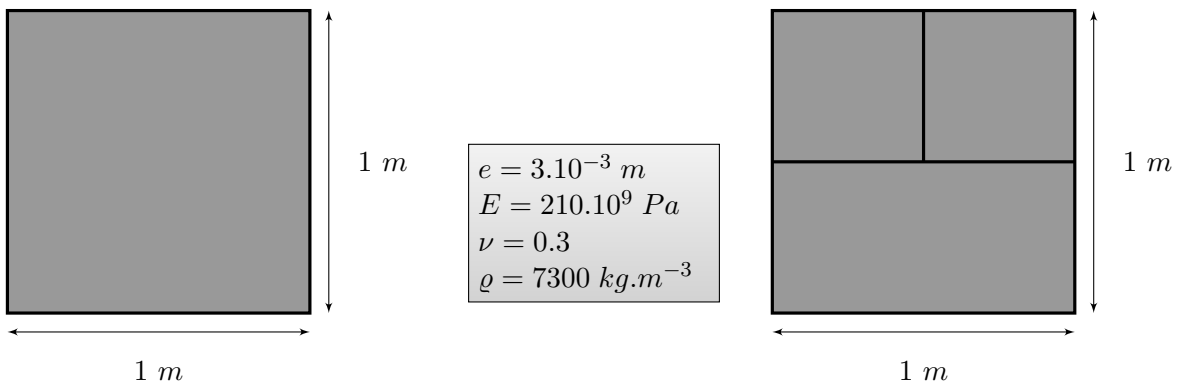
3.5.2.3 La toiture de Scordelis-Lo

Le problème de la toiture de Scordelis-Lo est un benchmark très répandu dans la littérature [Macneal et Harder, 1985]. Ce cas test consiste à vérifier le déplacement vertical d'un point d'une toiture cylindrique soumise à un chargement q. La structure est supportée au niveau de ses côtés courbés. La valeur de référence de 0.3024 *inches* est observée au milieu des arêtes libres (point O). La figure 3.26 donne un schéma de la configuration ainsi que les données du problème avec un et quatre patchs.

La figure 3.27 montre que les résultats pour les deux modélisations sont similaires. Ce constat se retrouve également lorsque l'on compare la convergence des résultats en fonction des discrétisations à la figure 3.28.

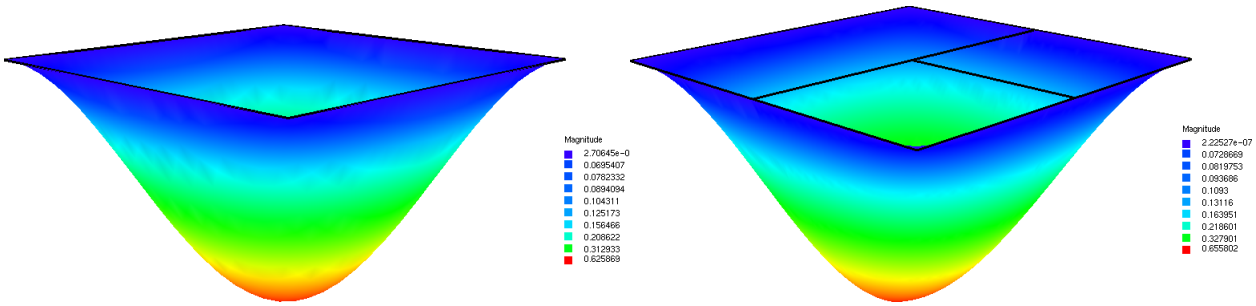
3.5.3 Conclusion sur les résultats des benchmarks

Les premiers cas test réalisés ont bien permis de retrouver les six modes de corps rigide. Ces résultats montrent que les jonctions entre coques sont bien prises en compte et que la structure étudiée se comporte comme un unique objet. De façon générale, les valeurs trouvées lors des simulations sont souvent inférieures aux valeurs théoriques. Toutefois, nous pouvons constater une certaine cohérence entre les résultats obtenus pour une coque modélisée avec un seul patch CAO et la même coque



(a) Configuration avec 1 patch (b) Configuration avec 3 patchs

FIGURE 3.20: Géométrie de la plaque avec un et trois patchs.



(a) Déplacement (×100) de la plaque avec un patch (b) Déplacement (×100) de la plaque avec trois patchs

FIGURE 3.21: Déplacements d'une plaque sous pression uniforme.

composée de plusieurs patchs. D'autre part, l'ensemble de ces benchmarks sont normalement réalisés sur une partie de la structure en prenant en compte des conditions de symétrie. Les résultats montrent que le code de calcul OPTSURF fournit une assez bonne approximation de la solution des problèmes étudiés.

3.6 Extension de la méthode à des configurations industrielles

L'analyse isogéométrique se base sur la géométrie et plus précisément sur les surfaces des modèles CAO. Afin de remplir sa fonction, l'analyse isogéométrique doit bénéficier de modèles CAO de qualité. Dans les faits, ce n'est pas toujours le cas. Dans l'industrie, les objets CAO le plus souvent utilisés ne sont pas les NURBS mais les patchs trimmés. Ce type de modélisation très présent dans les modeleurs comme CATIA©, consiste à définir les patchs par des surfaces parfois infinies qui sont ensuite découpées par des courbes appelées courbes de trim. Un exemple concret est celui du disque. À l'aide des modeleurs, une méthode simple pour construire un disque consiste à dessiner un cercle qui est ensuite « rempli » pour former une face. Le résultat obtenu est ainsi un disque (figure 3.29a). Toutefois, si l'on s'intéresse à la surface sous-jacente (figure 3.29b), cette dernière est très différente d'un disque. Lors de l'analyse isogéométrique, c'est cette surface sous-jacente qui est utilisée conduisant évidemment à des résultats erronés.

Il est cependant tout de même possible de représenter un disque avec des NURBS. Pour cela, il faut avoir recours à quatre surfaces NURBS. En théorie, de nombreuses géométries peuvent être dessinées à partir de NURBS. Elles nécessitent alors le redécoupage des patchs existants pour les transformer en NURBS. Cette option est malheureusement souvent insuffisante. Une approche en lien avec la méthode que nous proposons pourrait être de se baser sur la définition des patchs trimmés pour établir la fonction de forme.

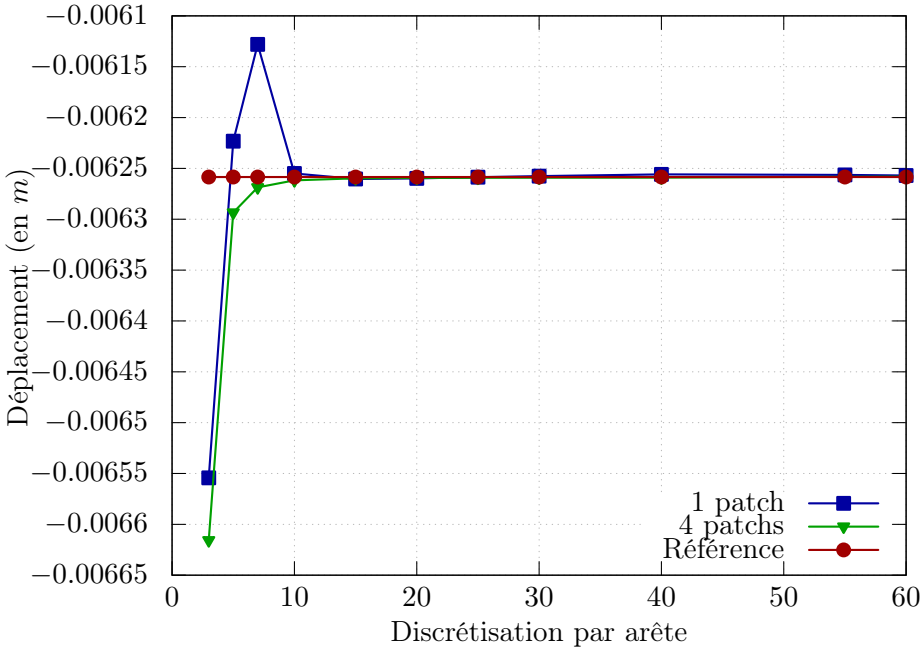


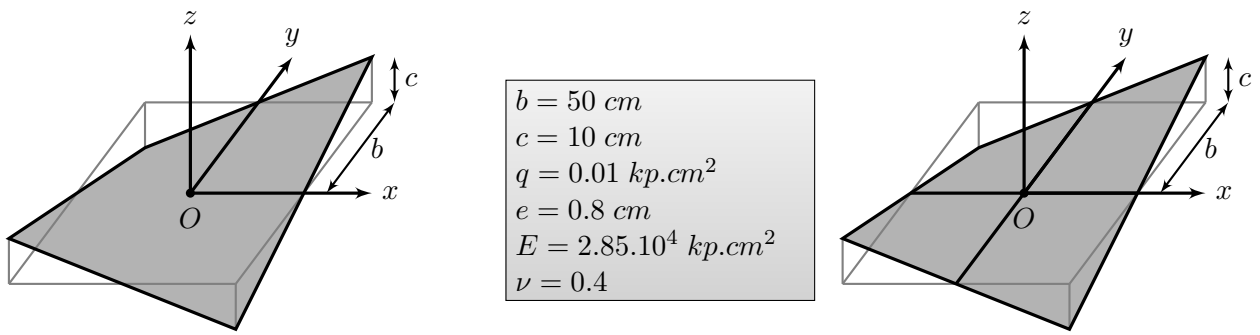
FIGURE 3.22: Déplacement du centre de la plaque sous pression en fonction de la discrétisation.

Discrétisation	Nombre d'éléments	Nombre de degrés de liberté
3	14	291
5	42	753
10	198	3069
15	460	6816
20	850	12291
25	135	19359
30	198	28047
40	355	49635
50	563	78135

TABLEAU 3.4: Nombre d'éléments et de degrés de liberté en fonction de la discrétisation par arête.

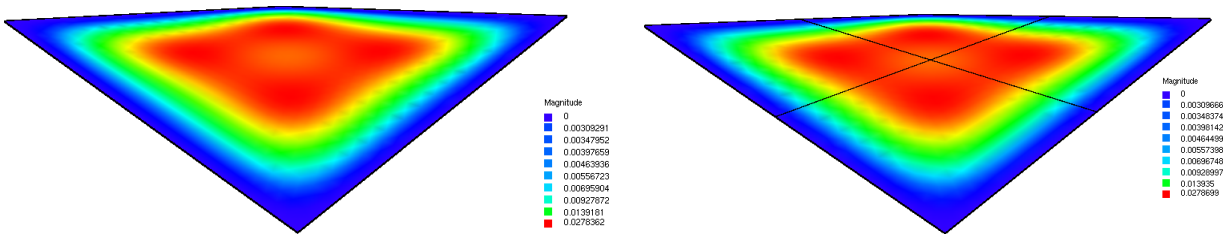
En plus des patchs trimmés, les CAO issues du monde industriel présentent également des problèmes au niveau des jonctions entre les patchs. Ces derniers ne sont pas toujours entièrement joints les uns aux autres (voir figure 3.30) en raison de la tolérance autorisée dans les modeleurs.

Les problèmes de connexions sont aussi présents lorsque les patchs sont des NURBS. Dans [Bazilevs *et al.*, 2010], un autre type de fonction de base est proposé : les T-splines. Les T-splines introduites dans [Sederberg *et al.*, 2003] se définissent de façon différente aux NURBS. Pour les NURBS, les vecteurs de nœuds sont communs à l'ensemble des fonctions de base B-splines. Les T-splines, en revanche, admettent un vecteur de nœuds propre à chaque fonction de base, ce qui permet une plus grande flexibilité. Des résultats montrent qu'en utilisant des T-splines le problème de patch non jointif n'existe plus car ils peuvent être modifiés localement pour assurer les liaisons. Les T-splines sont déjà implémentées au sein du logiciel RHINO© rendant leur utilisation de plus en plus répandue.



(a) Configuration avec 1 patch (b) Configuration avec 4 patchs

FIGURE 3.23: Géométrie du paraboloïde hyperbolique.



(a) Déplacement du paraboloïde avec un patch (b) Déplacement du paraboloïde avec quatre patchs

FIGURE 3.24: Déplacements du paraboloïde hyperbolique.

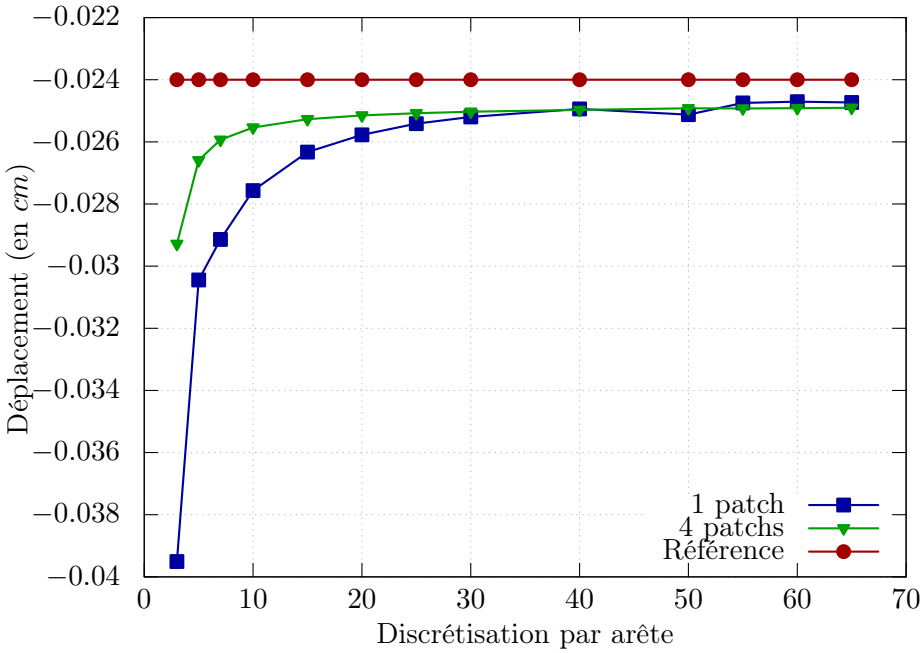
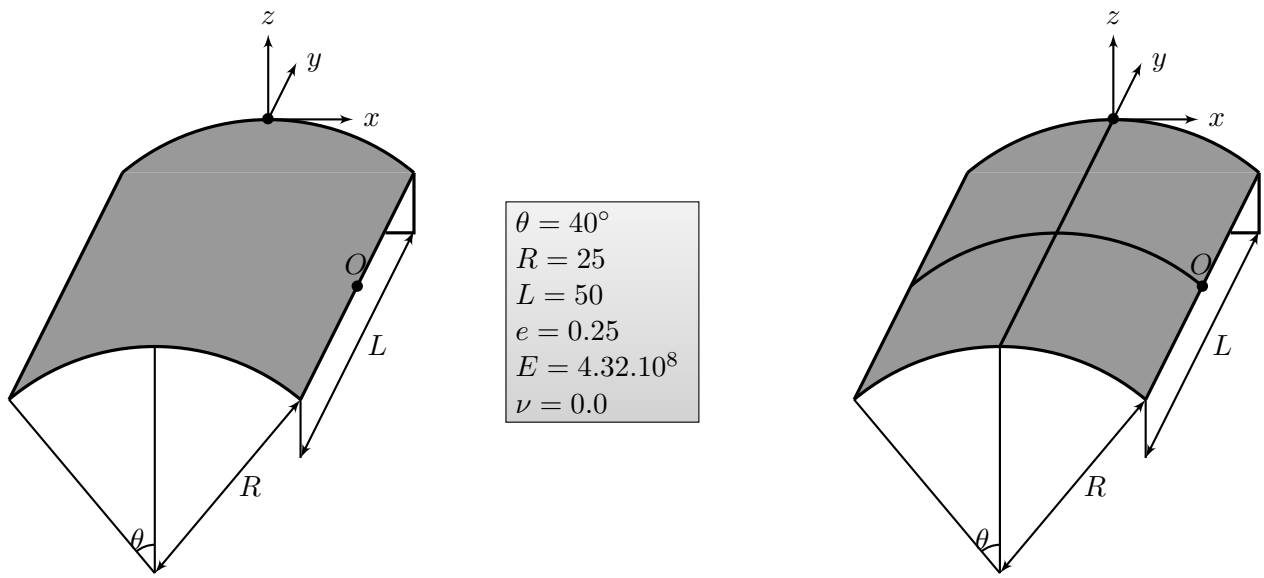


FIGURE 3.25: Déplacement du centre du paraboloïde hyperbolique en fonction de la discrétisation.

3.7 Synthèse du chapitre

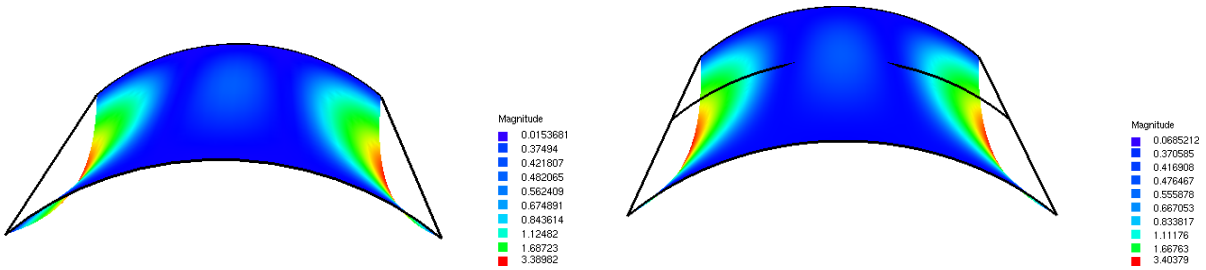
Dans ce chapitre, nous avons vu l'une des premières étapes vers l'industrialisation de la méthode, à savoir le traitement de surfaces multi-patchs. La question du raccordement entre patchs est essentielle car les modèles CAO issus de l'industrie comportent généralement de nombreux patchs pour satisfaire des géométries complexes. Ce raccordement est mis en œuvre ici en considérant chaque patch comme une coque et les raccords entre ces patchs comme des jonctions de coques. Le code de calculs OPT-



(a) Configuration avec 1 patch

(b) Configuration avec 4 patchs

FIGURE 3.26: Description du problème de la toiture de Scordelis Lo.



(a) Déplacement (×100) de la toiture avec un patch

(b) Déplacement (×100) de la toiture avec quatre patchs

FIGURE 3.27: Déplacements de la toiture de Scordelis Lo.

SURF développé fournit des résultats satisfaisants même s'il reste, avant tout, un démonstrateur du potentiel de la méthode proposée. Une industrialisation totale du procédé demandera encore plusieurs étapes pour lever les verrous restants, comme celui liée aux patchs trimmés. Il semble en effet difficile aujourd'hui de voir les industriels abandonner les modeleurs tels que CATIA© dans le but de rendre les géométries accessibles à l'analyse isogéométrique.

La suite du manuscrit aborde les questions d'optimisation de formes surfaciques dans le contexte de l'analyse isogéométrique. Le chapitre qui suit pose les bases de la stratégie d'optimisation pour un critère de compliance. Les critères liés à la vibro-acoustique seront étudiés dans le dernier chapitre.

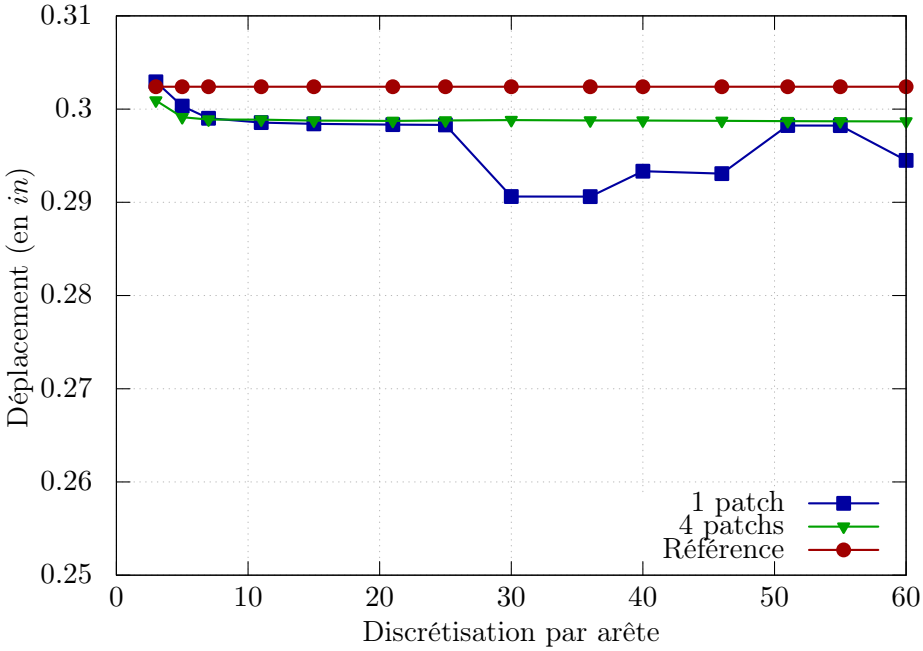
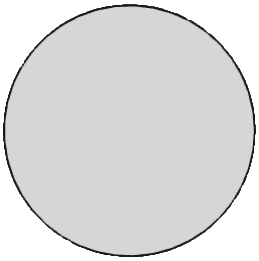
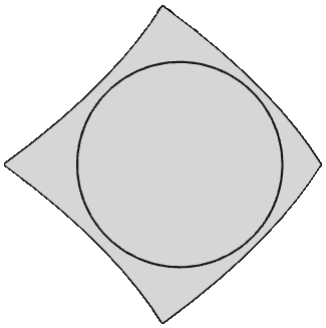


FIGURE 3.28: Déplacement du point O de la toiture de Scordelis Lo en fonction de la discrétisation.

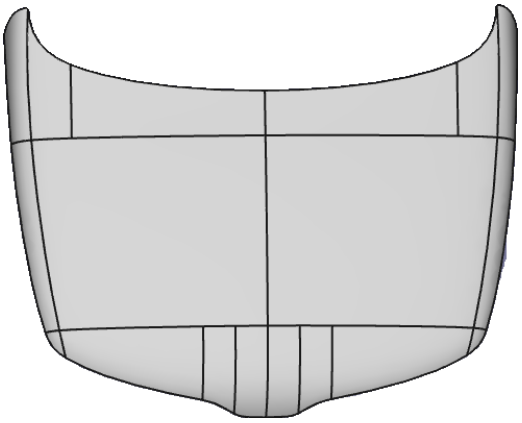


(a) CAO du disque

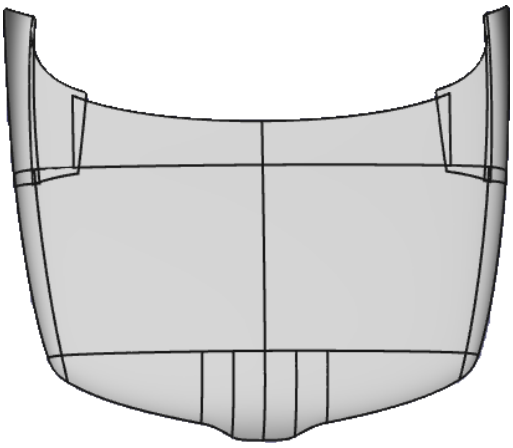


(b) Surface sous-jacente du disque et courbe de trim

FIGURE 3.29: Modèle CAO d'un disque et sa surface sous-jacente.



(a) CAO d'un capot



(b) Surfaces sous-jacentes

FIGURE 3.30: Modèle CAO d'un capot et ses surfaces sous-jacentes - Source : modèle CAO de Jean-François Remacle.

CHAPITRE

4 Optimisation de formes de coques pour un critère de compliance

Sommaire

4.1 Introduction **68**

4.2 Définition du problème d'optimisation **68**

4.2.1 Le critère à minimiser 68

4.2.2 Les variables d'optimisation 69

4.2.3 Les contraintes 69

4.2.3.1 Contraintes de bornes sur les points de contrôle 69

4.2.3.2 Contraintes sur l'aire 70

4.2.4 Formulation du problème d'optimisation 71

4.3 Calcul des gradients **71**

4.3.1 Différentielle de la fonction objectif et méthode de l'état adjoint 71

4.3.2 Calcul du gradient de la contrainte de l'aire 73

4.3.3 Implémentation des gradients 73

4.3.4 Comparaison des gradients avec des différences finies 74

4.3.4.1 Choix du pas pour les différences finies 74

4.3.4.2 Validation du gradient de la fonction objectif 76

4.3.4.3 Validation du gradient de la contrainte d'aire 76

4.4 L'algorithme d'optimisation **77**

4.4.1 Résolution du problème de barrière 77

4.4.2 Recherche linéaire 78

4.4.3 Les critères d'arrêt 79

4.5 Résultats d'optimisation **80**

4.5.1 Un plateau soumis à son propre poids 80

4.5.2 Un cube sous pression 82

4.5.3 Un bac de roue de secours 84

4.6 Synthèse du chapitre **86**

4.1 Introduction

La définition d'un problème d'optimisation dépend du choix du paramétrage de la forme. Nous avons choisi ici de définir les formes à l'aide de patchs CAO pour pouvoir utiliser leurs points de contrôle comme variables d'optimisation. Le chapitre précédent présentait la mise en place dans les simulations et en particulier la gestion de structures formées à partir de plusieurs patchs. Ce chapitre a pour objectif de mettre en œuvre la stratégie d'optimisation. La procédure choisie est présentée dans le cas d'un critère de compliance. Dans un premier temps, la définition précise du problème d'optimisation sera abordée. Les calculs de gradient et l'algorithme d'optimisation choisi seront ensuite présentés puis les résultats obtenus seront exposés.

4.2 Définition du problème d'optimisation

4.2.1 Le critère à minimiser

La compliance est un critère intervenant souvent en optimisation de formes. Elle correspond à l'énergie associée aux forces extérieures pour un problème en statique. L'objectif est de minimiser la compliance pour rendre la structure plus raide et donc limiter les déformations.

Considérons une structure $\mathcal{C}$ composée de $\mathcal{N}_p$ patchs CAO et de $\mathcal{N}_j$ jonctions. Soit $\Omega = \{\Omega^{(i)}\}_{i=\{1,\dots,\mathcal{N}_p\}}$ l'ensemble des domaines de référence associés à chaque patch. Soit $\gamma = \{\gamma^{(j)}\}_{j=\{1,\dots,\mathcal{N}_j\}}$ l'ensemble des frontières communes. La coque $\mathcal{C}$ est encastrée sur une partie ∂S_0 de surface moyenne image de $\partial\Omega_0$ par la fonction de forme. Elle est également chargée sur une partie ∂S_1, image de $\partial\Omega_1$. On notera $\vec{u}$ le déplacement de la coque entière tel que $\vec{u} = \{\vec{u}^{(i)}\}_{i=\{1,\dots,\mathcal{N}_p\}}$ où les $\vec{u}^{(i)}$ correspondent aux déplacements de chaque patch. La compliance j est donnée par la fonctionnelle suivante :

$$j(\varphi) \equiv J(\varphi, \vec{u}_\varphi) = \frac{1}{2} a(\vec{u}_\varphi, \vec{u}_\varphi) \tag{4.2.1}$$

où :

- φ est la forme qui appartient à l'ensemble des géométries admissibles $\mathcal{G}_{ad}$;
- $\vec{u}_\varphi \in \vec{\mathcal{V}}$ est la solution du problème de Koiter pour plusieurs patchs vérifiant :

$$\forall \vec{v} \in \vec{\mathcal{V}} \quad a(\vec{u}_\varphi, \vec{v}) = f(\vec{v}). \tag{4.2.2}$$

avec les définitions suivantes, introduites au chapitre 3 :

$$\begin{aligned} \vec{\mathcal{V}} = \Bigg\{ \vec{v} = \{\vec{v}^{(i)}\}_{i=\{1,\dots,\mathcal{N}_p\}} \;\Bigg|\; & \vec{v}^{(i)} = (v_\alpha^{(i)}, v_3^{(i)}) \in \left(H^1(\Omega^{(i)})\right)^2 \times H^2(\Omega^{(i)}); \\ & \vec{v}^{(i)}_{|\partial\Omega_0^{(i)}} = \vec{0}; \quad \left.\frac{\partial v_3^{(i)}}{\partial n}\right|_{\partial\Omega_0^{(i)}} = 0; \quad \vec{v}^{(i)}_{|\gamma^{(j)}} = \vec{v}^{(k)}_{|\gamma^{(j)}} \text{ avec } i \neq k \Bigg\} \end{aligned} \tag{4.2.3}$$

$$\begin{cases} a(\vec{u}, \vec{v}) = \displaystyle\sum_{i=1}^{\mathcal{N}_p} \int_{\Omega^{(i)}} eE^{\alpha\beta\lambda\mu} \left[\gamma_{\alpha\beta}(\vec{u}^{(i)}) \gamma_{\lambda\mu}(\vec{v}^{(i)}) + \frac{e^2}{12} \varrho_{\alpha\beta}(\vec{u}^{(i)}) \varrho_{\lambda\mu}(\vec{v}^{(i)}) \right] \sqrt{a}\, d\xi_1 d\xi_2 \\ \qquad + \displaystyle\sum_{j=1}^{\mathcal{N}_j} \int_{\gamma^{(j)}} k \left[(n^\beta (u_{3,\beta} + b_\beta^\alpha u_\alpha))^{(l)} - (\vec{t}^{(l)} \cdot \vec{t}^{(m)}) \cdot (n^\beta (u_{3,\beta} + b_\beta^\alpha u_\alpha))^{(m)} \right] \\ \qquad \left[(n^\beta (v_{3,\beta} + b_\beta^\alpha v_\alpha))^{(l)}) - (\vec{t}^{(l)} \cdot \vec{t}^{(m)}) \cdot (n^\beta (v_{3,\beta} + b_\beta^\alpha v_\alpha))^{(m)} \right] d\gamma \end{cases} \tag{4.2.4}$$

$$f(\vec{v}) = \sum_{i=1}^{\mathcal{N}_p} \int_{\Omega^{(i)}} \vec{p}^{(i)} \cdot \vec{v}^{(i)} \sqrt{a}\, d\xi_1 d\xi_2 + \int_{\partial\Omega_1^{(i)}} \vec{N}^{(i)} \cdot \vec{v}^{(i)} + \vec{M}^{(i)} \cdot \vec{\psi}(\vec{v}^{(i)})\, ds \tag{4.2.5}$$

L'équation (4.2.2) est en fait une généralisation pour plusieurs patchs de la formulation variationnelle du problème de jonction pour une charnière élastique introduite dans le chapitre 3. L'équation (4.2.3) regroupe les sommes des formes bilinéaires du modèle de Koiter pour chaque domaine de référence ainsi que les formes bilinéaires caractérisant les charnières. Rappelons que ces dernières font intervenir la définition de chacune des coques de part et d'autre de la jonction, ce qui explique la présence des exposants (l) et (m) dans l'équation (4.2.3).

Comme nous l'avons vu dans le chapitre 1, un problème d'optimisation de formes nécessite trois éléments : un critère, une équation d'état et un ensemble admissible de variables. Dans ce chapitre, la fonction objectif sera la compliance dont la fonctionnelle est donnée par l'équation (4.2.1) et l'équation d'état est donnée par l'équation (4.2.2). Le dernier élément, l'ensemble admissibles de variables, est l'objet des paragraphes suivants.

4.2.2 Les variables d'optimisation

En optimisation de formes, les variables d'optimisation décrivent la forme de la structure étudiée. Le chapitre 1 présentait quelques choix de paramétrage possible. Dans le cadre de cette étude, les formes sont définies à l'aide de surfaces NURBS. L'intérêt d'avoir recours à des NURBS est que ces surfaces peuvent être pilotées par un petit nombre de paramètres, tout en offrant d'importantes variations de la forme. Les variables d'optimisation sont donc les coordonnées ainsi que les poids des points de contrôles. Soit $\mathbb{P} = \{P_1, ..., P_n\}$ l'ensemble des n points de contrôles. La forme φ peut ainsi être définie de la façon suivante :

$$\varphi = \Big(x(P_1), y(P_1), z(P_1), w(P_1),\ \dots\ , x(P_n), y(P_n), z(P_n), w(P_n) \Big). \tag{4.2.6}$$

où $x(P_i), y(P_i), z(P_i), w(P_i)$ correspondent respectivement aux coordonnées suivant x, y, z et au poids du point de contrôle P_i. La forme φ est ainsi représentée comme un vecteur de $\mathbb{R}^{4n}$. En pratique, tous les points et toutes les coordonnées ne seront pas pris en compte. Par exemple, afin de respecter le cahier des charges, la position de certains points de contrôles doit rester inchangée. D'autre part, pour des raisons numériques, certaines contraintes sur les points de contrôle doivent être imposées. L'espace admissible des variables devra intégrer toutes les contraintes que nous allons à présent détailler.

4.2.3 Les contraintes

Lorsque que l'on cherche à optimiser une pièce pour un véhicule, certaines règles sont fournies par le cahier des charges. En effet, à la fin de l'optimisation, la pièce doit toujours pouvoir être assemblée au reste de la voiture. En plus des contraintes, on peut par exemple devoir prendre en compte les procédés de fabrication ou d'emboutissage. Le bon fonctionnement de l'optimisation repose aussi sur les contraintes à imposer aux variables. Les variables d'optimisation étant les coordonnées des points de contrôle, les contraintes doivent être exprimées en fonction de ces dernières. Dans les cas d'optimisation que nous traiterons, nous considérerons deux types de contraintes : les bornes à imposer aux points de contrôle et les variations d'aire par rapport à l'aire originale.

4.2.3.1 Contraintes de bornes sur les points de contrôle

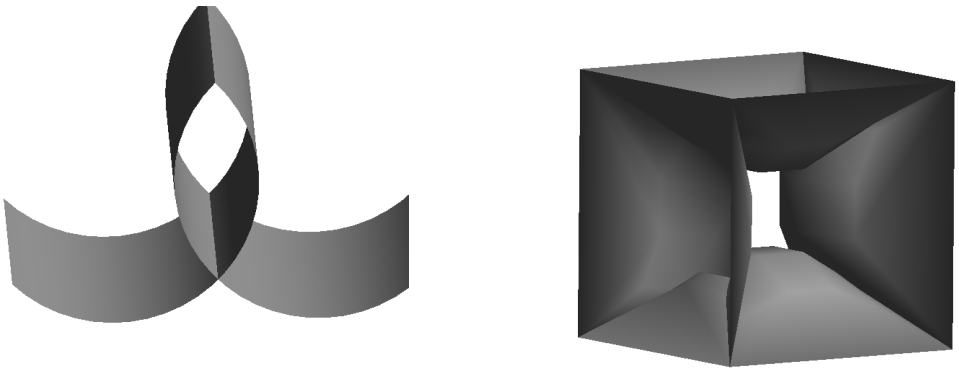
Parmi les contraintes de conception les plus fréquentes dans l'industrie, il est souvent demandé que les pièces n'excèdent pas un certain volume. Si l'on prend le cas d'une doublure de capot, cette dernière doit être conçue de sorte qu'elle puisse être placée entre le capot et le moteur. Pour satisfaire ce type

de contraintes, des bornes de variation peuvent être imposées aux paramètres. Ces contraintes sont de la forme :

$$\begin{cases} x_{min} \leq x(P_i) \leq x_{max} \\ y_{min} \leq y(P_i) \leq y_{max} \\ z_{min} \leq z(P_i) \leq z_{max} \end{cases} \tag{4.2.7}$$

Remarque 4.2.1. Dans l'équation (4.2.7), les contraintes sont uniquement appliquées sur les coordonnées des points de contrôle. Nous n'imposerons pas d'autres contraintes sur les poids associés que d'appartenir à $[0, 1]$.

En dehors des besoins industriels, plusieurs librairies proposant des algorithmes d'optimisation, comme NLopt ou Ipopt, requièrent des plages de variation pour les variables d'optimisation. De plus, le fait de borner les coordonnées des points de contrôle permet de limiter les risques d'intersection des surfaces. En effet, lors de modifications des positions des pôles, des formes comme celles de la figure 4.1, où les patchs s'intersectent, peuvent apparaître. Dans ce cas de figure, l'optimisation doit impérativement être arrêtée car les simulations ne pourront pas fournir un résultat fiable, le modèle n'ayant pas été défini pour ce type de formes.



(a) (b)

Figure 4.1: Exemple de figures où les patchs s'intersectent.

Pour éviter cette situation, en plus de borner les coordonnées des points de contrôle, nous nous limiterons parfois à ne prendre qu'une ou deux coordonnées des points de contrôle comme variables d'optimisation.

4.2.3.2 Contraintes sur l'aire

Lorsque la position des points des contrôle est modifiée au cours de l'optimisation, la forme peut être étirée et, par conséquent, l'aire de celle-ci augmenter. Ce phénomène implique donc que la nouvelle forme nécessitera davantage de matériau et si l'épaisseur reste constante, la masse sera également augmentée. Pour des raisons de coût et de performance des produits, il est préférable de limiter les variations d'aire. Dans cette optique, des contraintes de variations d'aire sont imposées. Il peut s'agir par exemple d'autoriser une variation d'un petit pourcentage de l'aire d'origine ou bien d'exiger de conserver l'aire initiale. En notant $\mathcal{A}$ l'aire de la structure totale, avec la modélisation utilisée, on a :

$$\mathcal{A} = \sum_{i=1}^{\mathcal{N}_p} \int_{\Omega^{(i)}} \sqrt{a}\, d\xi_1 d\xi_2. \tag{4.2.8}$$

Contrairement aux bornes sur les points de contrôle, la contrainte sur l'aire a une formulation moins explicite par rapport aux points de contrôle. Rappelons que l'élément d'aire $\sqrt{a}$ est déterminé à partir de la base covariante qui dépend de la forme et donc des points de contrôle. La mise en place de telles conditions nécessite un travail profond sur les questions de géométrie afin d'exprimer ces contraintes à partir des points de contrôle. Il faudra, par ailleurs, s'assurer d'utiliser un algorithme d'optimisation ne violant jamais les contraintes fortes du problème.

4.2.4 Formulation du problème d'optimisation

La section précédente a permis d'achever la définition de l'ensemble admissible des variables $\mathcal{G}_{ad}$. Le problème d'optimisation à traiter est donc de la forme suivante : on cherche

$$\begin{aligned} &\varphi \in \mathcal{G}_{ad} = \left\{ (x(P_i), y(P_i), z(P_i), w(P_i)) \in \mathbb{R}^{4n}, \text{tel que : } \begin{array}{c} x_{min} \leq x(P_i) \leq x_{max} \ , \\ y_{min} \leq y(P_i) \leq y_{max} \ , \\ z_{min} \leq z(P_i) \leq z_{max} \ , \\ \text{et } a_{min} \leq \mathcal{A} \leq a_{max} \ . \end{array} \right\} \\ &\text{qui minimise} \quad j(\varphi) \equiv J(\varphi, \vec{u}_\varphi) = \frac{1}{2} a(\vec{u}_\varphi, \vec{u}_\varphi) \\ &\text{où } u_\varphi \text{ est tel que} \quad \forall \vec{v} \in \vec{\mathcal{V}} \quad a(\vec{u}_\varphi, \vec{v}) = f(\vec{v}). \end{aligned} \tag{4.2.9}$$

L'ensemble $\mathcal{G}_{ad}$ est un sous espace compact de $\mathbb{R}^{4n}$. La fonction j est continue sur $\mathcal{G}_{ad}$, le problème d'optimisation donné par l'équation (4.2.9) admet donc au moins un minimum global.

Le problème d'optimisation introduit ci-dessus est résolu à l'aide d'un algorithme de descente de type gradient. Dans un premier temps, nous exposerons comment déterminer les gradients de la fonction objectif et des contraintes avant de présenter l'algorithme utilisé pour l'optimisation.

4.3 Calcul des gradients

Les algorithmes de gradient nécessitent le calcul des dérivées à la fois, de la fonction objectif et des contraintes. Le gradient de la fonction objectif sera calculé à l'aide de la méthode de l'état adjoint que nous présenterons dans ce qui suit. Concernant les contraintes, seul le calcul du gradient de l'aire sera détaillé puisque les bornes de points de contrôle seront directement incluses dans la routine de l'algorithme d'optimisation.

4.3.1 Différentielle de la fonction objectif et méthode de l'état adjoint

La compliance, donnée par l'équation (4.2.1), est une fonctionnelle qui dépend de la solution de l'équation d'état en plus des variables de forme φ. Soit u l'application qui à φ associe $u(\varphi) = \vec{u}_\varphi$ la solution de l'équation d'état supposée différentiable [Bernadou *et al.*, 1991] . La différentielle de j par rapport à la forme φ est donnée par :

$$Dj(\varphi) \cdot \psi = \partial_\varphi J(\varphi, \vec{u}_\varphi) \cdot \psi + \partial_u J(\varphi, \vec{u}_\varphi)[Du(\varphi) \cdot \psi]. \tag{4.3.1}$$

L'équation (4.3.1) fait apparaître la différentielle par rapport à la forme de la solution de l'équation d'état $Du(\varphi)$. Contrairement aux entités constituant la forme a, nous ne disposons pas d'une expression explicite de cette différentielle. Deux approches sont alors possibles : la méthode directe et la méthode de l'état adjoint [Rousselet, 1986].

La méthode directe consiste à déterminer $Du(\varphi)$ à partir de l'équation d'état. En dérivant par rapport à φ l'équation (4.2.2), on obtient :

$$\forall \vec{v} \in \mathcal{V} \quad \partial_\varphi a(\vec{u}_\varphi, \vec{v}) \cdot \psi + \partial_u a(\vec{u}_\varphi, \vec{v}) Du(\varphi) \cdot \psi = \partial_\varphi f(\vec{v}) \cdot \psi. \tag{4.3.2}$$

La résolution de l'équation (4.3.2) permet de calculer la différentielle $Du(\varphi)$. L'inconvénient de cette méthode est que cette équation devra être résolue pour chaque variable d'optimisation. Ainsi, pour une structure complexe comportant un nombre élevé de points de contrôle, la quantité de calcul sera importante avant de pouvoir obtenir un gradient. D'autre part, pour certains problèmes d'optimisation de forme, la méthode directe peut être difficile à mettre en place.

La technique de l'état adjoint repose sur l'idée de considérer le problème suivant : on cherche à minimiser j sous la contrainte que l'équation d'état soit vérifiée. Comme pour un problème d'optimisation sous contraintes, on introduit le Langragien $\mathcal{L}$ défini par l'équation suivante : [Céa, 1986].

$$\mathcal{L}(\vec{u}, \vec{v}, \varphi) = J(\varphi, \vec{u}) - \{a(\vec{u}, \vec{v}) - f(\vec{v})\}. \tag{4.3.3}$$

Remarquons que si $\vec{u} = \vec{u}_\varphi$ est solution de l'équation d'état, alors

$$\mathcal{L}(\vec{u}_\varphi, \vec{v}, \varphi) = J(\varphi, \vec{u}_\varphi) = j(\varphi), \tag{4.3.4}$$

et, par suite,

$$D_\varphi \mathcal{L}(\vec{u}_\varphi, \vec{v}, \varphi) \cdot \psi = Dj(\varphi) \cdot \psi. \tag{4.3.5}$$

Par définition, nous avons :

$$D_\varphi \mathcal{L}(\vec{u}, \vec{v}, \varphi) \cdot \psi = \partial_\varphi \mathcal{L}(\vec{u}, \vec{v}, \varphi) \cdot \psi + \partial_u \mathcal{L}(\vec{u}, \vec{v}, \varphi)[Du(\varphi) \cdot \psi] + \partial_v \mathcal{L}(\vec{u}, \vec{v}, \varphi)[Dv(\varphi) \cdot \psi] \tag{4.3.6}$$

Pour obtenir la dérivée de la fonction objectif, il suffit de résoudre les deux problèmes suivants :

$$\text{trouver } \vec{u} \text{ tel que} \quad \forall \vec{w} \in \vec{\mathcal{V}} \quad \partial_v \mathcal{L}(\vec{u}, \vec{v}, \varphi) \cdot \vec{w} = 0, \tag{4.3.7a}$$

$$\text{trouver } \vec{v} \text{ tel que} \quad \forall \vec{w} \in \vec{\mathcal{V}} \quad \partial_u \mathcal{L}(\vec{u}_\varphi, \vec{v}, \varphi) \cdot \vec{w} = 0. \tag{4.3.7b}$$

L'équation (4.3.7a), qui revient à écrire l'équation d'état, a donc pour solution $\vec{u}_\varphi$. L'équation (4.3.7b) est ce que l'on appelle le problème adjoint et a pour solution la variable adjointe $\vec{v}_\varphi$. En injectant ces deux solutions dans l'équation (4.3.6), on trouve finalement :

$$Dj(\varphi) \cdot \psi = \partial_\varphi \mathcal{L}(\vec{u}_\varphi, \vec{v}_\varphi, \varphi) \cdot \psi. \tag{4.3.8}$$

Proposition 4.3.1. La différentielle de la compliance (équation (4.2.1)) s'écrit

$$Dj(\varphi) \cdot \psi = -\frac{1}{2} \partial_\varphi a(\vec{u}_\varphi, \vec{u}_\varphi) \cdot \psi + \partial_\varphi f(\vec{u}_\varphi) \cdot \psi \tag{4.3.9}$$

Démonstration. On vérifie tout d'abord facilement que l'équation (4.3.7a) a pour solution $\vec{u}_\varphi$.

Il faut maintenant résoudre l'équation (4.3.7b). En utilisant le résultat trouvé précédemment, il vient :

$$\begin{aligned} \forall \vec{w} \in \vec{\mathcal{V}}, \qquad \partial_u \mathcal{L}(\vec{u}_\varphi, \vec{v}, \varphi) \cdot \vec{w} &= \frac{1}{2} \times 2 \times a(\vec{u}_\varphi, \vec{w}) - a(\vec{w}, \vec{v}) = 0 \\ &\iff a(\vec{u}_\varphi, \vec{w}) - a(\vec{w}, \vec{v}) = 0 \\ &\iff f(\vec{w}) - a(\vec{w}, \vec{v}_\varphi) = 0 \end{aligned}$$

La solution du problème adjoint est donc la même que celle de l'équation d'état : $\vec{v}_\varphi = \vec{u}_\varphi$ (problème d'optimisation auto-adjoint). Finalement,

$$\begin{aligned} Dj(\varphi) \cdot \psi &= \partial_\varphi \mathcal{L}(\vec{u}_\varphi, \vec{v}_\varphi, \varphi) \cdot \psi. \\ &= \frac{1}{2} \partial_\varphi a(\vec{u}_\varphi, \vec{u}_\varphi) \cdot \psi - \partial_\varphi a(\vec{u}_\varphi, \vec{u}_\varphi) \cdot \psi + \partial_\varphi f(\vec{u}_\varphi) \cdot \psi \\ &= -\frac{1}{2} \partial_\varphi a(\vec{u}_\varphi, \vec{u}_\varphi) \cdot \psi + \partial_\varphi f(\vec{u}_\varphi) \cdot \psi \end{aligned}$$

□

Les termes $\partial_\varphi a(\vec{u}_\varphi, \vec{u}_\varphi)\cdot\psi$ et $\partial_\varphi f(\vec{u}_\varphi)\cdot\psi$ sont déterminés à l'aide des dérivées des entités géométriques introduites au chapitre 2 dont les formulations complètes sont données dans [Bernadou *et al.*, 1991]. Ces expressions doivent être ensuite adaptées à la fonction de forme choisie et aux variables d'optimisation. Il faut donc utiliser la définition des NURBS et dériver par rapport aux coordonnées des points de contrôle. Par exemple, si l'on cherche à déterminer la dérivée du premier vecteur de la base covariante par rapport à la coordonnée x du point de contrôle $P_{k,l}$, on obtient :

$$\partial_{x(P_{k,l})}\vec{a}_1 = \begin{bmatrix} \left\{ \begin{array}{c} \dfrac{N'_{k,p}(\xi_1)\cdot N_{l,q}(\xi_2)w_{k,l} \times \displaystyle\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(\xi_1)\cdot N_{j,q}(\xi_2)w_{i,j}}{\left(\displaystyle\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(\xi_1)\cdot N_{j,q}(\xi_2)w_{i,j}\right)^2} \\ -\dfrac{N_{k,p}(\xi_1)\cdot N_{l,q}(\xi_2)w_{k,l} \times \displaystyle\sum_{i=0}^{n}\sum_{j=0}^{m} N'_{i,p}(\xi_1)\cdot N_{j,q}(\xi_2)w_{i,j}}{\left(\displaystyle\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(\xi_1)\cdot N_{j,q}(\xi_2)w_{i,j}\right)^2} \end{array} \right. \\ 0 \\ 0 \end{bmatrix} \tag{4.3.10}$$

avec $N_{i,p}$ les fonctions de bases B-splines et $N'_{i,p}$ leurs dérivées par rapport à ξ. Tous les éléments pour le calcul du gradient de la fonction objectif $\nabla_\varphi j$ sont ainsi connus. Nous l'écrirons sous la forme :

$$\nabla_\varphi j = \begin{bmatrix} \partial_{x(P_1)}j \\ \partial_{y(P_1)}j \\ \partial_{z(P_1)}j \\ \partial_{w(P_1)}j \\ \vdots \\ \partial_{x(P_n)}j \\ \partial_{y(P_n)}j \\ \partial_{z(P_n)}j \\ \partial_{w(P_n)}j \end{bmatrix} \tag{4.3.11}$$

4.3.2 Calcul du gradient de la contrainte de l'aire

Lors d'une optimisation sous contraintes, avec un algorithme de gradient, il faut également pouvoir calculer les gradients des contraintes.

Le calcul de l'aire de l'équation (4.2.8) ne fait intervenir que l'élément d'aire élémentaire. Ainsi :

$$(D_\varphi \mathcal{A})\cdot\psi = \int_\Omega \partial_\varphi(\sqrt{a})\cdot\psi \; d\xi_1 d\xi_2. \tag{4.3.12}$$

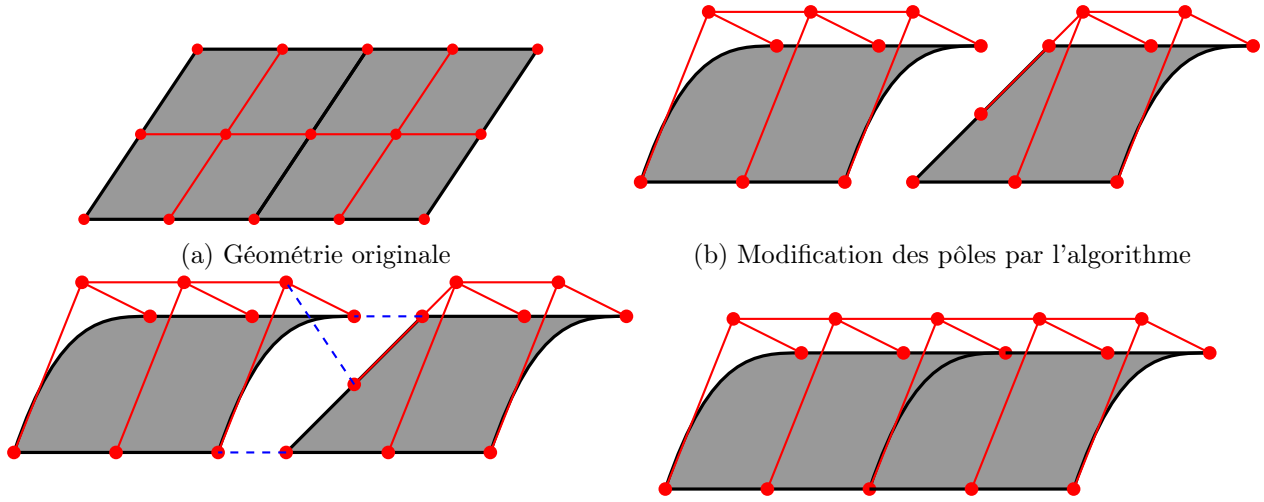
avec $\partial_\varphi(\sqrt{a})\cdot\psi = \sqrt{a}\times[a^\mu\cdot\partial_{\xi_\mu}\cdot\psi]$. De même que pour la fonction objectif, tous les éléments peuvent être calculés explicitement en fonction des points des contrôle.

4.3.3 Implémentation des gradients

Le gradient de la compliance est implémenté dans le code OptSurf en se basant sur les dérivées des entités composant la formulation variationnelle de l'équation (4.2.2). Les dérivées par rapport aux directions de tous les points de contrôles et poids sont codées en suivant le modèle de l'équation (4.3.10)

et les informations données par OCCT.

À l'heure actuelle, OptSurf permet d'optimiser les géométries modélisées exclusivement par des patchs conformes et de même degré. Lors de la lecture de la CAO, les points de contrôle de chaque patch sont détectés. Les points de contrôles se trouvant sur une jonction apparaissent donc deux fois dans la liste des points de contrôles. Dans le cas de patchs conformes, il suffit de considérer ceux-ci une seule fois. Après l'itération d'optimisation, il suffit ensuite d'imposer l'égalité des points de contrôle sur les deux patchs (voir figure 4.2).



(a) Géométrie originale

(b) Modification des pôles par l'algorithme

(c) Égalisation des points de contrôle

(d) Nouvelle géométrie

Figure 4.2: Gestion des points de contrôle des jonctions entre patchs.

Dans le cas de patchs non conformes ou de degrés différents, le nombre de points de contrôle de part et d'autre de la jonction diffère et la procédure devra être adaptée.

4.3.4 Comparaison des gradients avec des différences finies

Avant de pouvoir conduire une optimisation, une étape essentielle est la validation des calculs de gradients. À partir des expressions des dérivées des différentes entités et avec l'aide de la libraire OCCT, les dérivées données par les équations (4.3.9) et (4.3.12) ont été implémentées dans le code OptSurf. Afin de s'assurer des résultats obtenus, des comparaisons avec des calculs de gradients par différences finies ont été réalisées. Pour illustrer les performances du gradient, les résultats du gradient de la compliance dans le cas du problème du paraboloïde hyperbolique présenté au chapitre 3 sont exposés. La géométrie du paraboloïde est constituée de 4 patchs NURBS de degré 2. Elle compte 36 points de contrôle au total. Pour le calcul du gradient, les pôles répétés ne sont pas pris en compte ce qui ramène donc à 25 points de contrôle. La figure 4.3 donne la configuration des points de contrôle. Les variables d'optimisation considérées ici, sont les 25 coordonnées suivant la direction z des points de contrôle.

Dans un premier temps, nous justifierons le choix du pas des différences finies avant de discuter des résultats de comparaisons obtenus.

4.3.4.1 Choix du pas pour les différences finies

La validation du gradient est faite en comparant les résultats avec des différences finies d'ordre 1 de la forme suivante :

$$j'(\varphi) \simeq \frac{j(\varphi + h) - j(\varphi)}{h}. \tag{4.3.13}$$

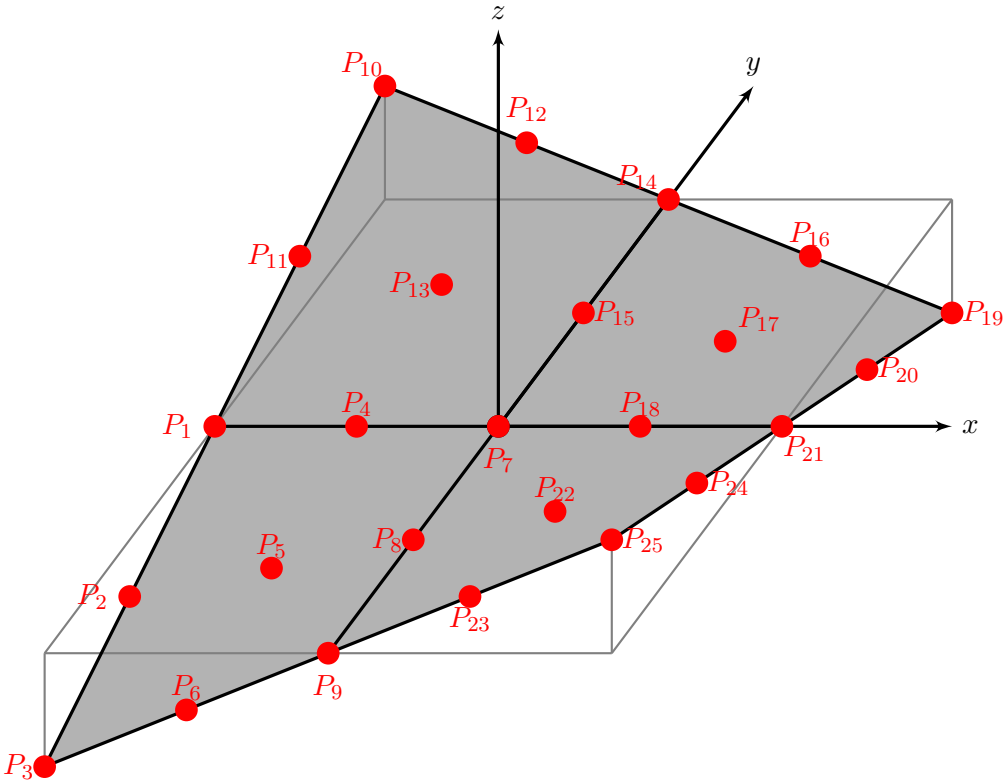


FIGURE 4.3: Points de contrôle de la géométrie du paraboloïde hyperbolique.

Lors d'un calcul de dérivée par différences finies, il faut veiller à choisir un paramètre h assez petit pour avoir une bonne approximation. Toutefois, il faut également s'assurer que le paramètre h ne soit pas trop petit à cause du risque d'instabilités numériques. La figure 4.4 montre un calcul par différences finies de la dérivée partielle $\dfrac{\partial j}{\partial z(P_8)}$ pour différents paramètres h.

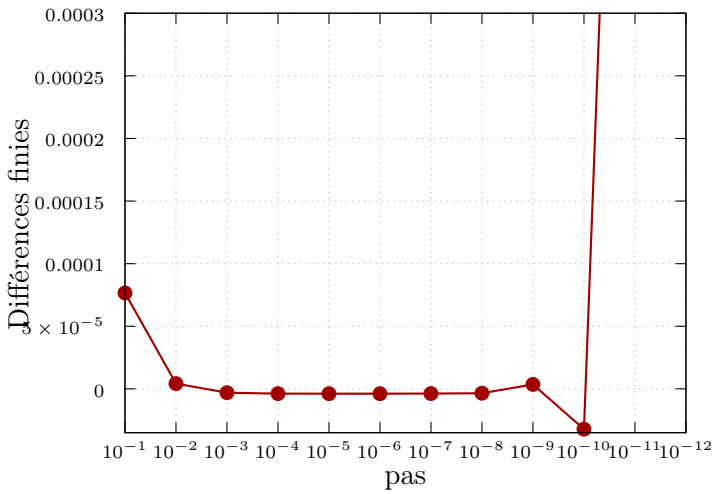
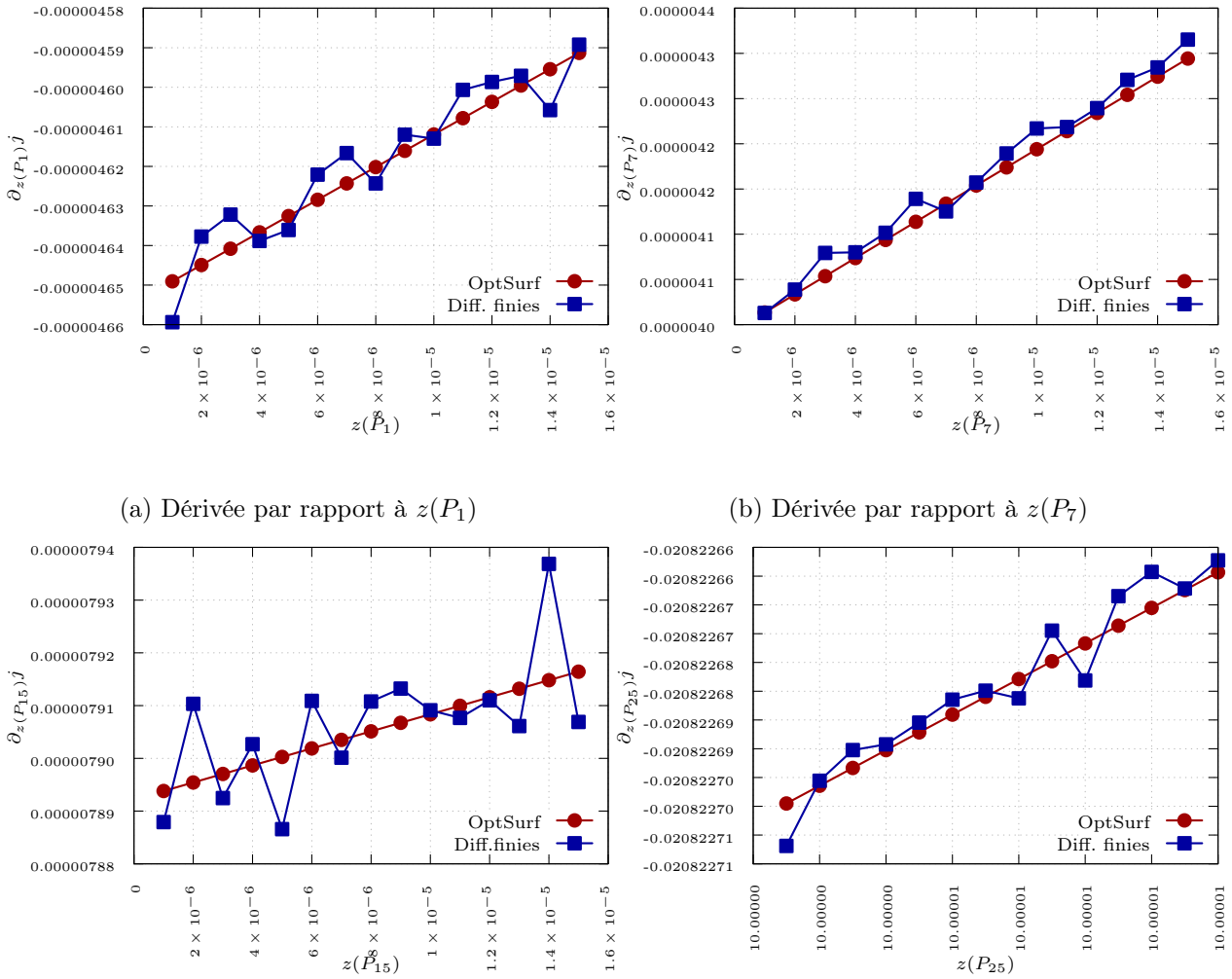


FIGURE 4.4: Évolution de la différence finie en fonction du paramètre h.

Comme nous pouvons le constater, pour $h \in [10^{-4}, 10^{-8}]$, la différence finie est stable alors que ce n'est pas le cas pour les valeurs en dehors de ce domaine. Pour les comparaisons avec les gradients, le paramètre $h = 10^{-6}$ a été retenu.

4.3.4.2 Validation du gradient de la fonction objectif

Les dérivées par rapport à tous les points de contrôle ont été comparées aux différences finies. La figure 4.5 regroupe quelques uns des résultats obtenus. Les dérivées ont été comparées pour plusieurs valeur de $z(P_i)$. Les calculs ont été effectués avec 360 éléments de discrétisation.



(a) Dérivée par rapport à $z(P_1)$

(b) Dérivée par rapport à $z(P_7)$

(c) Dérivée par rapport à $z(P_{15})$

(d) Dérivée par rapport à $z(P_{25})$

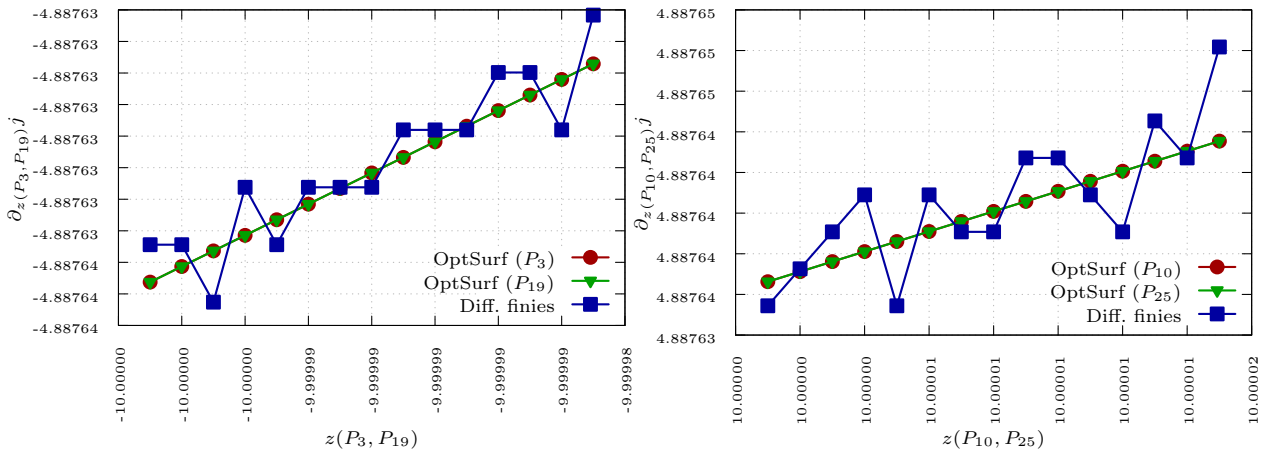
FIGURE 4.5: Comparaison entre les dérivées de la compliance calculées avec OPTSURF et leurs approximations pas différences finies.

Les résultats sont très satisfaisants. Les plus gros écarts entre les différences finies et les calcul sont de l'ordre de 0.004% en erreur relative (figure 4.5c). Le gradient peut être aussi bien calculé pour des points sur une jonction (figure 4.5c) que pour des crosspoints (figure 4.5b).

4.3.4.3 Validation du gradient de la contrainte d'aire

Étant donné que l'objectif est de réaliser une optimisation sous contrainte d'aire, le même exercice de validation doit être fait pour le gradient de celle-ci.

Les résultats sont similaires aux précédents, à savoir que l'écart entre les différences finies et le gradient calculé est faible. Nous remarquons également que ce dernier conserve la symétrie. En effet, les courbes des gradients des figures 4.6a et 4.6b sont confondues pour des points de contrôle symétriques. Ces



(a) Dérivée par rapport à $z(P_3)$ et$z(P_{19})$ (b) Dérivée par rapport à $z(P_{10})$ et$z(P_{25})$

FIGURE 4.6: Comparaison entre les dérivées de l'aire calculées avec OPTSURF et leurs approximations par différences finies.

résultats montrent que nous disposons d'un gradient valable pour le calcul d'une direction de descente. Le but de la section suivante est de présenter l'algorithme d'optimisation choisi. Nous verrons en particulier comment est utilisé le gradient pour déterminer une direction de descente.

4.4 L'algorithme d'optimisation

L'algorithme d'optimisation que nous utiliserons est celui de la libraire IPOPT (*Interior Point OPTimizer*). Il s'agit d'un algorithme primal-dual avec une méthode de recherche linéaire basée sur les techniques de filtre. La description complète de l'algorithme est disponible dans [Wächter et Biegler, 2006]. L'algorithme de IPOPT, en tant qu'algorithme de point intérieur, appartient aux algorithmes d'optimisation par pénalisation de la contrainte.

Considérons le problème d'optimisation suivant :

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} f(x) \\ \text{tel que} \quad c(x) = 0 \text{ et } x \geq 0. \end{cases} \tag{4.4.1}$$

La méthode de pénalisation consiste à résoudre une suite de problèmes de barrière de la forme :

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} g_\mu(x) = f(x) - \mu \sum\limits_{i=1}^{n} \ln(x_i) \\ \text{tel que } c(x) = 0 \end{cases} \tag{4.4.2}$$

pour une suite de paramètres de barrière (μ_j) décroissante et convergeant vers 0. À chaque itération, les deux principales étapes de l'algorithme sont la résolution du problème de barrière et le calcul du pas par une recherche linéaire avec une méthode de filtre.

4.4.1 Résolution du problème de barrière

Dans un premier temps, le problème de barrière est résolu dans le but de dégager une direction de descente. Soit $\mathcal{L}$ le Lagragien associé au problème (4.4.1) tel que :

$$\mathcal{L}(x, \lambda, z) = f(x) + c(x)^T \cdot \lambda - z \tag{4.4.3}$$

où $\lambda \in \mathbb{R}^m$ et $z \in \mathbb{R}^n$ sont les multiplicateurs de Lagrange respectivement associés aux conditions $c(x) = 0$ et $x \geq 0$. Le problème (4.4.2) peut se ramener à la résolution d'un problème à la résolution du système d'équations suivant :

$$\nabla f(x) + \nabla c(x) \cdot \lambda - z = 0 \; ; \tag{4.4.4}$$

$$c(x) = 0 \; ; \tag{4.4.5}$$

$$XZe - \mu e = 0 \; ; \tag{4.4.6}$$

avec

$$X = \begin{bmatrix} x_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & x_n \end{bmatrix}, \quad Z = \begin{bmatrix} z_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & z_n \end{bmatrix}, \quad e = \begin{bmatrix} 1 \\ \vdots \\ \vdots \\ 1 \end{bmatrix}$$

La méthode alors utilisée pour résoudre ce problème de barrière pour un paramètre μ_j donné, consiste à appliquer un algorithme de Newton aux équations primales-duales. À chaque itéré (x^k, λ^k, z^k) avec $x^k, z^k > 0$, les directions de descente $(d_x^k, d_\lambda^k, d_z^k)$ sont obtenues en résolvant :

$$\begin{bmatrix} W^k + \Sigma^k & A^k \\ (A^k)^T & 0 \end{bmatrix} \begin{bmatrix} d_x^k \\ d_\lambda^k \end{bmatrix} = - \begin{bmatrix} \nabla g_{\mu_j}(x^k) + A^k \lambda^k \\ c(x^k) \end{bmatrix} \tag{4.4.7}$$

avec :

- $A^k = \nabla c(x^k)$;
- $W^k = \nabla_{xx} \mathcal{L}(x_k, \lambda_k, z_k)$ la matrice Hessienne du Lagragien ;
- $\Sigma^k = X^{k^{-1}} Z^k$;

et $d_z^k = \mu_j X^{k^{-1}} e - z^k - \Sigma^k d_x^k$.

Remarque 4.4.1. Le système (4.4.7) nécessite les Hessiennes de la fonction objectif et des contraintes via W^k. Si ces dernières ne sont pas fournies, comme dans notre cas, la librairie IPOPT fait alors appel à une méthode de quasi-Newton pour approcher celles-ci.

Une fois la direction de descente déterminée, il reste à estimer le pas afin de pouvoir calculer le nouvel itéré $(x^{k+1}, \lambda^{k+1}, z^{k+1})$.

4.4.2 Recherche linéaire

La recherche linéaire, dans IPOPT, est réalisée à l'aide d'une méthode de filtre. L'idée est de voir le problème de barrière (4.4.2) pour μ_j comme un problème d'optimisation bi-objectif. Le but est de minimiser à la fois la fonction objectif g_{μ_j} et la violation de la contrainte $\theta(x) = ||c(x)||$. L'algorithme a de meilleures performances si deux pas sont considérés : α^k pour les direction d_x et d_λ et α_z^k pour la direction z . Soit $\tau_j \in]0, 1[$ vérifiant :

$$\tau_j = \max(\tau_{min}, 1 - \mu_j) \quad \text{où } \tau_{min} \in]0, 1[\text{ est la valeur minimale fixée.} \tag{4.4.8}$$

Les pas sont caractérisés par l'équation suivante :

$$\begin{cases} \alpha^k \in]0, \alpha_{max}^k] \quad \text{où} \quad \alpha_{max}^k & = \max\left\{\alpha \in]0, 1] \mid x^k + \alpha d_x^k \geq (1 - \tau_j) x^k\right\} \\ \qquad\qquad\qquad\qquad\qquad \alpha_z^k & = \max\left\{\alpha \in]0, 1] \mid x^k + \alpha d_z^k \geq (1 - \tau_j) z^k\right\} \end{cases} \tag{4.4.9}$$

Pour déterminer le pas α^k, il faut trouver un point d'essai $\tilde{x}^k = x^k + \alpha^{k,l} d_x^k$ acceptable. Pour cela, il doit vérifier les conditions suivantes :

$$\left|\begin{array}{lll} \theta(\tilde{x}^k) < (1-\gamma_\theta)\theta(x^k) & \text{avec} & \gamma_\theta \in]0,1[\text{ fixé}, \\ \text{ou } g_{\mu_j}(\tilde{x}^k) < g_{\mu_j}(x^k) - \gamma_g\theta(x^k) & \text{avec} & \gamma_g \in]0,1[\text{ fixé}, \end{array}\right. \tag{4.4.10}$$

ou alors si $\theta(\tilde{x}^k) \leq \theta_{min}$, (avec $\theta_{min} \in]0,\infty]$ fixé) :

$$\begin{cases} \nabla g_{\mu_j}(x^k)^T \cdot d_x^k < 0, \\ \alpha^{k,l}[-\nabla g_{\mu_j}(x^k)^T \cdot d_x^k]^{s_g} > [\delta\theta(x^k)]^{s_\theta}, \\ g_{\mu_j}(\tilde{x}^k) \leq g_{\mu_j}(x^k) + \eta_g \alpha^{k,l} \nabla g_{\mu_j}(x^k)^T d_x^k, \end{cases} \tag{4.4.11}$$

pour des paramètres $\delta > 0, s_\theta > 1, s_g \geq 1$ et $\eta_g \in \left]0, \frac{1}{2}\right[$.

Si le point d'essai ne vérifie pas les conditions des équations (4.4.10) et (4.4.11), il est ajouté au filtre. Au début de l'optimisation, le filtre est initialisé de la façon suivante :

$$\mathcal{F}_0 = \left\{(\theta(\tilde{x}^0), g(\tilde{x}^0)) \in \mathbb{R}^2 \text{ tel que } \theta(\tilde{x}^0) \geq \theta_{max}\right\} \quad \text{où } \theta_{max} \text{ est une constante.} \tag{4.4.12}$$

À chaque itération, le filtre est augmenté par

$$\mathcal{F}_{k+1} = \mathcal{F}_k \cup \left\{(\theta(\tilde{x}^k), g(\tilde{x}^k)) \in \mathbb{R}^2 \text{ tel que } \theta(\tilde{x}^k) \geq (1-\gamma_\theta)\theta(x^k) \text{ et } g(\tilde{x}^k) \geq f(x^k) - \gamma_g\theta(x^k)\right\}. \tag{4.4.13}$$

L'ajout des points non acceptables au filtre permet d'écarter de la zone de recherche les points de son voisinage. Dans le cas où il est impossible de dégager un point d'essai, d'autres stratégies sont mises en place comme une phase de restauration ou des méthodes de correction du second ordre. Tous les détails sont disponibles dans [Wächter et Biegler, 2006].

Une fois le pas déterminé, le nouvel itéré peut être calculé :

$$\begin{cases} x^{k+1} &= x^k + \alpha^k d_x^k \ ; \\ \lambda^{k+1} &= \lambda^k + \alpha^k d_\lambda^k \ ; \\ z^{k+1} &= z^k + \alpha_z^k d_z^k \ . \end{cases} \tag{4.4.14}$$

4.4.3 Les critères d'arrêt

L'algorithme propose une solution optimale une fois que le critère de convergence est vérifié. Ce critère est défini à partir de l'erreur d'optimalité du problème de barrière défini de la manière suivante :

$$E_\mu(x,\lambda,z) = \max\left\{\frac{||\nabla f(x) + \nabla c(x)\lambda - z||_\infty}{s_d}, ||c(x)||_\infty, \frac{||[X][Z][e] - \mu[e]||_\infty}{s_c}\right\} \tag{4.4.15}$$

où s_d et s_c sont des paramètres d'échelle tels que

$$s_d = \frac{1}{s_{max}} \max\left\{s_{max}, \frac{|\lambda| + |z|}{(m+n)}\right\}, \qquad s_c = \frac{1}{s_{max}} \max\left\{s_{max}, \frac{|z|}{n}\right\}, \qquad \text{avec } s_{max} = 100.$$

L'algorithme s'arrête si, pour une solution (x^*, λ^*, z^*), l'inégalité suivante est vérifiée :

$$E_0(x^*, \lambda^*, z^*) \leq \varepsilon_{tol} \tag{4.4.16}$$

avec ε_{tol} la tolérance fournie par l'utilisateur. Tant que la convergence n'a pas été atteinte et que les autres critères d'arrêts ne sont pas vérifiés (nombre maximal d'itérations excédé, limite du temps de

calcul, point d'essai infaisable, etc.), le paramètre de barrière μ_j est mis à jour et un nouveau problème de barrière est généré. Plus précisément, si le résultat du problème de barrière a convergé, c'est-à-dire que la condition :

$$E_{\mu_j}(x^*, \lambda^*, z^*) \leq \kappa_\varepsilon \mu_j, \quad (\text{ avec } \kappa_\varepsilon > 0 \text{ fixé })$$ (4.4.17)

est vérifiée, alors le nouveau paramètre de barrière est donné par :

$$\mu_{j+1} = \max\left\{\frac{\varepsilon_{tol}}{10}, \min\left\{\kappa_\mu \mu_j, \mu_j^{\theta_\mu}\right\}\right\}, \quad \kappa_\mu \in]0,1[, \quad \theta_\mu \in]1,2[.$$ (4.4.18)

4.5 Résultats d'optimisation

Nous avons vu dans les sections précédentes comment sont calculés les gradients et le fonctionnement de l'algorithme d'optimisation de IPOPT. À partir de ceci, des cas d'optimisation ont pu être traités. Nous présentons maintenant certains des résultats obtenus.

4.5.1 Un plateau soumis à son propre poids

Le premier cas étudié est l'optimisation d'un plateau de 2 m de longueur, 1 m de largeur et 0.5 m de hauteur. Il est constitué d'acier, dont le module de Young vaut 200 GPa, le coefficient de Poisson 0.3 et la masse volumique 8000 $kg.m^{-1}$. Le plateau a une épaisseur de 9 mm. La géométrie de la pièce est représentée sur la figure 4.7. La pièce a été modélisée avec 2 patchs NURBS de degrés 3. La forme est donc définie par 28 points de contrôle effectifs.

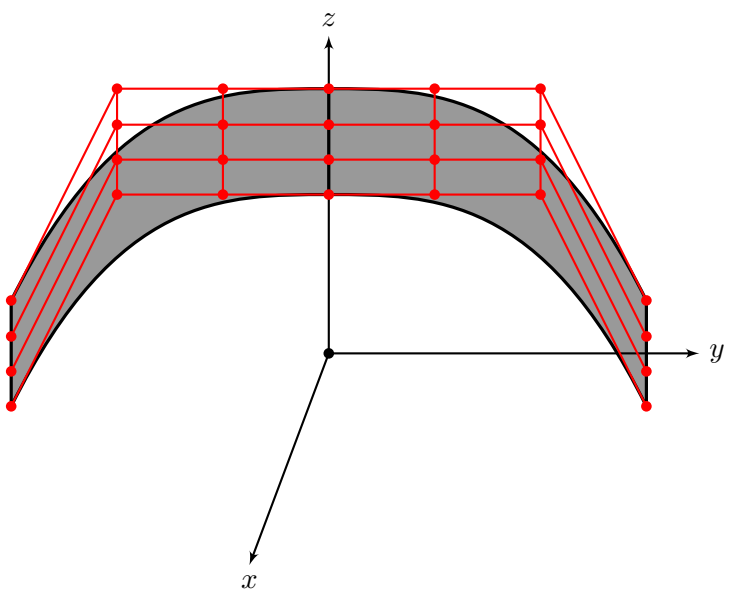
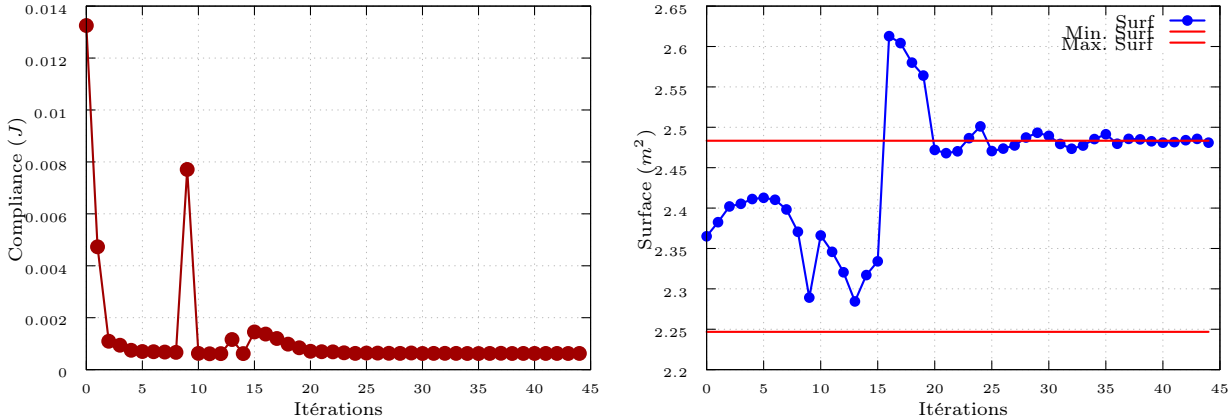


FIGURE 4.7: Géométrie du plateau avec la position des points de contrôle.

Le plateau est supposé encastré sur les deux arêtes droites ($z = 0$). Le chargement considéré est le poids propre de la structure. Les points de contrôle touchés par les conditions d'encastrement ne seront pas autorisés à se déplacer au cours de l'optimisation. Les variables d'optimisation pour ce cas sont les coordonnées en z des points de contrôle restant. Elles peuvent varier entre 0 et 2 m. Le problème d'optimisation compte donc 20 variables au total. Une variation d'aire de $\pm 5\%$ est tolérée.

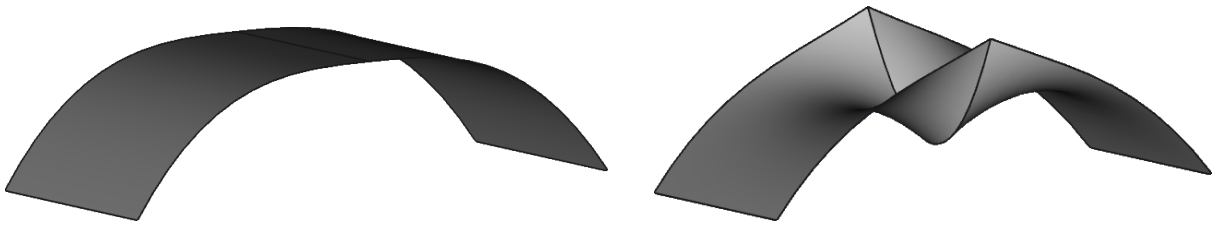
Les simulations ont été réalisées avec 920 éléments de discrétisation. La figure 4.8 donne l'évolution de la fonction coût et de l'aire en fonction des itérations de l'algorithme. L'algorithme a convergé après



(a) Compliance (b) Aire

FIGURE 4.8: Évolution de la compliance et de l'aire du plateau en fonction des itérations.

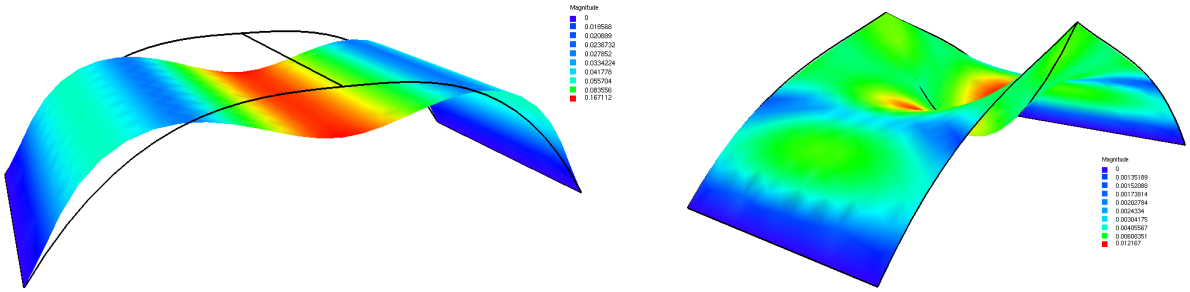
44 itérations. Il semble que le respect de la contrainte soit la raison des 24 dernières itérations. À la fin de l'optimisation, la compliance vaut environ 6.24×10^{-4} J contre 0.013 J au départ. L'aire pour sa part a augmenté, passant de 2.36507 m^2 à 2.48095 m^2. Une comparaison des formes avant et après optimisation est donnée en figure 4.9. L'algorithme a créé un raidisseur au niveau de la jonction afin de limiter les déformations. Le résultat obtenu rappelle les formes optimales présentées dans [Bletzinger *et al.*, 2010].



(a) Forme originale (b) Forme optimisée

FIGURE 4.9: Comparaison entre la forme originale du plateau et l'optimum trouvé.

La figure 4.10 témoigne de la diminution des déformations sur la forme optimisée. En effet, en rigidifiant le milieu du plateau, cette partie qui se déformait le plus ne bouge quasiment plus. De façon globale, le déplacement a été divisé par 10.



(a) Déformations avant optimisation (b) Déformations après optimisation

FIGURE 4.10: Comparaison des déformations du plateau (déplacements $\times100$).

4.5.2 Un cube sous pression

Pour ce cas, considérons un cube de dimension $1\ m \times 1\ m \times 1\ m$ avec un module de Young de 210 GPa et un coefficient de Poisson égal à 0.3. Comme le montre la figure 4.11, le cube, ouvert à ses bases ($z = 0$ et $z = 1$), est formé par 4 patchs NURBS de degré 2. La géométrie offre donc 24 points de contrôle pour faire varier la forme.

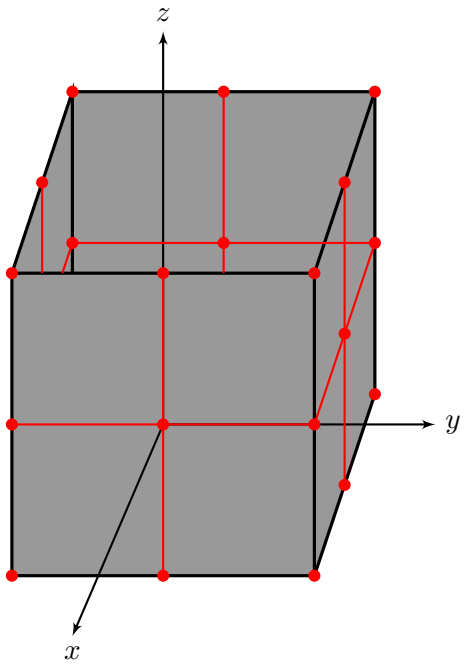


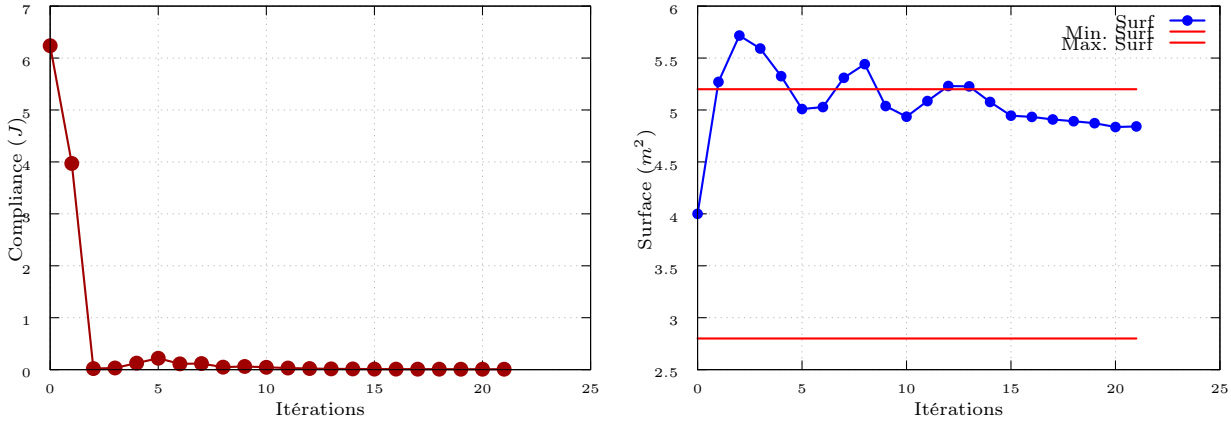
FIGURE 4.11: Géométrie du cube avec la position des points de contrôle.

Le cube est encastré à ses bases et est soumis à une pression uniformément répartie de 2000 N. Les variables d'optimisation sont les coordonnées en x, en y et les poids de tous les points de contrôle. Les positions des points de contrôle situés sur les zones d'encastrement sont autorisées à varier. Ce problème compte alors 72 variables d'optimisation. Afin d'éviter le scénario de la figure 4.1b, des bornes sont imposées à tous les points de contrôle. En pratique, leur position ne peut pas se rapprocher du centre du cube. D'autre part les points ne peuvent se déplacer que de 1 m vers l'extérieur du cube. En plus de cela, une contrainte sur la variation d'aire est appliquée. L'aire finale doit être comprise entre $\pm 30\%$ de l'aire originale. Le calcul éléments finis est effectué sur 1840 éléments de discrétisation.

L'algorithme a convergé après 21 itérations. La compliance initiale de 6.24 J a diminué pour atteindre 0.0087 J. Dans le même temps, l'aire a augmenté passant de 4 m^2 à 4.84166 m^2 soit une augmentation de 21%. L'évolution de la fonction objectif et de la surface au cours de l'optimisation sont présentées respectivement en figures 4.14a et 4.14b.

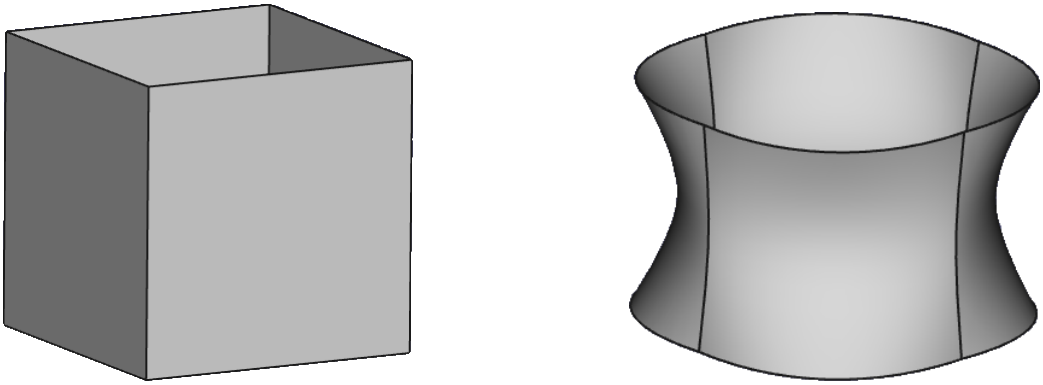
L'algorithme a fait rapidement chuter la fonction objectif dans un premier temps. Il a ensuite cherché à respecter la contrainte d'aire. Autour de l'itération 13, la contrainte étant de nouveau respectée, l'algorithme a amélioré la compliance jusqu'à convergence. La figure 4.13 donne une comparaison entre la forme d'origine et la forme optimisée. L'algorithme a cherché à « arrondir » la forme afin d'offrir une meilleure résistance à la structure face à la pression.

La figure 4.14 compare les déformations pour la forme d'origine et la forme optimisée. Les résultats mettent en évidence que les déformations ont été fortement limitées par l'optimisation.



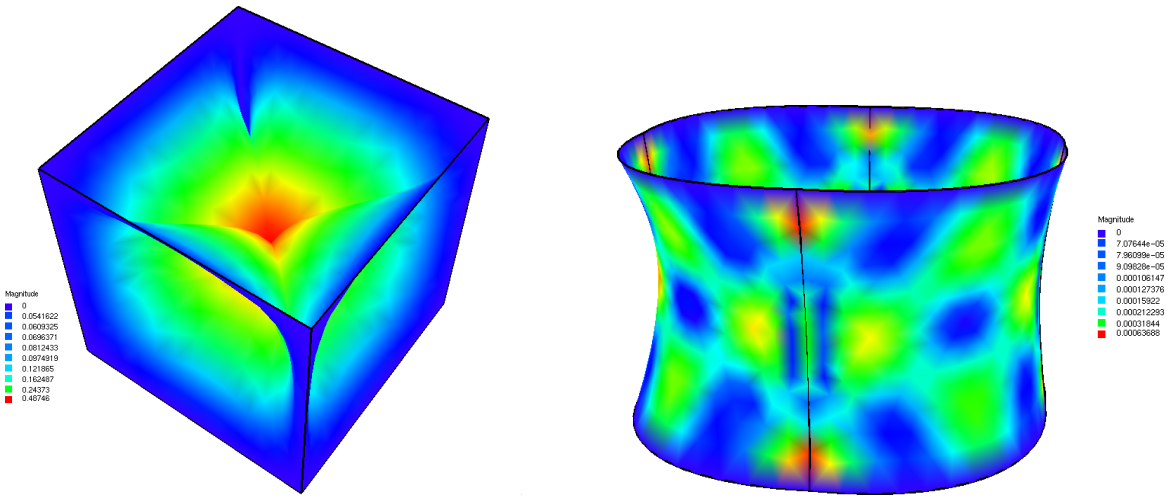
(a) Compliance (b) Aire

FIGURE 4.12: Évolution de la compliance et de l'aire en fonction des itérations pour le cube sous pression.



(a) Forme originale (b) Forme optimisée

FIGURE 4.13: Comparaison entre la forme originale et l'optimum trouvé.

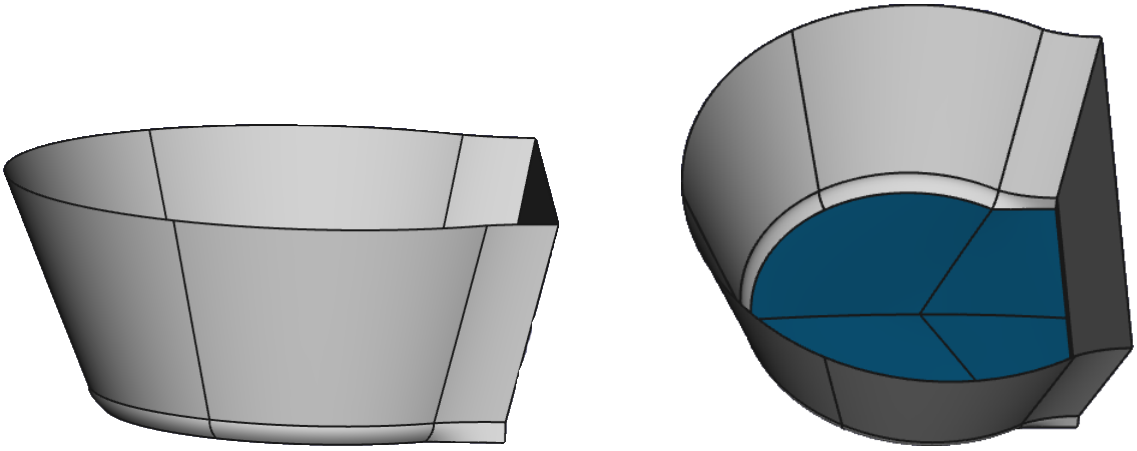


(a) Déformations du cube d'origine (b) Déformations du cube optimisé

FIGURE 4.14: Comparaison des déformations avant et après optimisation (déplacement $\times 100$).

4.5.3 Un bac de roue de secours

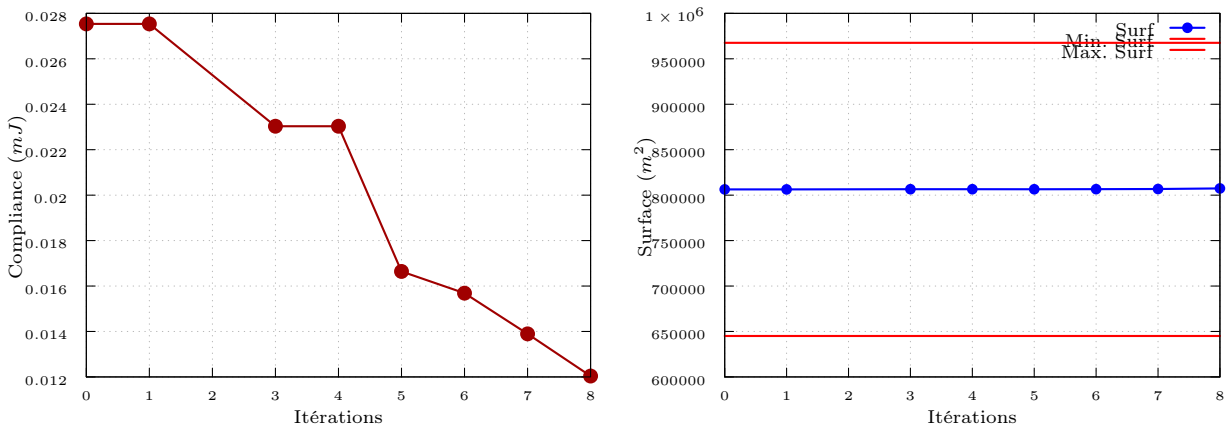
Le dernier cas traité ici est l'optimisation de la forme d'un bac de roue de secours. Le bac, épais de 3 mm, est en acier avec un module de Young de 210 GPa et un coefficient de Poisson 0.3. Il est soumis à la force exercée par une roue de secours de 10 kg. Le cahier des charges autorise uniquement une modification maximale de 8 mm selon la normale du fond du bac , modélisé en bleu sur la figure 4.15b. La géométrie du bac, présentée en figure 4.15, comprend 18 patchs NURBS avec un total de 210 points de contrôle. Le bac est encastré sur son bord supérieur.



(a) (b)

FIGURE 4.15: Géométrie du bac de roue.

La partie autorisée à être modifiée par l'optimisation comporte 4 patchs NURBS de degré 4 soit 64 points de contrôle. Le problème d'optimisation compte 49 variables d'optimisation au total. Concernant l'aire totale, une variation de ±20% est autorisée. Les simulations ont été réalisées avec 8534 éléments de discrétisation.



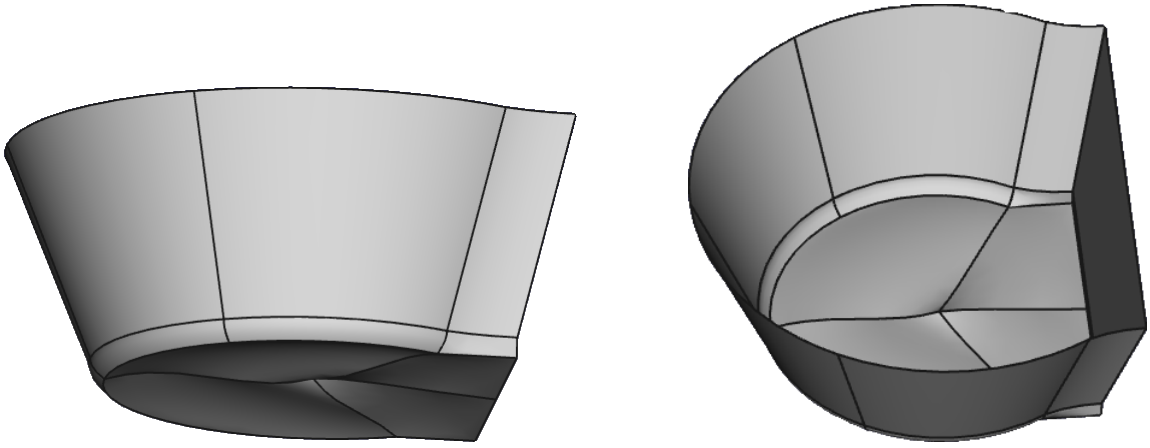
(a) Compliance (b) Aire

FIGURE 4.16: Évolution de la compliance et de l'aire pour le problème du bac de roue en fonction des itérations.

Comme le montre la figure 4.16a l'algorithme s'est arrêté à la 8ème itération. La compliance a diminué d'environ 43% de sa valeur d'origine alors que l'aire a augmenté de 0.13% dans le même temps. L'arrêt rapide de l'algorithme et la faible variation d'aire s'expliquent principalement par les bornes très

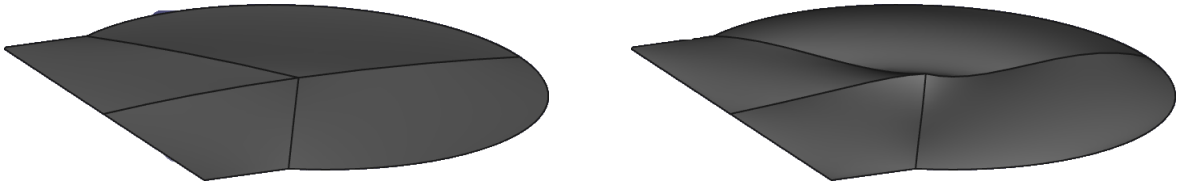
rapprochées imposées à la position des points de contrôle.

Les figures 4.17 présentent la forme optimale proposée par l'algorithme. La comparaison en figure 4.18 entre le fond du bac d'origine et celui optimisé met en évidence que l'algorithme a cherché à courber le fond du bac pour le rigidifier.



(a) (b)

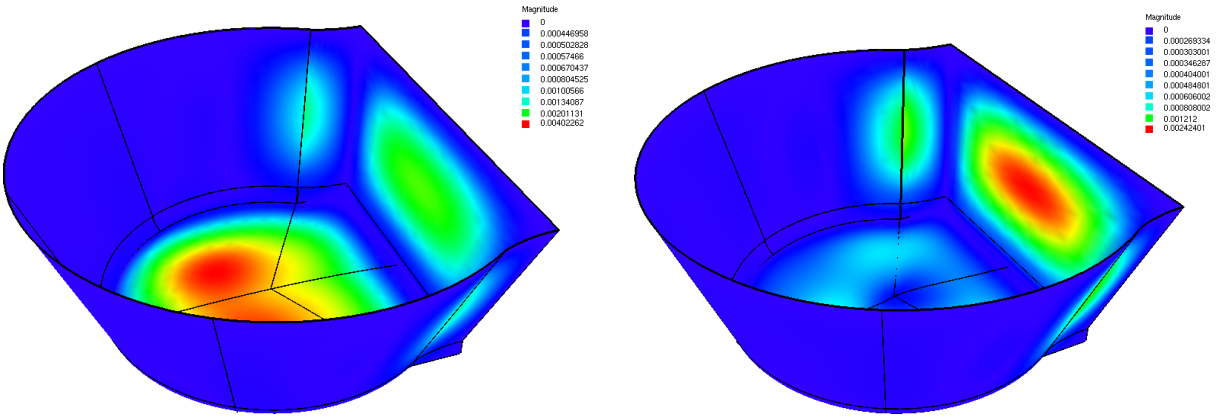
FIGURE 4.17: Forme du bac de roue à l'issue de l'optimisation.



(a) Avant optimisation (b) Après optimisation

FIGURE 4.18: Forme du fond du bac de roue.

De façon générale, la comparaison entre les déformations avant et après optimisation de la figure 4.19 indique que celles-ci sont moins importantes pour la forme optimisée. Toutefois, on peut remarquer que la zone qui se déforme le plus se situe désormais sur la partie droite du bac alors qu'elle était concentrée sur le fond auparavant.



(a) Déformations du bac d'origine (b) Déformations du bac optimisé

FIGURE 4.19: Comparaison des déformations avant et après optimisation du bac de roue.

4.6 Synthèse du chapitre

L'objectif de ce chapitre était la mise en place de la stratégie d'optimisation pour un critère de compliance dans le cas de géométries complexes. L'optimisation a été implémentée dans le code OptSurf au moyen de la libraire Ipopt. La définition des différents objets pour le fonctionnement du code sont disponibles en annexe A. À travers différents problèmes d'optimisation, les calculs de gradient par la méthode de l'état adjoint ont été introduits et validés par comparaison avec des approximations par différences finies. Grâce à ce principe de différentiation, différents cas d'optimisation de formes avec un grand nombre de paramètres, ont peu être réalisés, y compris un cas test industriel.

L'implémentation de l'optimisation a soulevé quelques questions qui devront être résolues, dans le futur, en particulier vis à vis de la conception des modèles CAO avec des patchs non conformes ou la traduction des contraintes sur les points de contrôle. D'autre part, il faut garder à l'esprit que le modèle CAO de la forme de départ joue un rôle important. En effet, le nombre et la disposition des points de contrôle modifient l'espace de conception et *a fortiori* les formes optimales possibles.

D'un point de vue purement opérationnel, le code devra également être amélioré. L'optimisation du bac de roue a nécessité 7 heures et 43 minutes sur 12 processeurs et 120 Go de RAM, une configuration qui semble minimale au bon fonctionnement du programme. Par conséquent, le traitement de pièces industrielles plus complexes nécessitera une implémentation informatique plus légère.

Dans ce chapitre, le critère de compliance était posé pour un problème statique. Le chapitre suivant propose une stratégie pour un critère différent, lié cette fois à la réponse dynamique des coques.

CHAPITRE

5

Optimisation de formes de coques pour un critère dynamique

Sommaire

5.1 Introduction **88**
5.2 Définition de la réponse forcée **88**
5.2.1 Formulation variationnelle de la réponse forcée 88
5.2.2 Définition de l'amortissement 90
5.2.2.1 L'amortissement structural 90
5.2.2.2 L'amortissement visqueux 90
5.3 Méthodes de résolution numérique **91**
5.3.1 La méthode directe 91
5.3.2 La méthode de superposition modale 91
5.3.3 La méthode des accélérations modales 92
5.4 Implémentation de la réponse forcée harmonique **93**
5.4.1 Identification des coefficients pour l'amortissement 93
5.4.2 Validation de l'implémentation 93
5.5 Optimisation de formes en régime harmonique **95**
5.5.1 Définition du critère 95
5.5.2 Calcul du gradient 96
5.5.2.1 Différentielles des fonctions j_k 96
5.5.2.2 Dérivée de l'amortissement 98
5.5.3 Validation du gradient 99
5.5.4 Résultats d'optimisation 101
5.6 Méthode de résolution numérique en régime transitoire **102**
5.6.1 Motivations 102
5.6.2 Résolution numérique 102
5.6.3 Résultats de benchmarks 104
5.6.3.1 Plaque soumise à une pression 104
5.6.3.2 Calotte sphérique 105
5.7 Synthèse du chapitre **106**

5.1 Introduction

L'étude des comportements vibratoires des structures est essentielle lors de la création des produits. Dans le secteur automobile, ces phénomènes sont considérés comme étant l'une des principales causes du bruit perçu à l'intérieur et à l'extérieur d'un habitacle. Les vibrations des pièces, répétées au cours du temps, peuvent également être à l'origine de la casse de ces dernières. L'analyse vibratoire joue donc un rôle majeur lors de la conception des structures notamment pour des raisons vibro-acoustiques et de tenue à la fatigue. L'une des premières approches dans le but d'améliorer les prestations acoustiques, est l'optimisation de la réponse dynamique de ces structures. Nous verrons dans cette partie comment est résolu le problème de réponse forcée avec un modèle de Koiter puis la définition d'un critère d'optimisation et enfin la procédure numérique associée. La dernière partie sera consacrée à la résolution du problème en régime transitoire.

5.2 Définition de la réponse forcée

Dans le cadre des études vibratoires, deux types de phénomènes sont particulièrement étudiés. Le premier est l'étude des vibrations libres. Elle consiste à identifier les fréquences propres et les modes propres de la structure, c'est-à-dire sans aucune sollicitation extérieure. Par opposition, le second phénomène est la réponse forcée. La réponse forcée correspond au comportement vibratoire d'une structure lorsqu'elle est soumise à une excitation extérieure. Il existe plusieurs natures de réponse forcée, chacune définie par la forme de la sollicitation appliquée. Les catégories souvent considérées sont les suivantes :

- la réponse harmonique due à une excitation harmonique ;
- la réponse périodique résultat d'une somme d'excitations harmoniques ;
- la réponse transitoire liée à une excitation dépendant du temps ;
- la réponse à une sollicitation aléatoire où la force appliquée est définie par des moyens statistiques.

Le critère d'optimisation dont il sera question ici, est en lien avec la réponse harmonique de la structure étudiée. Avant d'aborder le problème d'optimisation, rappelons dans un premier temps comment déterminer cette réponse dynamique.

5.2.1 Formulation variationnelle de la réponse forcée

Considérons une coque $\mathcal{C} \in \mathcal{E}^3$ encastrée sur une partie de sa frontière $\partial\mathcal{C}_0$ et chargée sur $\partial\mathcal{C}_1$ la partie complémentaire de sa frontière. Soit $\vec{U}(\xi, t)$ le champ de déplacement de la coque tridimensionnelle au temps t. Soit $\vec{u} = u_i \vec{a}^i$ le champ de déplacement de la surface moyenne. D'après les hypothèses du modèle de Koiter, $\vec{U}(\xi, t)$ est donné par :

$$\vec{U} = u_i \vec{a}^i - \xi_3 (u_{3,\alpha} + b^\lambda_\alpha u_\lambda) \vec{a}^\alpha. \tag{5.2.1}$$

L'équilibre de la coque est régie par les équations dynamiques d'élasticité :

$$\rho \frac{\partial^2 \vec{U}}{\partial t^2} + W\left(\frac{\partial \vec{U}}{\partial t}\right) = div(\underline{\underline{\sigma}}) + \vec{p}(t) \tag{5.2.2}$$

avec $\underline{\underline{\sigma}} = \underline{\underline{C}} : \underline{\underline{\varepsilon}}$ où $\underline{\underline{\sigma}}, \underline{\underline{\varepsilon}}$ et $\underline{\underline{C}}$ sont respectivement les tenseurs des contraintes, des déformations et d'élasticité de la coque tridimensionnelle, $\vec{p}$ est la résultante des forces volumiques appliquées à la

coque et W, est une loi de comportement modélisant l'amortissement fonction de la vitesse $\dfrac{\partial \vec{U}}{\partial t}$. À l'équation (5.2.2), s'ajoutent les conditions aux limites et initiales suivantes :

$$\begin{cases} \vec{U} = \vec{0} & \text{sur } \partial\mathcal{C}_0 \quad (5.2.3a) \\ \underline{\underline{\sigma}}.\vec{n} = \vec{T} & \text{sur } \partial\mathcal{C}_1 \quad (5.2.3b) \\ \vec{U}(\xi,0) = \vec{U}_0(\xi)\ ;\ \dfrac{d\vec{U}}{dt}(\xi,0) = \vec{U}_1(\xi) & (5.2.3c) \end{cases}$$

avec $\vec{T}$ la résultante des forces surfaciques.

En multipliant par $\vec{V} \in \vec{\mathcal{V}} = \{\vec{v} \in (H^1(\mathcal{C}))^3, \vec{v}_{|\partial\mathcal{C}_0} = \vec{0}\}$ et en intégrant sur la coque, les équations deviennent :

$$\int_{\mathcal{C}} \frac{\partial^2 \vec{U}}{\partial t^2}\vec{V}\ d\mathcal{C} + \int_{\mathcal{C}} W\left(\frac{\partial \vec{U}}{\partial t}\right)\vec{V}\ d\mathcal{C} + \int_{\mathcal{C}} C^{ijkl}\varepsilon_{ij}(\vec{U})\varepsilon_{kl}(\vec{V})\ d\mathcal{C} = \int_{\mathcal{C}} \vec{p}(t).\vec{V}\ d\mathcal{C} + \int_{\partial\mathcal{C}_1} \underline{\underline{\sigma}}.\vec{V}.\vec{n}\ dS \quad (5.2.4)$$

Finalement, en ramenant les différents termes sur la configuration de référence Ω selon les hypothèses du modèle de Koiter, nous avons respectivement, pour les termes de masse, rigidité et forces de l'équation précédente :

$$\begin{aligned} b(\vec{u},\vec{v}) = \int_\Omega \rho e \Bigg\{ & \left[1 + \frac{e^2}{12}(b_1^1 b_2^2 - b_1^2 b_2^1)\right]\left[a^{\alpha\beta}\ddot{u}_\alpha v_\beta + \ddot{u}_3 v_3\right] \\ & + \frac{e^2}{12}a^{\alpha\beta}\left[(\ddot{u}_{3|\alpha} + b_\alpha^\lambda \ddot{u}_\lambda)(v_{3|\beta} b_\beta^\mu v_\mu) + (\ddot{u}_\alpha v_{3|\beta} + \ddot{u}_{3|\alpha} v_\beta + 2b_\alpha^\lambda \ddot{u}_\lambda v_\beta) b_\eta^\eta\right]\Bigg\}\sqrt{a}\ d\xi_1 d\xi_2 \end{aligned} \quad (5.2.5)$$

$$a(\vec{u},\vec{v}) = \int_\Omega eE^{\alpha\beta\lambda\mu}\left[\gamma_{\alpha\beta}(\vec{u})\gamma_{\lambda\mu}(\vec{v}) + \frac{e^2}{12}\rho_{\alpha\beta}(\vec{u})\rho_{\lambda\mu}(\vec{v})\right]\sqrt{a}\ d\xi_1 d\xi_2 \quad (5.2.6)$$

$$f(\vec{v}) = \int_\Omega \vec{p}(t)\vec{v}\sqrt{a}\ d\xi_1 d\xi_2 + \int_{\Gamma_1}\left[\vec{N}(t)\vec{v} + M^\alpha(t)(v_{3,\alpha} + b_\alpha^\lambda v_\lambda)\right]\ d\gamma \quad (5.2.7)$$

Ainsi, en notant $c(\vec{u},\vec{v})$ le terme restant, le problème à résoudre pour déterminer la réponse forcée du système s'exprime de la manière suivante :

$$\begin{aligned} & \text{trouver une fonction } t \to \vec{u}(\xi,t) \text{ de [0,T] dans } \vec{\mathcal{W}} \text{ telle que} \\ & \forall \vec{v} \in \vec{\mathcal{W}},\ \forall t \in [0,T], \quad a(\vec{u},\vec{v}) + b(\vec{u},\vec{v}) + c(\vec{u},\vec{v}) = f(\vec{v}) \end{aligned} \quad (5.2.8)$$

où

$$\vec{\mathcal{W}} = \left\{\vec{w} \in (H^1(\Omega))^2 \times H^2(\Omega), \quad \vec{w}_{|\Gamma_0} = \vec{0}, \quad \frac{\partial w_3}{\partial \vec{n}}_{\Gamma_0} = 0\right\} \quad (5.2.9)$$

L'équation (5.2.8) donne la formulation variationnelle générale pour un problème de réponse forcée. Elle peut être adaptée en fonction du type d'excitation. Le critère que nous allons considérer dans la suite est basé sur une réponse harmonique de la structure. La particularité de ce type de réponse forcée est que la sollicitation est supposée harmonique et qu'en réponse, la structure vibre à la même fréquence que celle de l'excitation. En pratique, on suppose que la structure est excitée par une force de pulsation ω :

$$\vec{p}(\xi,t) = \mathcal{R}e(p_j(\xi)e^{i\omega t})\vec{a}^j$$

où $\mathcal{R}e$ désigne la partie réelle et i est le nombre imaginaire pur. Le déplacement sera lui de la forme suivante :

$$\vec{u}(\xi,t) = \mathcal{R}e(u_j(\xi)e^{i\omega t})\vec{a}^j.$$

En tenant compte des définitions du déplacement et de la force exercée, le problème de réponse forcée harmonique étudié s'exprime alors de la manière suivante :

$$\text{trouver } \vec{u} \in \vec{\mathcal{W}} \quad \text{tel que} \quad \forall \vec{v} \in \vec{\mathcal{W}} \quad a(\vec{u}, \vec{v}) - \omega^2 \tilde{b}(\vec{u}, \vec{v}) + c(\vec{u}, \vec{v}) = \tilde{f}(\vec{v}) \tag{5.2.10}$$

avec :

$$\begin{aligned} \tilde{b}(\vec{u}, \vec{v}) = \int_\Omega \rho e \Bigg\{ & \left[1 + \frac{e^2}{12}(b_1^1 b_2^2 - b_1^2 b_2^1)\right]\left[a^{\alpha\beta} u_\alpha v_\beta + u_3 v_3\right] \\ & + \frac{e^2}{12} a^{\alpha\beta}\left[(u_{3|\alpha} + b_\alpha^\lambda u_\lambda)(v_{3|\beta} b_\beta^\mu v_\mu) + (u_\alpha v_{3|\beta} + u_{3|\alpha} v_\beta + 2 b_\alpha^\lambda u_\lambda v_\beta) b_\eta^\eta\right]\Bigg\} \sqrt{a}\, d\xi_1 d\xi_2 \end{aligned} \tag{5.2.11}$$

$$\tilde{f}(\vec{v}) = \int_\Omega p_j v_j \sqrt{a}\, d\xi_1 d\xi_2. \tag{5.2.12}$$

Remarque 5.2.1. L'équation (5.2.10) est en fait définie avec des variables complexes. Une fois le déplacement calculé, seule la partie réelle est conservée.

Afin de compléter la formulation variationnelle de l'équation (5.2.10), il faut définir une loi pour le terme d'amortissement. Cette définition fait l'objet du paragraphe suivant.

5.2.2 Définition de l'amortissement

L'amortissement au sein d'un système engendre une dissipation de l'énergie qui atténue le mouvement de la structure. Pour le simuler en ingénierie, deux modèles sont souvent employés. Le premier s'appelle l'amortissement structural et le second est l'amortissement visqueux.

5.2.2.1 L'amortissement structural

L'amortissement structural, ou amortissement hystérétique, permet une dissipation proportionnelle au déplacement de la structure. Il s'écrit :

$$c(\vec{u}, \vec{v}) = i\eta a(\vec{u}, \vec{v}) \tag{5.2.13}$$

où η est le coefficient d'amortissement hystérétique. Le coefficient η se détermine à partir de l'énergie dissipée par un cycle de vibration. Cette forme d'amortissement peut être utilisée uniquement lorsqu'il s'agit de réponse forcée harmonique. Ce modèle, qui est simple à mettre en œuvre, présente quelques inconvénients, le principal étant qu'il ne prend pas en compte la fréquence d'excitation alors que le comportement de la structure est en lien direct avec celle-ci. En effet, plus la fréquence d'excitation est proche d'une fréquence de résonance, plus la structure se déforme. Nous préférerons donc la seconde modélisation qui reste la plus répandue.

5.2.2.2 L'amortissement visqueux

Avec l'amortissement visqueux, l'énergie est dissipée de façon proportionnelle à la vitesse du mouvement. Il est défini comme étant une combinaison des termes de rigidité et masse :

$$c(\vec{u}, \vec{v}) = i\omega[\beta a(\vec{u}, \vec{v}) + \alpha \tilde{b}(\vec{u}, \vec{v})]. \tag{5.2.14}$$

Les coefficients α et β sont appelés les coefficients de Rayleigh. Ces coefficients permettent de définir le terme d'amortissement comme un pourcentage d'amortissement critique par rapport à un mode.
Les coefficients α et β ne sont généralement pas connu *a priori*. Ils nécessitent souvent un calcul modal afin de les estimer.

Les différents termes du problème de réponse forcée étant explicités, la section suivante vise à présenter les méthodes de résolution numérique de ce type de problème. Davantage de détails en rapport avec la théorie et la formulation des problèmes de réponse forcée sont disponibles dans des ouvrages comme [Ohayon et Soize, 1997] ou [Choi et Kim, 2006].

5.3 Méthodes de résolution numérique

Dans un premier temps, nous allons exprimer l'équation (5.2.8) sous forme discrétisée avant de présenter les deux méthodes de résolution possibles : la méthode directe ou par superposition modale. Des résultats de benchmark pour les problèmes de coque en réponse forcée seront ensuite discutés.

5.3.1 La méthode directe

Le problème de réponse forcée générale peut s'écrire de la manière suivante sous forme matricielle :

$$M\ddot{U} + C\dot{U} + KU = F \tag{5.3.1}$$

où M la matrice de masse, C celle de l'amortissement, K est la matrice de rigidité, F le vecteur force et U la réponse de la structure. Dans le cas de la réponse forcée harmonique, la force et la réponse peuvent être mises respectivement sous la forme $F = fe^{i\omega t}$ et $U = ue^{i\omega t}$. Le problème discrétisé devient alors :

$$-\omega^2 Mu + i\omega Cu + Ku = f. \tag{5.3.2}$$

Rappelons que l'amortissement est de type visqueux et que par conséquent $C = \alpha M + \beta K$. La première méthode de résolution consiste à résoudre de façon classique le système linéaire de l'équation (5.3.2). Cette méthode bien qu'efficace présente un désavantage majeur. Lors des analyses en réponse forcée, la réponse de la structure est généralement étudiée sur une bande de fréquence. Ce type d'étude nécessite donc une discrétisation en fréquence et par suite, la résolution du système (5.3.2) pour chacune d'elles. Pour des matrices de taille importante et une discrétisation en fréquence très fine, le temps de calcul de la réponse devient extrêmement long. Cette solution n'est donc pas la plus recommandée, en particulier pour un processus d'optimisation, où il peut y avoir de nombreux appels à la fonction d'objectif. Pour cette raison, nous nous tournerons vers la méthode de superposition modale.

5.3.2 La méthode de superposition modale

La méthode de superposition modale consiste à décomposer le problème sur la base modale du système libre de toute sollicitation. En pratique, une base modale tronquée est utilisée et le système est réduit à un nombre d'équations découplées inférieur à la taille du système original. Des explications complètes sur cette méthode sont proposées dans [Géradin et Rixen, 1996] et [Petyt, 2010].

Soit $\Phi = \{\phi_1, ..., \phi_n\}$ la base modale composée de n vecteurs propres tels que :

$$K\phi_j = \omega_j^2 M\phi_j, \quad \text{avec} \quad \phi_j^T M\phi_j = 1. \tag{5.3.3}$$

Les matrices K et M se diagonalisent simultanément dans la base $\{\phi_1, .., \phi_n\}$ de la façon suivante :

$$\Phi^T K\Phi = \begin{bmatrix} \omega_1^2 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \omega_n^2 \end{bmatrix}, \qquad \Phi^T M\Phi = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix}. \tag{5.3.4}$$

La méthode de superposition modale consiste à exploiter cette propriété :

Méthode numérique 5.3.1 (méthode de superposition modale)**.** On cherche à résoudre :

$$-\omega^2 Mu + i\omega Cu + Ku = f \quad \text{avec} \quad C = \alpha M + \beta K$$

Soit Φ une base modale tronquée de taille $m << n$. Posons $u = \Phi q$. Alors :

$$-\omega^2 Mu + i\omega Cu + Ku = f \quad \Longleftrightarrow \quad -\omega^2 M\Phi q + i\omega C\Phi q + K\Phi q = f.$$

En multipliant par Φ^T à gauche :

$$-\omega^2\Phi^T M\Phi q + i\omega\Phi^T C\Phi q + \Phi^T K\Phi q = \Phi^T f$$

il suffit de résoudre les m équations découplées :

$$\boxed{(-\omega^2 + i\omega(\alpha + \beta\omega_j^2) + \omega_j^2)q_j = \phi_j \cdot f.} \tag{5.3.5}$$

Cette méthode justifie en partie le choix des définitions des amortissements. En effet, qu'il soit structural ou visqueux, l'amortissement génère une matrice qui vérifie systématiquement l'hypothèse de Basile. En d'autres termes, par sa définition, la matrice d'amortissement se diagonalise toujours dans la même base que celle des matrices de masse et raideur. La méthode de décomposition modale offre également la possibilité de définir un amortissement en fonction de chaque mode. Effectivement, plutôt que de choisir des α et β globaux, il est possible de les définir pour chacun des modes. Dans [Adhikari, 2006], l'auteur propose une écriture de la matrice C correspondante. Dans le cadre de notre étude, nous choisirons toujours des coefficients de Rayleigh globaux pour pouvoir être en mesure de calculer les gradients.

5.3.3 La méthode des accélérations modales

La convergence de la solution avec la méthode de superposition modale n'est pas toujours assurée. En effet, il a été montré que la convergence dépend principalement des modes de vibration non retenus [Géradin et Rixen, 1996]. En particulier, si le nombre de modes extraits n'est pas suffisant, la solution sera mal approximée. Cependant, ces résultats peuvent être améliorés par la méthode des accélérations modales. Celle-ci propose d'enrichir la solution calculée avec la réponse statique du problème. En d'autres termes, la réponse statique complète la solution tronquée en remplaçant les termes manquants.

Méthode numérique 5.3.2 (méthode des accélarations modales)**.** Avec les mêmes notations que précédemment, la solution statique du problème s'écrit :

$$u = K^{-1}\left(f - i\omega\sum_{j=1}^{m} C\phi_j q_j + \omega^2\sum_{j=1}^{m} M\phi_j q_j\right)$$

D'après l'équation (5.3.5), on prend alors pour nouvelle valeur de u :

$$\boxed{u = \left(K^{-1} + \sum_{j=1}^{m}\phi_j\frac{\omega^2 + i\omega(\alpha + \beta\omega_j^2)}{\omega_j^2(\omega_j^2 + i\omega(\alpha + \beta\omega_j^2) - \omega^2)}\phi_j^T\right)\cdot f.} \tag{5.3.6}$$

La méthode des accélérations modales peut souffrir des mêmes limites que la méthode de superposition classique, à savoir une base tronquée pas assez représentative. Elle permet néanmoins dans la plupart des cas d'obtenir de meilleurs résultats que la méthode de superposition modale simple.

Remarque 5.3.1. La plupart des éléments présentés jusqu'ici sont à la fois valables pour une coque $\mathcal{C}$ composée d'un seul ou plusieurs patchs. Les seules modifications à effectuer dans le cas multi-patchs, sont la définition du terme de rigidité qui doit être écrit pour chaque patch en incluant les contributions des charnières à l'image de l'équation (4.2.4) et, le fait de sommer les termes de masse et d'amortissement de chaque patch.

5.4 Implémentation de la réponse forcée harmonique

La résolution du problème de réponse forcée a été implémentée pour des problèmes avec des amortissements visqueux. Cette analyse peut être aussi bien réalisée pour une fréquence donnée que sur une bande de fréquences. Dans le cas d'une réponse sur une fréquence unique, le problème est résolu de façon directe (équation (5.3.2)). Si l'analyse est demandée pour une bande de fréquences, alors la méthode des accélérations modales est utilisée.

Dans un premier temps, nous traiterons l'identification des coefficients de Rayleigh α et β pour l'amortissement visqueux avant de présenter des résultats de benchmarks.

5.4.1 Identification des coefficients pour l'amortissement

Le principe de l'amortissement visqueux est de définir l'amortissement comme une combinaison linéaire des matrices de masse et de raideur. Pour l'exprimer, il faut alors choisir deux coefficient de Rayleigh α et β. Ces coefficients sont généralement déterminés à l'aide de la relation suivante :

$$\alpha = \zeta\omega \text{ et } \beta = \frac{\zeta}{\omega} \quad \text{où } \zeta = \frac{1}{2}\left(\frac{\alpha}{\omega} + \beta\omega\right) \text{ est un facteur de proportionnalité du mode de pulsation } \omega. \tag{5.4.1}$$

Ils permettent ainsi de modéliser un amortissement comme un pourcentage de l'amortissement critique par rapport à un mode de déformation. Afin de généraliser la procédure, l'amortissement sera toujours défini à partir du premier mode de déformation présent dans la bande de fréquence étudiée. Si la réponse forcée est étudiée pour une seule fréquence donnée, alors l'amortissement est déterminé à partir du mode de déformation le plus proche de cette dernière.

5.4.2 Validation de l'implémentation

Le benchmark le plus populaire en réponse forcée harmonique pour des structures minces est celui proposé par la NAFEMS (*National Agency for Finite Elements Methods and Standards*) dans [Maguire *et al.*, 1989]. Ce cas test consiste à étudier la réponse harmonique d'une plaque simplement posée sur ses quatre côtés soumise à une pression harmonique $F = F_0 \sin(\omega t)$. La réponse est évaluée sur une bande de fréquence allant de 0 Hz à 4.16 Hz. Pour ce cas, la plaque a été modélisée avec 4 patchs NURBS de degré 3. Les paramètres matériau et la géométrie sont indiqués en figure 5.1.

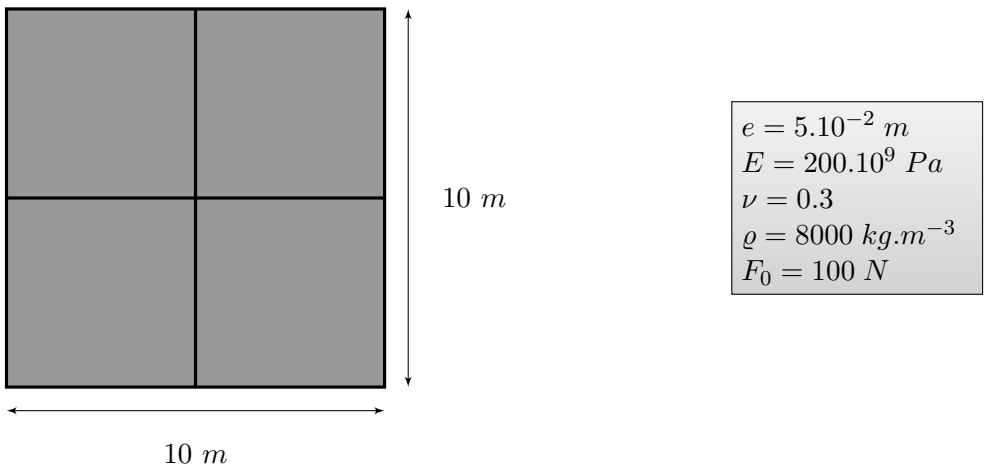


FIGURE 5.1: Géométrie et paramètres matériau de la plaque du cas test NAFEMS.

D'après les données du benchmark, dans la bande de fréquence étudiée se trouve une seule fréquence propre égale à 2.377 Hz. Par ailleurs, l'amortissement équivaut à 2% du premier mode. Les coefficients de Rayleigh donnés sont donc $\alpha = 0.299\ s^{-1}$ et $\beta = 1.339 \times 10^{-3}\ s$. La valeur de référence à retrouver est le déplacement selon z au centre de la plaque à la fréquence propre 2.377 Hz. La valeur théorique donnée est $u_z = 45.42\ mm$.

La simulation a été réalisée avec 792 éléments de discrétisation et la bande fréquences a été discrétisée en 50 échantillons. La base modale tronquée comprend seulement 3 modes. La figure 5.2 donne le déplacement en z, noté u_z, au centre de la plaque en fonction des fréquences des excitations. Dans un premier temps, un calcul modal a permis de retrouver précisément les valeurs théoriques de α et β pour $\zeta = 0.02$. Comme attendu, une forte variation apparaît à la fréquence propre présente dans la bande d'étude.

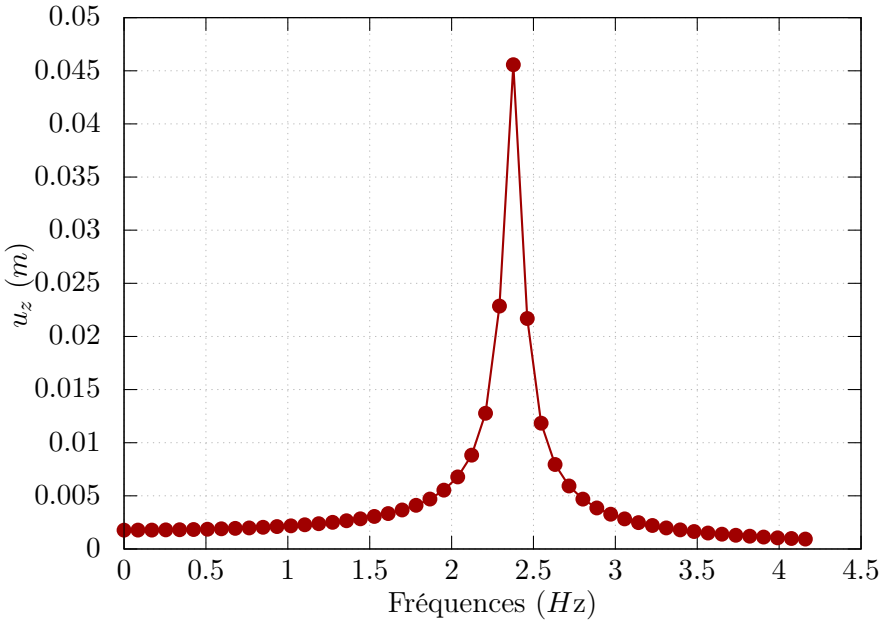


FIGURE 5.2: Réponse au centre de la plaque en fonction des fréquences d'excitation.

Le tableau 5.2 offre une comparaison entre les valeurs théoriques du benchmark et le calcul avec OPTSURF.

	OPTSURF	Théorique	Erreur Relative
1ère fréquence propre	2.3764 Hz	2.377 Hz	0.042%
α	0.2986 s^{-1}	0.299 s^{-1}	0.12%
β	$1.3394 \times 10^{-3}\ s$	$1.339 \times 10^{-3}\ s$	0.03%
u_z	45.57 mm	45.42 mm	0.33%

TABLEAU 5.1: Comparaison entre les valeurs théoriques et calculées pour le benchmark de réponse forcée harmonique.

Les résultats fournis par OPTSURF s'avèrent satisfaisants. Nous pouvons donc à présent considérer dans la suite de ce chapitre un critère d'optimisation de formes basé sur la réponse harmonique : la minimisation de vitesse normale.

5.5 Optimisation de formes en régime harmonique

Face aux besoins de réduction des vibrations et du bruit en automobile, le recours à une optimisation automatique de formes est capitale. Dans ce contexte, l'optimisation ne dépend alors plus du problème statique comme dans le chapitre 4, mais de la réponse dynamique de la structure. Plusieurs critères peuvent être pris en compte : la compliance dynamique [Ma *et al.*, 1993], [Yoon, 2010], [Olhoff et Du, 2014] a, par exemple, pour objectif d'augmenter la raideur pour limiter les vibrations aux fréquences d'excitation. Parmi les critères vibro-acoustiques des cahiers des charges Renault, la minimisation du facteur de rayonnement est souvent demandée mais ce problème d'optimisation nécessiterait une étude acoustique supplémentaire. Un autre critère peut être à considéré en première approche pour l'amélioration des prestations acoustiques, en l'occurrence, la minimisation de la moyenne de la vitesse normale de vibration de la structure. Nous verrons dans une première partie la définition de ce critère, puis le calcul du gradient associé avant de présenter des résultats d'optimisation.

5.5.1 Définition du critère

Dans [Lamancusa, 1993], il est expliqué que la puissance acoustique est définie à partir de la vitesse moyenne de la surface. L'optimisation de la vitesse normale moyenne permet donc d'influer sur le comportement acoustique de la structure sans avoir à réaliser une étude acoustique complète. Comme nous l'avons vu lors des calculs de benchmark, la figure 5.2 montre que la réponse peut avoir de très fortes variations en fonction des fréquences. Pour cette raison, il est plus judicieux de chercher à minimiser la vitesse normale moyenne sur une bande de fréquences. La fonction $j(\varphi)$ que nous chercherons à minimiser est définie de la façon suivante :

$$j(\varphi) = \frac{1}{\omega_{\max} - \omega_{\min}} \left(\int_{\omega_{\min}}^{\omega_{\max}} \frac{1}{S} \int_{\Omega} |v_n(\omega)|^2 \, dS \, d\omega \right) \tag{5.5.1}$$

avec :

- $\omega_{\min}$ et $\omega_{\max}$ les bornes de la plage de pulsations ;
- $|v_n|$ le module de la vitesse normale ;
- S la surface.

Compte tenu du caractère harmonique de la réponse, la vitesse de la structure est donnée par $\vec{v}(\omega) = i\omega\vec{u}$ où $\vec{u}$ est solution du problème variationnel :

$$\forall \vec{v} \in \vec{\mathcal{W}}, \quad a(\vec{u}, \vec{v}) + i\omega c(\vec{u}, \vec{v}) - \omega^2 \tilde{b}(\vec{u}, \vec{v}) = \tilde{f}(\vec{v}). \tag{5.5.2}$$

tel que $c(\vec{u}, \vec{v}) = \alpha b(\vec{u}, \vec{v}) + \beta a(\vec{u}, \vec{v})$. Par ailleurs, la composante normale correspond par définition à la composante u_3 du déplacement. Ainsi :

$$v_n = i\omega u_3. \tag{5.5.3}$$

En pratique, l'intégration sur la bande de fréquence est faite au moyen d'une méthode de quadrature de Gauss-Legendre de la forme suivante :

$$\int_a^b f(t) \, dt = \frac{b-a}{2} \sum_{j=1}^{n} \lambda_j f\left(\frac{b-a}{2} x_j + \frac{a+b}{2} \right) \tag{5.5.4}$$

avec (x_j, λ_j) les n points et poids de la quadrature.

Finalement, la fonction objectif peut s'écrire :

$$j(\varphi) = \frac{1}{2}\sum_{k=1}^{n} \lambda_k j_k(\varphi), \quad \text{avec} \quad j_k(\varphi) = \frac{1}{S}\int_\Omega \omega_k^2 u_3 \cdot \bar{u}_3 \sqrt{a}\, d\xi_1 d\xi_2 \tag{5.5.5}$$

où $\bar{u}_3$ désigne le conjugué de u_3.

La fonctionnelle j de l'équation (5.5.5) est bien définie lorsque $u_3 \in H^2(\Omega)$ puisque $H^2(\Omega)$ s'injecte continûment dans $\mathcal{C}^0(\Omega)$.

Les contraintes du problème d'optimisation seront de même type que celles des problèmes d'optimisation pour le critère de compliance. Il s'agira donc de bornes sur les coordonnées des points de contrôle et sur la variation de l'aire.

Le problème d'optimisation étant défini, il reste maintenant à déterminer le gradient de la fonction objectif afin de pouvoir réaliser l'optimisation.

5.5.2 Calcul du gradient

La fonction objectif j est définie comme la somme des fonctions j_k. Nous verrons ici comment calculer le gradient des fonctions j_k et en particulier le calcul de la dérivée du terme d'amortissement.

5.5.2.1 Différentielles des fonctions j_k

Proposition 5.5.1. Les différentielles des fonctions j_k définies par :

$$j_k(\varphi) = \frac{1}{S}\int_\Omega \omega_k^2 u_3 \cdot \bar{u}_3 \sqrt{a}\, d\xi_1 d\xi_2$$

sont données par

$$\begin{aligned} Dj_k(\varphi)\cdot\psi = \; & \frac{1}{S}\int_\Omega \omega_k^2 u_{\varphi 3}\cdot \bar{u}_{\varphi 3}\partial_\varphi(\sqrt{a})\cdot\psi\, d\xi_1 d\xi_2 - \frac{\partial_\varphi S}{S^2}\cdot\psi\int_\Omega \omega_k^2 u_{\varphi 3}\cdot\bar{u}_{\varphi 3}\sqrt{a}\, d\xi_1 d\xi_2 \\ & - \mathcal{R}e\left(\partial_\varphi a(\vec{u}_\varphi, \vec{v}_\varphi) + i\omega_k \partial_\varphi c(\vec{u}_\varphi, \vec{v}_\varphi) - \omega_k^2 \partial_\varphi \tilde{b}(\vec{u}_\varphi, \vec{v}_\varphi) - \partial_\varphi \tilde{f}(\vec{v}_\varphi)\right)\cdot\psi \end{aligned} \tag{5.5.6}$$

où $\vec{u}_\varphi$ est solution de l'équation (5.5.2) et $\vec{v}_\varphi$ est solution du problème adjoint :

$$\forall w \in \vec{\mathcal{W}}, \quad a(\vec{v}_\varphi, \vec{w}) - i\omega_k c(\vec{v}_\varphi, \vec{w}) - \omega_k^2 \tilde{b}(\vec{v}_\varphi, \vec{w}) = \int_\Omega 2\omega_k^2 u_{\varphi 3} w_3 \sqrt{a}\, d\xi_1 d\xi_2. \tag{5.5.7}$$

Démonstration. Les fonctions j_k sont des fonctions de variables complexes à valeurs réelles. Afin d'appliquer la méthode de l'état adjoint, nous considérerons les variables complexes comme des couples de variables réelles. On aura ainsi :

$$\vec{u} = (\mathcal{R}e(\vec{u}), \mathcal{I}m(\vec{u})), \quad \vec{v} = (\mathcal{R}e(\vec{v}), \mathcal{I}m(\vec{v})).$$

Posons $j_k(\varphi) = J_k(\varphi, \vec{u}_\varphi)$. Pour utiliser la méthode de l'état adjoint, on introduit en général le Lagrangien égal à la somme de la fonction coût et de la formulation variationnelle de l'équation d'état. En écrivant les composantes réelle et imaginaire de l'équation d'état séparément, c'est-à-dire :

$$\forall \vec{w} \in \mathcal{W}, \quad \begin{cases} a(\mathcal{R}e(\vec{u}), \vec{w}) - \omega_k c(\mathcal{I}m(\vec{u}), \vec{w}) - \omega_k^2 \tilde{b}(\mathcal{R}e(\vec{u}), \vec{w}) = & \mathcal{R}e(f(\vec{w})) \\ a(\mathcal{I}m(\vec{u}), \vec{w}) + \omega_k c(\mathcal{R}e(\vec{u}), \vec{w}) - \omega_k^2 \tilde{b}(\mathcal{I}m(\vec{u}), \vec{w}) = & \mathcal{I}m(f(\vec{w})) \end{cases}$$

on obtient le Lagrangien suivant :

$$\begin{aligned}\mathcal{L}(\varphi,\vec{u},\vec{v}) = \quad & J_k(\varphi,\vec{u}) - \Big(a(\mathcal{R}e(\vec{u}),\mathcal{R}e(\vec{v})) - \omega_k c(\mathcal{I}m(\vec{u}),\mathcal{R}e(\vec{v})) \\ & -\omega_k^2\tilde{b}(\mathcal{R}e(\vec{u}),\mathcal{R}e(\vec{v}))\Big) - \Big(a(\mathcal{I}m(\vec{u}),\mathcal{I}m(\vec{v})) + f(\mathcal{R}e(\vec{v})) \\ & +\omega_k c(\mathcal{R}e(\vec{u}),\mathcal{I}m(\vec{v})) - \omega_k^2\tilde{b}(\mathcal{I}m(\vec{u}),\mathcal{I}m(\vec{v}))\Big) + f(\mathcal{I}m(\vec{v})).\end{aligned}$$

Remarquons que le Lagrangien peut également s'écrire

$$\mathcal{L}(\varphi,\vec{u},\vec{v}) = J_k(\varphi,\vec{u}) - \mathcal{R}e\left(a(\vec{u},\vec{v}) + i\omega_k c(\vec{u},\vec{v}) - \omega_k^2\tilde{b}(\vec{u},\vec{v}) + f(\vec{v})\right)$$

Afin de déterminer, le gradient de j_k, intéressons nous dans un premier temps aux dérivées du Lagragien par rapport à $\mathcal{R}e(\vec{v})$ et $\mathcal{I}m(\vec{v})$. On a :

$$\forall\vec{w}\in\mathcal{W},\quad \begin{cases}\dfrac{\partial\mathcal{L}}{\partial\mathcal{R}e(\vec{v})}(\varphi,\vec{u},\vec{v})\cdot\vec{w} & = a(\mathcal{R}e(\vec{u}),\vec{w}) - \omega_k c(\mathcal{I}m(\vec{u}),\vec{w}) - \omega_k^2\tilde{b}(\mathcal{R}e(\vec{u}),\vec{w}) - \mathcal{R}e(f(\vec{w})) \\ \dfrac{\partial\mathcal{L}}{\partial\mathcal{I}m(\vec{v})}(\varphi,\vec{u},\vec{v})\cdot\vec{w} & = a(\mathcal{I}m(\vec{u}),\vec{w}) + \omega_k c(\mathcal{R}e(\vec{u}),\vec{w}) - \omega_k^2\tilde{b}(\mathcal{I}m(\vec{u}),\vec{w}) - \mathcal{I}m(f(\vec{w}))\end{cases}$$

Ainsi $\dfrac{\partial\mathcal{L}}{\partial\mathcal{R}e(\vec{v})}(\varphi,\vec{u},\vec{v})\cdot\vec{w} = \dfrac{\partial\mathcal{L}}{\partial\mathcal{I}m(\vec{v})}(\varphi,\vec{u},\vec{v})\cdot\vec{w} = 0$ si

$$\forall\vec{w}\in\mathcal{W},\quad \begin{cases}a(\mathcal{R}e(\vec{u}),\vec{w}) - \omega_k c(\mathcal{I}m(\vec{u}),\vec{w}) - \omega_k^2\tilde{b}(\mathcal{R}e(\vec{u}),\vec{w}) = & \mathcal{R}e(f(\vec{w})) \\ a(\mathcal{I}m(\vec{u}),\vec{w}) + \omega_k c(\mathcal{R}e(\vec{u}),\vec{w}) - \omega_k^2\tilde{b}(\mathcal{I}m(\vec{u}),\vec{w}) = & \mathcal{I}m(f(\vec{w}))\end{cases}$$

ce qui correspond à la partie réelle et la partie imaginaire de l'équation d'état, soit $\vec{u} = \vec{u}_\varphi$.
Il reste désormais à établir le problème adjoint. Pour cela, il faut calculer les dérivées du Lagrangien par rapport à $\mathcal{R}e(\vec{u})$ et $\mathcal{I}m(\vec{u})$. On a :

$$\forall\vec{w}\in\mathcal{W},\quad \begin{cases}\dfrac{\partial\mathcal{L}}{\partial\mathcal{R}e(\vec{u})}(\varphi,\vec{u},\vec{v})\cdot\vec{w} = & \dfrac{\partial J}{\partial\mathcal{R}e(\vec{u})}(\vec{u},\varphi)\cdot\vec{w} - a(\vec{w},\mathcal{R}e(\vec{v})) \\ & +\omega_k^2\tilde{b}(\vec{w},\mathcal{R}e(\vec{v})) - \omega_k c(\vec{w},\mathcal{I}m(\vec{v})) \\ \\ \dfrac{\partial\mathcal{L}}{\partial\mathcal{I}m(\vec{u})}(\varphi,\vec{u},\vec{v})\cdot\vec{w} = & \dfrac{\partial J}{\partial\mathcal{I}m(\vec{u})}(\vec{u},\varphi)\cdot\vec{w} - a(\vec{w},\mathcal{I}m(\vec{v})) \\ & +\omega_k^2\tilde{b}(\vec{w},\mathcal{I}m(\vec{v})) + \omega_k c(\vec{w},\mathcal{R}e(\vec{v}))\end{cases}$$

On peut écrire que :

$$J_k(\varphi,\vec{u}) = \frac{1}{S}\int_\Omega \omega_k^2 u_3\cdot\bar{u}_3\sqrt{a}\,d\xi_1 d\xi_2 = \frac{1}{S}\int_\Omega \omega_k^2(\mathcal{R}e(u_3)^2 + \mathcal{I}m(u_3)^2)\sqrt{a}\,d\xi_1 d\xi_2$$

Par suite,

$$\begin{cases}\dfrac{\partial J}{\partial\mathcal{R}e(\vec{u})}(\varphi,\vec{u})\cdot\vec{w} = & \dfrac{1}{S}\displaystyle\int_\Omega 2\omega_k^2\,\mathcal{R}e(u_3)w_3\sqrt{a}\,d\xi_1 d\xi_2, \\ \dfrac{\partial J}{\partial\mathcal{I}m(\vec{u})}(\varphi,\vec{u})\cdot\vec{w} = & \dfrac{1}{S}\displaystyle\int_\Omega 2\omega_k^2\,\mathcal{I}m(u_3)w_3\sqrt{a}\,d\xi_1 d\xi_2.\end{cases}$$

Par conséquent $\dfrac{\partial\mathcal{L}}{\partial\mathcal{R}e(\vec{u})}(\varphi,\vec{u},\vec{v})\cdot\vec{w} = \dfrac{\partial\mathcal{L}}{\partial\mathcal{I}m(\vec{u})}(\varphi,\vec{u},\vec{v})\cdot\vec{w} = 0$ si

$$\begin{cases}a(\vec{w},\mathcal{R}e(\vec{v})) - \omega_k^2\tilde{b}(\vec{w},\mathcal{R}e(\vec{v})) + \omega_k c(\vec{w},\mathcal{I}m(\vec{v})) = & \dfrac{1}{S}\displaystyle\int_\Omega 2\omega_k^2\,\mathcal{R}e(u_3)w_3\sqrt{a}\,d\xi_1 d\xi_2 \\ a(\vec{w},\mathcal{I}m(\vec{v})) - \omega_k^2\tilde{b}(\vec{w},\mathcal{I}m(\vec{v})) - \omega_k c(\vec{w},\mathcal{R}e(\vec{v})) = & \dfrac{1}{S}\displaystyle\int_\Omega 2\omega_k^2\,\mathcal{I}m(u_3)w_3\sqrt{a}\,d\xi_1 d\xi_2.\end{cases}$$

ce qui revient à résoudre :

$$\forall w \in \vec{\mathcal{W}}, \quad a(\vec{v}, \vec{w}) - i\omega_k c(\vec{v}, \vec{w}) - \omega_k^2 \tilde{b}(\vec{v}, \vec{w}) = \int_\Omega 2\omega_k^2 u_3 w_3 \sqrt{a}\, d\xi_1 d\xi_2$$

Notons $\vec{v}_\varphi$ la solution de l'équation adjointe. On a finalement :

$$\begin{aligned} Dj_k(\varphi) \cdot \psi &= \frac{\partial \mathcal{L}}{\partial \varphi}(\varphi, \vec{u}_\varphi, \vec{v}_\varphi) \cdot \psi \\ &= \frac{\partial J}{\partial \varphi}(\varphi, \vec{u}_\varphi) \cdot \psi - \mathcal{R}e \left(\partial_\varphi a(\vec{u}_\varphi, \vec{v}_\varphi) + i\omega_k \partial_\varphi c(\vec{u}_\varphi, \vec{v}_\varphi) - \omega_k^2 \partial_\varphi \tilde{b}(\vec{u}_\varphi, \vec{v}_\varphi) - \partial_\varphi \tilde{f}(\vec{v}_\varphi) \right) \cdot \psi \end{aligned}$$

Avec :

$$\frac{\partial J}{\partial \varphi}(\varphi, \vec{u}_\varphi) \cdot \psi = \frac{1}{S} \int_\Omega \omega_k^2 u_{\varphi 3} \cdot \bar{u}_{\varphi 3} \partial_\varphi(\sqrt{a}) \cdot \psi \, d\xi_1 d\xi_2 - \frac{\partial_\varphi S}{S^2} \cdot \psi \int_\Omega \omega_k^2 u_{\varphi 3} \cdot \bar{u}_{\varphi 3} \sqrt{a}\, d\xi_1 d\xi_2.$$

où $\partial_\varphi S$ est la dérivée de l'aire de l'équation (4.3.12).

□

Remarque 5.5.1. Une autre approche pour déterminer l'état adjoint consiste d'abord à écrire le problème discrétisé sous forme matricielle puis à différentier, comme proposé dans [Nikolova *et al.*, 2004].
En posant $u = [u_R \; u_I]^T$ et $v = [v_R \; v_I]^T$, on a :

$$\nabla_\varphi j_k(\varphi) = \frac{\partial J_k}{\partial \varphi}(\varphi, \vec{u}_\varphi) - \begin{bmatrix} v_{\varphi R} \\ v_{\varphi I} \end{bmatrix}^T \left(\begin{bmatrix} \partial_\varphi K - \omega_k^2 \partial_\varphi M & -\omega_k \partial_\varphi C \\ \omega_k \partial_\varphi C & \partial_\varphi K - \omega_k^2 \partial_\varphi M \end{bmatrix} \begin{bmatrix} u_{\varphi R} \\ u_{\varphi I} \end{bmatrix} - \begin{bmatrix} \partial_\varphi f_R \\ \partial_\varphi f_I \end{bmatrix} \right)$$

où $\begin{bmatrix} u_{\varphi R} \\ u_{\varphi I} \end{bmatrix}$ est solution de l'équation d'état :

$$\begin{bmatrix} K - \omega_k^2 M & -\omega_k C \\ \omega_k C & K - \omega_k^2 M \end{bmatrix} \begin{bmatrix} u_R \\ u_I \end{bmatrix} = \begin{bmatrix} f_R \\ f_I \end{bmatrix},$$

et $\begin{bmatrix} v_{\varphi R} \\ v_{\varphi I} \end{bmatrix}$ est solution de l'équation adjointe :

$$\begin{bmatrix} K - \omega_k^2 M & \omega_k C \\ -\omega_k C & K - \omega_k^2 M \end{bmatrix} \begin{bmatrix} v_R \\ v_I \end{bmatrix} = \begin{bmatrix} \dfrac{\partial J_k}{\partial u_R} & \dfrac{\partial J_k}{\partial u_I} \end{bmatrix}^T.$$

Les dérivées des termes de masse et raideur se font par la dérivation des termes élémentaires (base covariante, base contravariante, forme mixte etc.) comme pour la compliance. L'expression détaillée de ces différentes dérivées sont disponibles dans [Bernadou *et al.*, 1991]. La dérivée de l'amortissement en revanche demande un traitement particulier.

5.5.2.2 Dérivée de l'amortissement

Proposition 5.5.2. Soit ω la pulsation associée au premier mode propre $\vec{\phi}$ de la structure présent dans la bande de fréquence d'étude. La dérivée du terme d'amortissement vaut alors :

$$\partial_\varphi c(\vec{u}, \vec{v}) \cdot \psi = \alpha \partial_\varphi \tilde{b}(\vec{u}, \vec{v}) \cdot \psi + (\partial_\varphi \alpha) \tilde{b}(\vec{u}, \vec{v}) \cdot \psi + \beta \partial_\varphi a(\vec{u}, \vec{v}) \cdot \psi + (\partial_\varphi \beta) a(\vec{u}, \vec{v}) \cdot \psi. \quad (5.5.8)$$

où

$$(\partial_\varphi \alpha) \cdot \psi = \frac{\zeta}{2\omega} \partial_\varphi [a(\vec{\phi}, \vec{\phi}) - \omega^2 \tilde{b}(\vec{\phi}, \vec{\phi})] \cdot \psi, \quad (\partial_\varphi \beta) \cdot \psi = -\frac{\zeta}{2\omega^3} \partial_\varphi [a(\vec{\phi}, \vec{\phi}) - \omega^2 \tilde{b}(\vec{\phi}, \vec{\phi})] \cdot \psi. \quad (5.5.9)$$

Démonstration. D'après l'expression du terme d'amortissement (équation (5.2.14)), la dérivée s'exprime :

$$\partial_\varphi c(\vec{u}, \vec{v}) \cdot \psi = \partial_\varphi(\alpha \tilde{b}(\vec{u}, \vec{v})) \cdot \psi + \partial_\varphi(\beta a(\vec{u}, \vec{v})) \cdot \psi.$$

Rappelons que les coefficients α et β dépendent de façon explicite d'une pulsation propre de la structure (équation 5.4.1). De fait, ils dépendent également de la forme, d'où l'équation (5.5.8).

Déterminons dans un premier temps la dérivée par rapport à la forme de la valeur propre $\lambda = \omega^2$ associé aux modes $\vec{\phi} \in \vec{\mathcal{W}}$. On a :

$$\forall \vec{v} \in \vec{\mathcal{W}}, \quad a(\vec{\phi}, \vec{v}) = \lambda \tilde{b}(\vec{\phi}, \vec{v}) \quad \text{tel que} \quad \tilde{b}(\vec{\phi}, \vec{\phi}) = 1. \tag{5.5.10}$$

En différenciant par rapport à φ, on obtient :

$$\partial_\varphi a(\vec{\phi}, \vec{v}) \cdot \psi + \partial_\phi a(\vec{\phi}, \vec{v})[D\phi(\varphi) \cdot \psi] = \partial_\varphi \lambda \tilde{b}(\vec{\phi}, \vec{v}) \cdot \psi + \lambda \partial_\varphi \tilde{b}(\vec{\phi}, \vec{v}) \cdot \psi + \lambda \partial_\phi \tilde{b}(\vec{\phi}, \vec{v})[D\phi(\varphi) \cdot \psi]$$

soit

$$\Big(\partial_\varphi a(\vec{\phi}, \vec{v}) - \lambda \partial_\varphi \tilde{b}(\vec{\phi}, \vec{v})\Big) \cdot \psi = \partial_\varphi \lambda \tilde{b}(\vec{\phi}, \vec{v}) \cdot \psi - \partial_\phi \Big(a(\vec{\phi}, \vec{v}) - \lambda \tilde{b}(\vec{\phi}, \vec{v})\Big)[D\phi(\varphi) \cdot \psi] \tag{5.5.11}$$

L'équation (5.5.11) est valable pour tout $\vec{v} \in \vec{\mathcal{W}}$, en particulier pour $\vec{v} = \vec{\phi}$. Ainsi d'après les égalités de l'équation (5.5.10), on déduit :

$$\partial_\varphi \lambda \cdot \psi = \partial_\varphi a(\vec{\phi}, \vec{\phi}) \cdot \psi - \lambda \partial_\varphi \tilde{b}(\vec{\phi}, \vec{\phi}) \cdot \psi \tag{5.5.12}$$

Par suite,

$$(\partial_\varphi \omega) \cdot \psi = \frac{1}{2\omega} \Big(\partial_\varphi a(\vec{\phi}, \vec{\phi}) - \omega^2 \partial_\varphi \tilde{b}(\vec{\phi}, \vec{\phi})\Big) \cdot \psi \tag{5.5.13}$$

ce qui donne les résultats de l'équation (5.5.9). □

Remarque 5.5.2. Cette dérivée de l'amortissement est uniquement valable si la pulsation propre choisie est simple. En effet, dans le cas de pulsations multiples, ces dernières ne sont plus dérivables par rapport à la forme [Rousselet et Chenais, 1990].

5.5.3 Validation du gradient

L'expression du gradient étant explicitée, il reste à le valider par une comparaison avec des différences finies. La solution de l'équation d'état est obtenue par la méthode des accélérations modales et celle de l'équation adjointe par une méthode de superposition modale simple. Le pas des différences finies vaut, comme pour la compliance, 10^{-6}.

Les tests ont été effectués sur une configuration similaire à celle du premier cas test de réponse forcée : on considère une plaque de $10\ m \times 10\ m$ et de $5\ cm$ d'épaisseur. La géométrie est cette fois modélisée par un seul patch NURBS de degré 3. Comme indiqué en figure 5.3, la géométrie comprend 16 points de contrôle. La force appliquée est de $1000\ N.m^{-2}$. Les calculs ont été faits avec 460 éléments de discrétisation et 20 fréquences.

La figure 5.4 compare les résultats de la dérivée par l'état adjoint selon la coordonnée z pour le point de contrôle P_1 et des calculs par différences finies. Il apparaît que l'écart entre le gradient calculé par OptSurf et le calcul par différences finies est important. Ce constat est confirmé par les résultats du tableau 5.2.

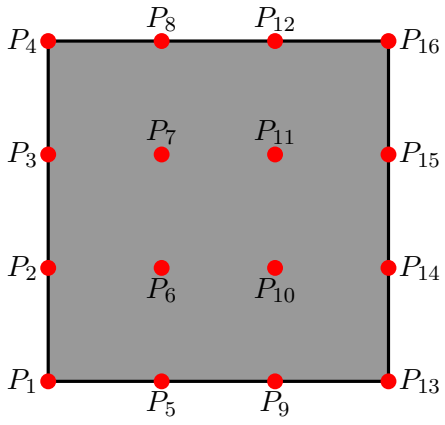


FIGURE 5.3: Géométrie de la plaque avec en rouge les points de contrôle

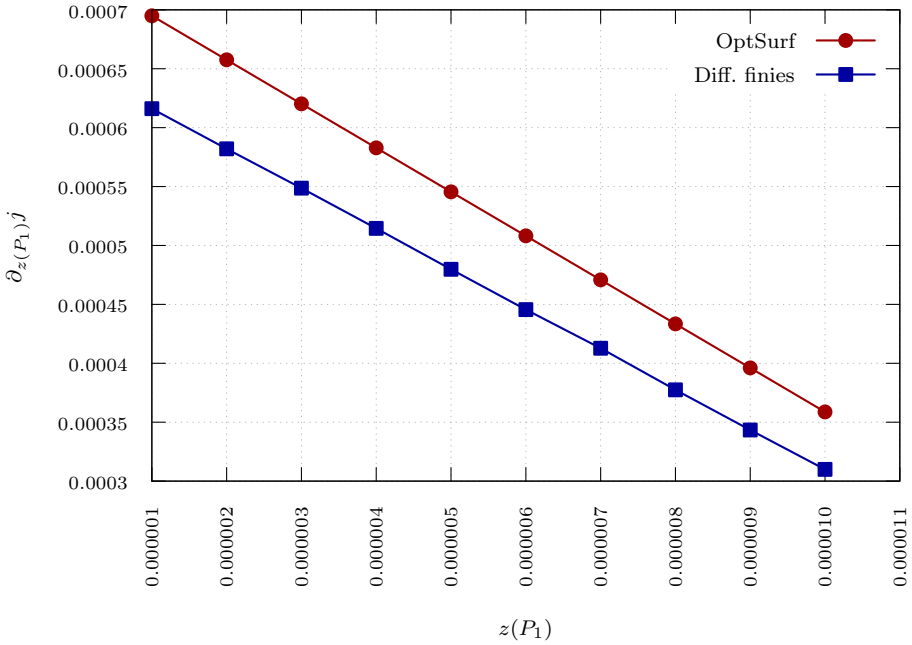


FIGURE 5.4: Comparaison entre le gradient de la vitesse normale par méthode adjointe et par différences finies

Point	Gradient OPTSURF	Différences finies	Erreur relative
P_1	0.0006949544061	0.000616107692641	12.8%
P_2	0.000300078268666	0.000259555182902	15.6%
P_3	0.000300599512441	0.000259243265743	16%
P_4	0.000695283062214	0.000616512174645	12.8%
P_5	0.000301179031025	0.000260005739161	16%
P_6	-0.00129639038287	-0.00119895005035	8.12%
P_7	-0.0012961603298	-0.00119929563502	8.10%
P_8	0.000300540797516	0.000259140514602	16%

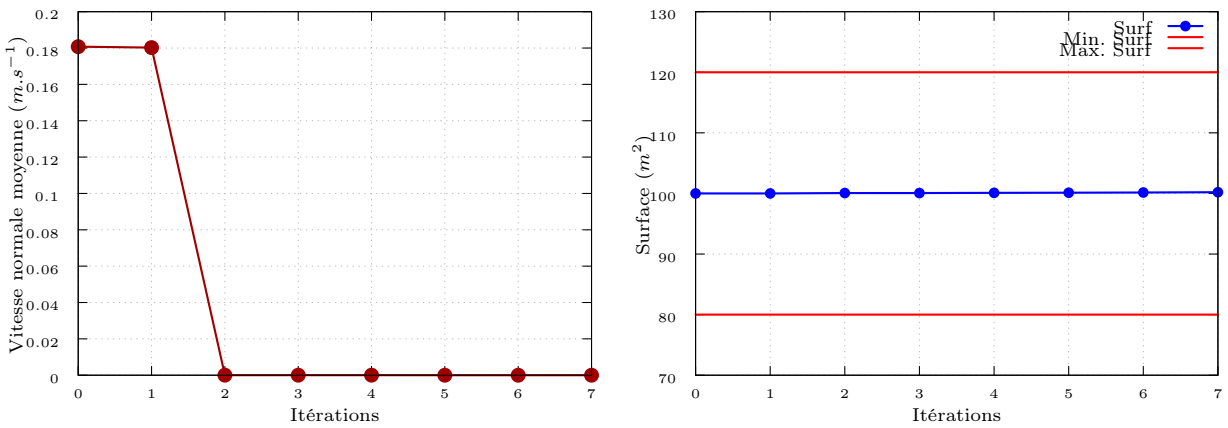
TABLEAU 5.2: Comparaison du gradient par méthode adjointe selon la coordonnée z et par différences finies pour plusieurs points de contrôle de la plaque.

L'erreur par rapport aux différences finies est en moyenne de l'ordre de 10%. Cet écart s'explique par la méthode de calcul de la différentielle de la fonction objectif. En effet, le problème de réponse forcée harmonique est résolu par la méthode des accélérations modales qui est une approximation de la solution exacte. De ce fait, la formulation de l'équation adjointe n'est pas tout à fait juste puisqu'elle est définie pour la solution exacte. On peut donc dire que la méthode de calcul pour la dérivée n'est pas en accord avec le problème réel. Une approche plus juste serait de considérer non plus la solution exacte lors de la formulation du problème adjoint mais la solution approchée.

Malgré les écarts avec les calculs par différences finies, le gradient semble donner une bonne direction de descente. Nous avons ainsi testé le gradient calculé par OptSurf et pouvons donc entreprendre l'étude d'un problème d'optimisation.

5.5.4 Résultats d'optimisation

Le problème d'optimisation traité est la minimisation de la vitesse normale moyenne de la plaque décrte précédemment sur la bande fréquence de 0 à 4.16 Hz. Lors de l'optimisation, les bases modales ont été tronquées aux modes inférieurs à cinq fois la fréquence supérieure de la bande d'étude, soit 20.8 Hz. Les variables d'optimisation sont les coordonnées z de tous les points de contrôle. Les variables sont bornés à ± 1 m de variation. Une variation de $\pm 20\%$ de l'aire de la plaque est tolérée. Les simulations ont été réalisées avec 460 éléments de discrétisation et une discrétisation de 20 fréquences.



(a) Fonction objectif en fonction des itérations

(b) Aire en fonction des itérations

Figure 5.5: Évolution de la vitesse normale moyenne et de l'aire de la plaque en fonction des itérations.

Les figures 5.5a et 5.5b montrent l'évolution de la vitesses normale moyenne et de la surface au cours de l'optimisation. La fonction objectif est passée de 0.18 $m.s^{-1}$ à environ 4.74×10^{-6} $m.s^{-1}$ en 7 itérations. La surface finale vaut 100.155 m^2 alors qu'elle était de 100 m^2 au départ. Comme le montre la figure 5.6b, l'algorithme d'optimisation a apporté une légère courbure à la forme ayant eu pour effet d'augmenter la première fréquence de vibration de la plaque. Ainsi, la première fréquence propre qui valait 2.376 Hz vaut pour la nouvelle forme 16.06 Hz. La diminution de la vitesse normale est donc due à l'absence de fréquences de résonance dans la bande d'étude.

(a) Forme d'origine

(b) Forme optimisée

Figure 5.6: Comparaison entre la forme originale de la plaque et la formé optimisée

Pour ce cas d'optimisation, le gradient a permis une amélioration de la fonction objectif. Davantage

de tests devront être réalisés afin de déterminer si, dans tous les cas, ce calcul de gradient permet de dégager une direction de descente pour l'optimisation.

5.6 Méthode de résolution numérique en régime transitoire

5.6.1 Motivations

Nous avons détaillé jusqu'à présent l'implémentation du calcul en réponse forcée harmonique. Celle-ci intervient dans la définition de nombreux critères d'optimisation, notamment ceux de nature acoustique. La réponse à une excitation harmonique suppose que l'on s'intéresse à un système pour lequel le régime permanent est établi. Dans cette configuration, la dépendance vis à vis du temps est négligée. Il existe cependant plusieurs phénomènes physiques pour lesquels l'étude du régime transitoire est essentielle. Par exemple, la définition d'un critère tel que la tenue à la fatigue nécessite des informations temporelles. Dans cette optique, une première étape consiste en la résolution du problème de réponse forcée en régime transitoire est nécessaire.

5.6.2 Résolution numérique

D'un point de vue discret, le problème de réponse forcée s'écrit sous la forme

$$M\ddot{u}(t) + C\dot{u}(t) + Ku(t) = F(t), \quad u(0) = u_0, \quad \dot{u}(0) = \dot{u}_0. \tag{5.6.1}$$

avec :

- $u(t)$ le déplacement, $\dot{u}(t)$ la vitesse et $\ddot{u}(t)$ l'accélaration.
- M la matrice de masse de taille $n \times n$;
- C la matrice d'amortissement de taille $n \times n$;
- K la matrice de rigidité de taille $n \times n$;
- F les forces appliquées à la structure de taille n ;
- t le temps ;

Pour résoudre ce problème, on utilise généralement la méthode de superposition modale vue précédemment et rappelée ici.

Méthode numérique 5.6.1. Soit $\Phi = \{\phi_1, ..., \phi_n\}$ la base modale tels que $\phi_j M \phi_j = 1$ et $\phi_j K \phi_j = \omega_j^2$. En posant $u(t) = \Phi q$, on obtient :

$$M\Phi\ddot{q}(t) + C\Phi\dot{q}(t) + K\Phi q(t) = F(t) \tag{5.6.2}$$

En multipliant ensuite par Φ^T, l'équation (5.6.2) devient :

$$\Phi^T M\Phi\ddot{q}(t) + \Phi^T C\Phi\dot{q}(t) + \Phi^T K\Phi q(t) = \Phi^T F(t) \tag{5.6.3}$$

Comme pour le problème harmonique, la matrice d'amortissement est supposée vérifier l'hypothèse de Basile. La méthode de superposition permet ainsi d'obtenir un système de n équations différentielles découplées de la forme :

$$\ddot{q}_j(t) + c_j\dot{q}(t) + \omega_j^2 q(t) = f_j(t) \tag{5.6.4}$$

avec c_j les termes de la matrice d'amortissement diagonalisée et $f_j(t) = \phi_j \cdot F(t)$. En pratique, l'amortissement est défini par rapport aux modes en posant $c_j = 2\zeta\omega_j$ où ζ est le facteur d'amortissement. Pour augmenter la vitesse de calcul, on considère une base réduite de taille $m << n$. Il reste maintenant

à résoudre les m équations différentielles non homogènes d'ordre 2. Pour plus de lisibilité, on omettra les indices j et la dépendance en temps. On cherche donc à résoudre :

$$\ddot{q} + 2\zeta\omega\dot{q} + \omega^2 q = f. \tag{5.6.5}$$

Dans un premier temps, on cherche la solution q_1 de l'équation homogène suivante :

$$\ddot{q}_1 + 2\zeta\omega\dot{q}_1 + \omega^2 q_1 = 0. \tag{5.6.6}$$

L'équation caractéristique s'écrit :

$$s^2 + 2\zeta\omega s + \omega^2 s = 0. \tag{5.6.7}$$

Le discriminant vaut donc $\Delta = 4\zeta^2\omega^2 - 4\omega^2$. On distigue trois cas :

- Si $\zeta > 1$ (amortissement sur-critique) alors $\Delta > 0$ et l'équation (5.6.7) admet deux racines réelles s_1 et s_2 :
$$s_1 = -\zeta\omega - \omega\sqrt{\zeta^2 - 1}, \quad s_2 = -\zeta\omega + \omega\sqrt{\zeta^2 - 1}.$$
En posant $\delta = \omega\sqrt{\zeta^2 - 1}$ et en tenant compte des conditions aux limites q_0, et $\dot{q}_0$, la solution de (5.6.6) est donnée par :
$$q_1(t) = \mathrm{e}^{-\zeta\omega t}\left(q_0 \cosh \delta t + \frac{\dot{q}_0 + \zeta\omega q_0}{\delta} \sinh \delta t\right) \tag{5.6.8}$$
- Si $\zeta = 1$ (amortissement critique) alors $\Delta = 0$ et l'équation (5.6.7) admet une racine double réelle s :
$$s = \zeta\omega = \omega$$
En tenant compte des conditions aux limites q_0, et $\dot{q}_0$, la solution de (5.6.6) est donnée par :
$$q_1(t) = \mathrm{e}^{-\omega t}(q_0 + (\dot{q}_0 + \omega q_0)t) \tag{5.6.9}$$
- Si $\zeta < 1$ (amortissement sous-critique) alors $\Delta < 0$ et l'équation (5.6.7) admet deux racines complexes s_1 et s_2 :
$$s_1 = -\zeta\omega - i\omega\sqrt{1 - \zeta^2}, \quad s_2 = -\zeta\omega + i\omega\sqrt{1 - \zeta^2}.$$
En posant $\delta = \omega\sqrt{1 - \zeta^2}$ et en tenant compte des conditions aux limites q_0, et $\dot{q}_0$, la solution de (5.6.6) est donnée par :
$$q_1(t) = \mathrm{e}^{-\zeta\omega t}\left(q_0 \cos \delta t + \frac{\dot{q}_0 + \zeta\omega q_0}{\delta} \sin \delta t\right) \tag{5.6.10}$$

Afin de déterminer la solution de l'équation (5.6.5), il faut ajouter à la solution du problème homogène $q_1(t)$ une solution particulière $q_2(t)$ de (5.6.5). Étant donné que nous traiterons uniquement des cas où l'amortissement est sous-critique, nous détaillerons uniquement celui-ci.

Pour une excitation quelconque f, $q_2(t)$ est donnée par l'intégrale de Duhamel [Géradin et Rixen, 1996] :

$$q_2(t) = \frac{1}{\delta}\,\mathrm{e}^{-\zeta\omega t}\int_0^t f(s)\,\mathrm{e}^{\zeta\omega s}\sin\delta(t-s)\,ds, \tag{5.6.11}$$

Au final :

$$\boxed{q(t) = \mathrm{e}^{-\zeta\omega t}\left(q_0 \cos \delta t + \frac{\dot{q}_0 + \zeta\omega q_0}{\delta} \sin \delta t\right) + \frac{1}{\delta}\int_0^t \mathrm{e}^{-\zeta\omega(t-s)}\,f(s)\sin\delta(t-s)\,ds.} \tag{5.6.12}$$

Remarque 5.6.1. L'intégrale de Duhamel correspond en fait à un produit de convolution. Pour le calcul d'un point de vue discret, le recours à une FFT (*Fast Fourier Transform*) peut s'avérer utile. En effet, rappelons que la transformée de Fourier transforme les produits de convolution en simples produits.

Cette méthode de résolution a été implémentée dans OptSurf. Pour les calculs, la base modale sera composée de l'ensemble des modes dont les fréquences propres associée sont inférieures à la moitié de la fréquence d'échantillonnage. Les résultats de benchmarks issus de la littérature sont présentés dans la partie suivante.

5.6.3 Résultats de benchmarks

5.6.3.1 Plaque soumise à une pression

Le premier benchmark présenté est le cas test NAFEMS 13T issu de [Maguire *et al.*, 1989]. Il reprend la même configuration que celle de la réponse forcée harmonique. La plaque de la figure 5.7 est simplement posée sur ses quatre côtés et soumise à une pression uniformément répartie d'intensité F_0. La plaque a été modélisée avec un patch NURBS de degré 3. Le facteur d'amortissement est de 2%.

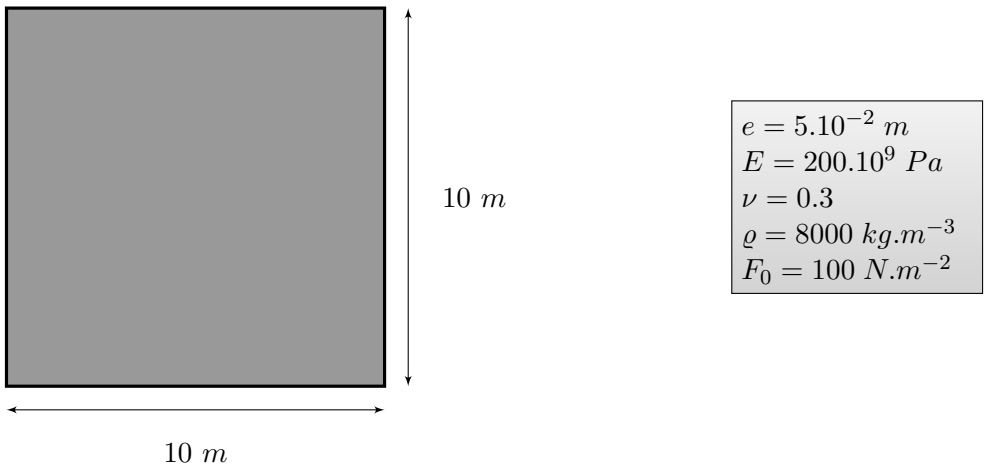


Figure 5.7: Géométrie de la plaque du benchmark NAFEMS 13T

La valeur d'étude est le déplacement maximal selon la direction normale à la plaque, noté u_z, au centre de la plaque. La figure 5.8 présente l'évolution de u_z en fonction du temps et le tableau offre une comparaison entre les valeurs de références et celles calculées par OptSurf avec 1358 éléments de discrétisation et une fréquence d'échantillonage de 100 Hz. La base modale tronquée comporte 28 fréquences au total.

	OptSurf	Théorique	Erreur relative
u_z max (dynamique)	3.43 mm	3.523 mm	2.64%
u_z max (statique)	1.76704 mm	1.817 mm	2.75%

Tableau 5.3: Comparaison entre les valeurs calculées et théoriques pour le cas NAFEMS 13T.

Les résultats fournis par OptSurf présentent un léger écart avec la solution de référence (voir tableau 5.3. L'erreur peut en partie s'expliquer par le fait que ce benchmark concerne normalement un modèle de plaque alors qu'un modèle de coque a été utilisé ici.

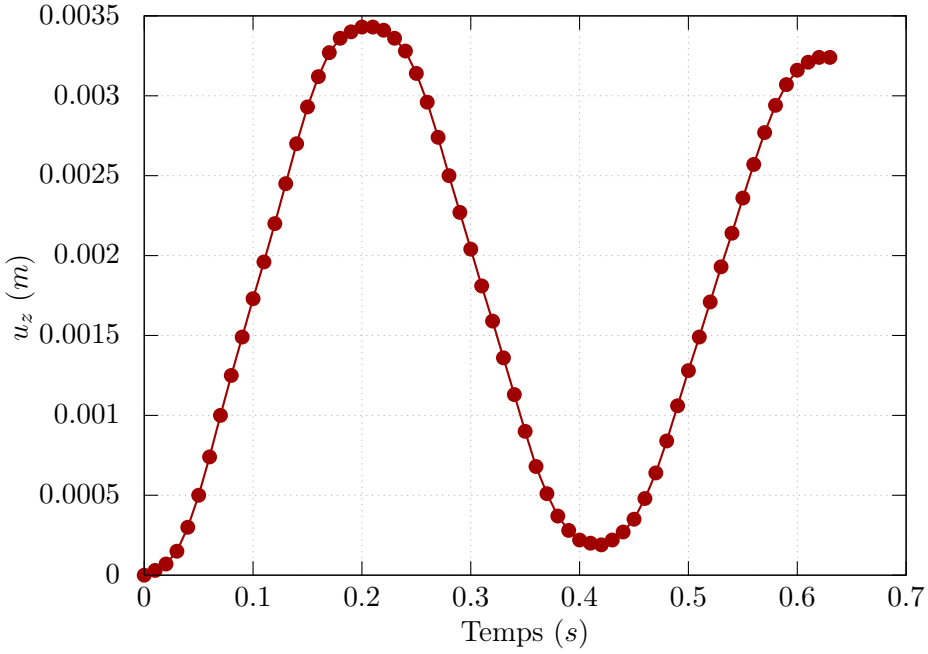


FIGURE 5.8: Évolution de u_z au centre de la plaque en fonction du temps

5.6.3.2 Calotte sphérique

Le second benchmark tiré de [Mallikarjuna *et al.*, 1992] et basé sur les travaux de [Owen et Hinton, 1980] concerne la réponse d'une calotte sphérique soumise à une pression d'intensité F_0. La figure 5.9 donne la configuration du problème . La calotte est encastrée sur toute sa frontière et il n'y a pas d'amortissement. La géométrie du problème a été modélisée à l'aide 4 patchs NURBS de degré 2.

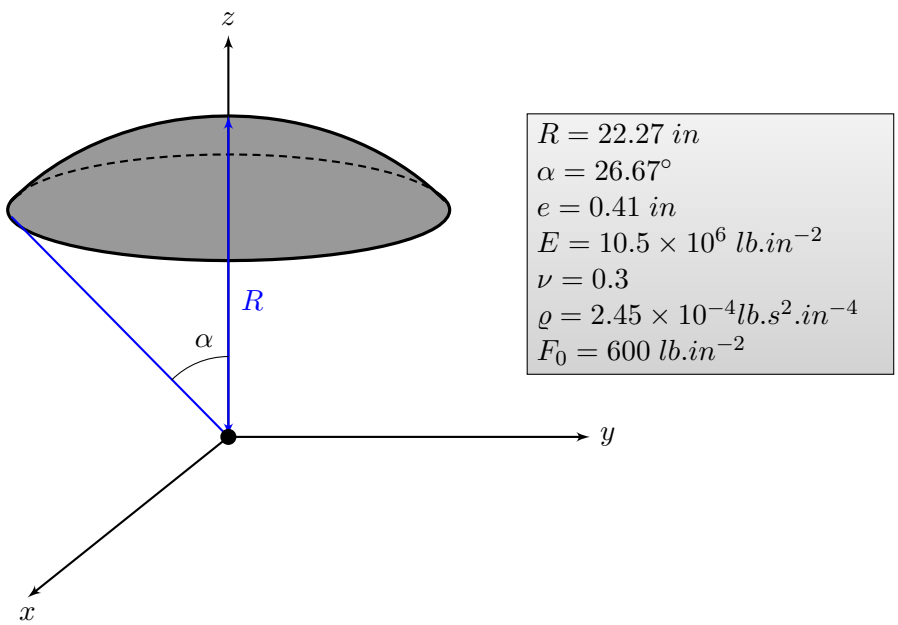


FIGURE 5.9: Configuration pour le problème de calotte sphérique

La réponse est étudiée sur une période de 1 ms. Le calcul a été effectué sur 1840 éléments de discrétisation avec une base modale tronquée comprenant 541 modes. La figure 5.10 permet de comparer

les résultats obtenus avec OptSurf et ceux de référence par le déplacement selon z du sommet de la calotte.

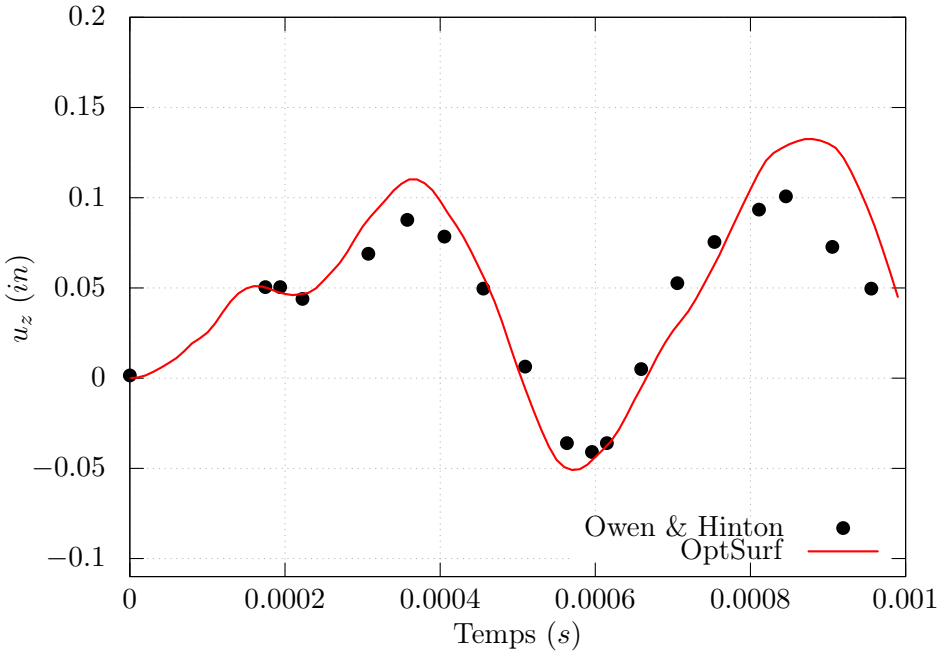


Figure 5.10: Évolution de u_z du sommet de la calotte en fonction du temps

Les résultats montrent qu'il subsiste un certain écart entre les valeurs théoriques et les valeurs calculées. Cependant, le comportement de la structure donné par la simulation suit globalement le comportement théorique.
Avant de pouvoir entreprendre une étude d'optimisation dans ce cas, la principale difficulté consistera à calculer les gradients d'une fonctionnelle dépendant du temps et restant à définir. L'écriture d'un état adjoint demandera certainement un travail différent de celui effectué précedemment pour les autres critères. À noter que la formulation d'un état adjoint pour un critère de fatigue avec une application à un problème de poutre est proposée dans [Fourcade, 2015].

5.7 Synthèse du chapitre

Dans ce chapitre, le problème de réponse forcée en régime harmonique ou transitoire pour le modèle de coques choisi a été présenté. La détermination de la réponse harmonique de la coque a permis d'étudier des critères d'optimisation de grande importance dans les questions de dimensionnement. Le critère qui a été présenté ici, la vitesse normale de la structure, appartient aux critères de type vibro-acoustiques. Il a permis, dans une première approche, d'améliorer les performances acoustiques sans avoir à réaliser une étude acoustique complète.

En ce qui concerne l'implémentation des gradients des critères à caractère dynamique, les résultats sont satisfaisants mais il sera nécessaire dans la suite de mettre en place une méthode de calcul plus adaptée, où la solution de l'équation d'état est définie comme une combinaison linéaire des modes de la base tronquée.

Conclusions et perspectives

La problématique principale de cette thèse était de confirmer et de développer une stratégie d'optimisation de formes pour les problèmes de coques minces basée sur l'analyse isogéométrique. Cette méthode repose sur la définition de la forme à partir des entités géométriques issues des modèles CAO. Un démonstrateur, le code OptSurf, a été implémenté en lien direct avec une librairie d'outils CAO. Des simulations et des optimisations de formes ont pu être réalisées pour des géométries complexes formées de plusieurs patchs CAO. Les premiers essais ont été satisfaisants mais ont également pu montrer les limites actuelles de la méthode. De part sa définition, la simulation repose principalement sur la qualité du modèle CAO. Cette méthode, telle qu'elle existe aujourd'hui, requiert des modèles exclusivement composés de patchs NURBS. Or les outils de conception utilisés chez Renault n'offrent pas cette fonctionnalité. Une évolution possible de la démarche présentée pourra être de chercher à définir les fonctions de forme des modèles de coques à l'aide d'objets issus des outils de conception utilisés chez Renault.

La stratégie d'optimisation proposée consiste à utiliser comme variables d'optimisation les points de contrôle des surfaces NURBS et à calculer le gradient par rapport à ces variables suivant une méthode adjointe. Des résultats satisfaisants pour un critère de compliance ont été obtenus pour un cas industriel, ce qui atteste du potentiel de la méthode. Des critères dynamiques issus de la vibro-acoustique ont également été discutés. Ils ont ensuite servi de base à une optimisation, pour l'instant limitée à des formes simples. Avant l'implémentation de critères comme le facteur de rayonnement, la méthode proposée pour calculer les gradients devra être davantage investiguée. D'un point de vue plus général, la gestion des patchs non conformes sera également utile. L'expression d'une contrainte en fonction des points de contrôle pour assurer la connexion entre les patchs au cours de l'optimisation en est probablement la clé.

Les critères vibro-acoustiques appartiennent aux critères de dimensionnement pour une caisse automobile. L'implémentation de ces derniers est donc précieuse pour l'industrie automobile. Les autres critères de dimensionnement, la tenue à la fatigue et les déformations en grands déplacements, sont tout aussi décisifs. La définition des problèmes d'optimisation pour ces différents critères s'ajoutent donc naturellement aux perspectives des travaux présentés.

ANNEXE

A Classes et routines principales du code OptSurf

Sommaire

A.1 La classe Shape 109
A.2 Calcul et stockage des matrices 112
A.3 Les classes pour l'optimisation 117
A.4 Exemple de fichier de commande pour OptSurf 121

Le code OptSurf est fondé sur un ensemble de librairies open source telles que OCCT, MUMPS, ARPACK etc. Les routines développées combinent différents langages : le C, le C++ et le Fortran 90. L'implémentation des fonctions exige donc une certaine cohérence pour assurer le transfert des informations entre les modules. À cette fin, plusieurs classes C++ ont été développées. Elles permettent de regrouper un certain nombre de données qui peuvent ensuite être accessibles grâce à des fonctions membres adaptées. Dans cette rubrique, nous verrons quelques uns des principaux objets qui permettent le fonctionnement du code.

A.1 La classe Shape

La première étape du fonctionnement de OptSurf consiste en la lecture de la forme et l'identification des surfaces qui la composent. La définition de ces surfaces est essentielle en analyse isogéométrique. C'est à partir de ces définitions que les diverses entités de géométrie différentielle, utiles pour le modèle de coque, peuvent être déterminées. Ces informations doivent donc être accessibles en permanence. Pour cette raison, tous les éléments liés à la géométrie (surfaces, jonctions entre les patchs et maillages) sont stockés dans un objet de type `Shape`. Le fichier `<Shape.hpp>` qui définie l'objet `Shape` est écrit de la façon suivante :

```
//*****************************************************************************
// Definition de la classe Shape :
// Attributs : o edgesMap  = Liste des aretes presentes dans la forme
//             o facesMap  = Liste des faces presentes dans la forme
//             o nbSurf    = Nombre de surfaces (patchs) composant la forme
//             o surfList  = Tableau contenant les surfaces formant la forme
```

```
  //              o junctions = Tableau contenant les informations sur les jonctions
  //              o meshList  = Tableau contenant les maillages associes a chaque patch
  //*****************************************************************************


class Shape {

 public :

  Shape(const char* stepFile,const char* directoryName);
  ~Shape ();

  int getNbSurf();
  int getNbJunctions();
  void getJunctions(int* connect);
  void getDimNodes (int *tabDimNodes);
  void getDimElements (int *tabDimElements);
  void getNodesList(int numPatch,double*Nodes);
  void getElementsList(int numPatch, int*Elements);
  void getLabel(int numPatch,int numNode,char *labelsList) ;

  int getNbRefDomainBorder ();
  char getRefDomainBorder(int numPatch, int labelEdge);

  Handle_Geom_BSplineSurface & operator[] (int i) ;
  Triangle & meshElem (int i, int j);
  Point & meshNode(int i, int j) ;

  int locatePointInMesh(int numPatch,const double u, const double v);


  // Fonctions pour les reperages des noeuds et des triangles communs sur les jonctions

  void getJunctionsLength (double **length) ;                                    //
      Fonctions pour connaitre la longueur de chaque jonction dans les domaines de
      references
  int countCommonNodes (int numJunction);                                        //
      Comptage des noeuds commun sur une jonctions
  void locateCommonNodes (int numJunction,int nbCommonNodes,int **commonNodes);  //
      Reperage des noeuds communs.

  void locateCommonElements(int numJunction,int nbCommonNodes,int **commonNodes,int **
      commonElements); // Reperage des elemnts communs.

  void getTangentVector(int numPatch,int lab, double x,double y,double *tgt);   //
      Calcul le vecteur tangent
  void getRefDomainTangent(int numPatch,int lab, int & du, int & dv);            //
      Calcul l'orientation de la tangente dans le domaine de references

  void getNbPoles(int *nbPoles);                                                 //
      Fonction pour connaitre de point de controle dans chaque patch
  void findPolesEqty() ;                                                         //
      Determine les egalites entre les points de controles
  void getPolesEqty(int *eqty,int nbPoles) ;

  void update () ;                                                               //
      Mise a jour des faces et des aretes
  void exportStep(const char* stepFile) ;                                        //
      Pour exporter la forme dans un fichier step

private :
  TopTools_IndexedMapOfShape edgesMap;
  TopTools_IndexedMapOfShape facesMap;
  int nbSurf;
```

```
  Handle_Geom_BSplineSurface *surfList;
  int nbJunctions;
  int ** junctions;
  Mesh **meshList ;
  int *polesEqty ;


  Shape (const Shape & aShape) ;
  Shape & operator =(const Shape &) ;

};

// Definition d'une fonction de tri qui se base sur un autre tableau

void sortRegardingArray (int *tab,double *order,int size,bool ascending);

//*****************************************************************************
// Definition des fonctions exterieures a la classe pour les appels depuis le  fortran
//*****************************************************************************

extern "C" void cpp_getnbsurf_(Shape *aShape,int *nbSurf) ;
extern "C" void cpp_getnbjunctions_(Shape *aShape, int *nbJunctions);
extern "C" void cpp_getjunctions_(Shape *aShape,int *connect);
extern "C" void cpp_getdimnodes_(Shape *aShape, int *tabDimNodes);
extern "C" void cpp_getdimelements_(Shape *aShape, int *tabDimElements);
extern "C" void cpp_getnodeslist_(Shape *aShape,int *numPatch,double *Nodes) ;
extern "C" void cpp_getelementslist_(Shape *aShape,int *numPatch, int *Elements);
extern "C" void cpp_getnbrefdomainborder_(Shape *aShape,int *nbRefDomainBorder);
extern "C" void cpp_getrefdomainborder_(Shape *aShape,int *numPatch,int *labelElge,char
     *shapeBorders);
extern "C" void cpp_getlabel_(Shape *aShape,int *numPatch,int *numNode,char *labelsList
    );
extern "C" void cpp_countcommonnodes_(Shape *aShape,int *numJunction, int *
    nbCommonNodes);
extern "C" void cpp_locatecommonnodes_(Shape *aShape,int *numJunction,int *
    nbCommonNodes, int *commonNodes);
extern "C" void cpp_locatecommonelements_(Shape *aShape,int *numJunction,int *
    nbCommonNodes, int *commonNodes,int *commonElements);
extern "C" void cpp_getjunctionslength_(Shape *aShape,double *length);
extern "C" void cpp_gettangentvector_(Shape *aShape,int *numPatch,int *lab,double *x,
    double*y,double *tgt);
extern "C" void cpp_getrefdomaintangent_(Shape *aShape,int *numPatch,int *lab, int *du,
     int *dv);
extern "C" void cpp_getcurvilinearcoordinates_(Shape *aShape,double *Pnt,int *Patch,int
     *triangle,double *u,double *v);
extern "C" void cpp_getnbpoles_(Shape *aShape,int *nbPoles) ;
extern "C" void cpp_getpoleseqty_(Shape *aShape,int *eqty,int *nbPoles) ;
```

L'objet `Shape` permet d'exploiter la géométrie à l'aide des objets issue de la librairie OCCT notamment avec le tableau `surfList` qui contient des `Geom_BSplineSurface`. Il est donc possible de déterminer la base covariante en faisant appel à la fonction `cpp_base_cov_`.

```
extern "C" void cpp_base_cov_(Shape *aShape,int *numPatch, double *u, double *v, double
     *a_cov)
{
  gp_Pnt  P;
  gp_Vec  d1u,  d1v ;
  double eps = 1e-15 ;

  (*aShape)[*numPatch]->D1(*u,*v,P,d1u,d1v) ;

  // Vecteur a1
  a_cov[0] = d1u.X();
  a_cov[1] = d1u.Y();
  a_cov[2] = d1u.Z();
```

```
  // Vecteur a2
  a_cov[3] = d1v.X();
  a_cov[4] = d1v.Y();
  a_cov[5] = d1v.Z();

  double norme_a3 = d1u.CrossMagnitude(d1v) ;
  gp_Vec a3 =d1u.Crossed(d1v);

  if(fabs(norme_a3) < eps)
    {

      std::cout <<endl;
      std::cout<<"ssssssssssssssssssssssssssssssssssssssssssssssssssss"<<std::endl;
      std::cout<<"Error␣:␣A␣covariant␣basis␣vector␣is␣nil␣"<<std::endl;
      std::cout<<"ssssssssssssssssssssssssssssssssssssssssssssssssssss"<<std::endl;
      std::cout<<endl;
      exit(0) ;
    }

  else
    {
       // Vecteur a3
      a_cov[6] = a3.X()/norme_a3;
      a_cov[7] = a3.Y()/norme_a3;
      a_cov[8] = a3.Z()/norme_a3;

    }
}
```

Une fois la base covariante calculée, le reste des entités est déterminé par le biais de fonctions en Fortran. On donne ici l'exemple du calcul de la base contravariante :

```
  subroutine base_cont(aShape,numPatch,u,v,a_cont)

  implicit none

  integer :: aShape, numPatch
  double precision :: u,v
  double precision, dimension (3,3) :: a_cont

  double precision ::a11,a12,a21,a22
  double precision, dimension (3,3) :: a_cov


  call base_cov(aShape,numPatch,u,v,a_cov)
  call metric_cont(aShape,numPatch,u,v,a11,a12,a21,a22)

  a_cont(:,1) = a11*a_cov(:,1)+a12*a_cov(:,2)
  a_cont(:,2) = a21*a_cov(:,1)+a22*a_cov(:,2)
  a_cont(:,3) = a_cov(:,3)


end subroutine base_cont
```

A.2 Calcul et stockage des matrices

À partir des fonctions permettant de calculer les entités, les tenseurs de membrane et flexion sont déterminés. Les matrices élémentaires de rigidité et de masse sont ensuite assemblées par la subroutine `koiter_rigiElem`.

```
subroutine koiter_rigiElem(aShape,numPatch,T,Ke,Me,param)
```

```
  implicit none
  include 'parameters_struct.h'

  integer :: aShape, numPatch
  double precision,dimension(3,2) ::T
  double precision,dimension(63,63) :: Ke,Me
  type (parameters_struct) :: param

  double precision :: x,y,sqrtA,Aire
  double precision :: g_11,g_12,g_21,g_22
  double precision,dimension(4,4) :: E_elem
  double precision,dimension (12,12) :: Elas
  double precision,dimension(4,12) :: def_bending,def_memb
  double precision,dimension(5,5) :: Mass
  double precision,dimension(63,12) :: VV
  double precision,dimension(21) :: polargyris,dx_polargyris,dy_polargyris,
      dxx_polargyris,dxy_polargyris,dyy_polargyris
  integer, parameter :: rule = 8
  integer :: order_num,i,j
  integer, dimension(63) :: ordre
  double precision, allocatable, dimension ( : ) :: weights
  double precision, allocatable, dimension ( :, : ) :: lambda12

!     Integration numerique : Dunavant [1985a]
      call dunavant_order_num ( rule, order_num )

      allocate ( weights(1:order_num) )
      allocate ( lambda12(1:2,1:order_num) )

      call dunavant_rule ( rule, order_num, lambda12, weights )

!     Initialisation des matrice Ke et Me
      Ke = 0D0
      Me = 0D0

!     Calcul de l'aire du triangle
      Aire = 0.5D0*((T(3,1)-T(2,1))*(T(1,2)-T(2,2)) - (T(1,1)-T(2,1))*(T(3,2)-T(2,2)))


!     Procedure d'Integration
      do i = 1,order_num
            x = lambda12(1,i)*T(1,1)+lambda12(2,i)*T(2,1)+(1-lambda12(1,i)-lambda12(2,i
                ))*T(3,1)
            y = lambda12(1,i)*T(1,2)+lambda12(2,i)*T(2,2)+(1-lambda12(1,i)-lambda12(2,i
                ))*T(3,2)

!           Calcul des interpolations d'Argyris au point (lambda1,lambda2,lambda3)

            call interpolargyris(lambda12(1,i),lambda12(2,i),1D0-lambda12(1,i)-lambda12
                (2,i),T,&
                  &polargyris,dx_polargyris,dy_polargyris,dxx_polargyris,dxy_polargyris
                      ,dyy_polargyris)

!           Calcul des elements geometriques locaux au point (x,y)
             call metric_cov(aShape,numPatch,x,y,g_11,g_12,g_21,g_22)

!           Calcul du tenseur d'elasticite
            call koiter_elastic_tensor(aShape,numPatch,x,y,param,E_elem)
            call koiter_membrane(aShape,numPatch,x,y,def_memb)
            call koiter_bending(aShape,numPatch,x,y,def_bending)

!           Definition du tenseur elastique
            Elas = param%thick**3*matmul(matmul(transpose(def_bending),E_elem),
                def_bending)/12D0 &
```

```
                & +param%thick*matmul(matmul(transpose(def_memb),E_elem),def_memb)

            Definition du tenseur de masse
            call koiter_mass(aShape,numPatch,x,y,param,Mass)

!           Calcul du sqrtA
            sqrtA = sqrt(g_11*g_22-g_12*g_21)

            VV = 0D0

            ! premiere composante du dep. de la surface moy.
            VV(1:21,1) = polargyris
            VV(1:21,2) = dx_polargyris
            VV(1:21,3) = dy_polargyris


            ! Seconde composante du dep. de la surface moy.
            VV(22:42,4) = polargyris
            VV(22:42,5) = dx_polargyris
            VV(22:42,6) = dy_polargyris


            ! Troisieme composante du dep. de la surface moy.
            VV(43:63,7) = polargyris
            VV(43:63,8) = dx_polargyris
            VV(43:63,9) = dy_polargyris
            VV(43:63,10) = dxx_polargyris
            VV(43:63,11) = dxy_polargyris
            VV(43:63,12) = dyy_polargyris

            Ke = Ke + sqrtA*weights(i)*matmul(matmul(VV,Elas),transpose(VV))
            Me = Me + sqrtA*weights(i)*matmul(matmul(VV(:,(/1,4,7,8,9/)),Mass),
                transpose(VV(:,(/1,4,7,8,9/))))

        end do


      ordre = (/1,4,5,10,11,12,22,25,26,31,32,33,43,46,47,52,53,54,&
            & 2,6,7,13,14,15,23,27,28,34,35,36,44,48,49,55,56,57,&
            & 3,8,9,16,17,18,24,29,30,37,38,39,45,50,51,58,59,60,&
            & 19,40,61, &
            & 20,41,62, &
            & 21,42,63/)

      Ke = Ke(ordre,ordre)*Aire
      Me = Me(ordre,ordre)*Aire*param%rho*param%thick**3/12D0

      deallocate ( weights )
      deallocate ( lambda12 )


return
end subroutine
```

Les matrices élémentaires sont ensuite assemblées pour former la matrice globale. Toutes les entrées sont passées à la librairie CSPARSE via la fonction `add_cs_entry_`.

```
extern "C" void add_cs_entry_(cs *M, int *i, int *j,double *x)
{
  cs_entry(M,*i,*j,*x);
}
```

La fonction `cs_compress` de CSPARSE est utilisée pour supprimer les doublons. Enfin, la matrice globale est stockée au format sparse COO dans un objet `COOMatrix`. Le format COO consiste à ne stocker que les entrées non nulles dans trois tableaux : un pour les indices de la ligne, un pour les indices de la

colonne et le dernier pour la valeur. Ces trois tableaux font donc naturellement partie des attributs de la classe `COOMatrix`.

```
//*****************************************************************************
// Definition de la classe COOMatrix :(Ecriture sparse des matrices)
// Attributs : o dim        = dimensions de la matrice
//             o dimInd     = dimensions des tablaux row,col,val
//             o row        = indices des lignes des coeff non nuls de la matrice dans
    la numerotation globale
//             o col        = indices des colonnes des coeff non nusl de la matrice
    dans la numerotation globale
//             o val        = valeurs des coeff non nuls de la matrice
//             o reverseCOO = correspondance des indices COO et des indices reels de la
     matrice
//*****************************************************************************

class COOMatrix {

public :

  // Constructeur et destructeurs :

  COOMatrix() ;                                          // Constructeur par defaut
  COOMatrix(int dimMat,int dimIndices)  ;                // Constructeur pour creer une
      matrice COO  des tableaux de taille dimInd
  COOMatrix(Shape *aShape);                              // Constructeur a partir d'un
      objet Shape
  COOMatrix (COOMatrix const & matrix) ;                 // Constructeur par copie
  COOMatrix & operator= (COOMatrix const & matrix);      // Operateur d'affectation
  ~COOMatrix ();

  // Acesseurs et mutateurs :

  int getDim () ;                                        // Renvoie la dimension de la
      matrice
  int getDimInd();                                       // Renvoie la taille des vecteurs
       row,col,val,reverseCOO
  void getRow(int *row);                                 // Renvoie row dans row (tableau
      de taille dimInd)
  void getCol(int *col);                                 // Revoie col dans col (tableau
      de taille dimInd)
  void getVal(double *values);                           // Renvoie val dans values (
      tableau de taille dimInd)

  void setDim (int dim) ;                                // Modifie la dimension de la
      matrice
  void setDimInd(int dimInd) ;                           // Modifie la dimension des
      tableaux coo
  void setRow(int *row);                                 // Copie dans row le tableau row
      (tableau de taille dimInd)
  void setCol(int *col);                                 // Copie dans col le tableau col
      (tableau de taille dimInd)
  void setVal(double *values);                           // Copie dans val le tableau
      values (tableau de taille dimInd)


  void computeDimensions(Shape *aShape, int & dimMat,int & dimIndices) ; // Calcul de
      la dimension et du nombre d'indices theorique

private :

  // attributs:

  int dim ;
```

```
  int dimInd;
  int *row;
  int *col;
  double *val;

};

// Definitions des fonctions externes pouvant etre appelees dans le fortran

extern "C" void cpp_getdim_(COOMatrix*aCOOMatrix, int *dim);
extern "C" void cpp_getdimind_(COOMatrix*aCOOMatrix,int *dimInd);
extern "C" void cpp_getrow_(COOMatrix*aCOOMatrix,int *row);
extern "C" void cpp_getcol_(COOMatrix*aCOOMatrix,int *col);
extern "C" void cpp_getval_(COOMatrix*aCOOMatrix,double *values);
extern "C" void cpp_setdim_(COOMatrix*aCOOMatrix, int *dim);
extern "C" void cpp_setdimind_(COOMatrix*aCOOMatrix, int *dimInd);
extern "C" void cpp_setrow_(COOMatrix*aCOOMatrix,int *row);
extern "C" void cpp_setcol_(COOMatrix*aCOOMatrix,int *col);
extern "C" void cpp_setval_(COOMatrix*aCOOMatrix,double *values);
extern "C" void cpp_getrow_from_tab_(COOMatrix**aCOOMatrix,int *num,int *row);
extern "C" void cpp_getcol_from_tab_(COOMatrix**aCOOMatrix,int *num,int *col);
extern "C" void cpp_getval_from_tab_(COOMatrix**aCOOMatrix,int *num,double *val);
extern "C" void cpp_setrow_in_tab_(COOMatrix**aCOOMatrix,int *num,int *row);
extern "C" void cpp_setcol_in_tab_(COOMatrix**aCOOMatrix,int *num,int *col);
extern "C" void cpp_setval_in_tab_(COOMatrix**aCOOMatrix,int *num,double *val);
extern "C" void cpp_setdim_in_tab_(COOMatrix**aCOOMatrix,int *num,int *dim);
extern "C" void cpp_setdimind_in_tab_(COOMatrix**aCOOMatrix,int *num,int *dimind);
```

Les conditions aux limites sont définies à travers une objet `CondLim`. Sa principale fonction est de déterminer les indices de la matrice qui sont concernés par l'application de la condition aux limites.

```
//*****************************************************************************
// Definition de la classe CondLim :
// Attributs : o label   = le numero du label auquel est attachee la condition
//             o ddl     = le tableau indiquant quels ddls sont bloques
//             o nbToDel = le nombre de ddl a supprimer
//             o IndToDel= la liste des ddl a supprimer
///*****************************************************************************

class CondLim {

public :

  // Constructeurs et destructeur :

  CondLim() ;
  CondLim(Shape & aShape,int label,const char*type) ;                          //
      Constructeur avec des conditions predefinies encastrement ou simplement pose
  CondLim(Shape & aShape,int label,int *ddlManual) ;                           //
      Constructeur avec des conditions aux limites definies (remplissage du tableau ddl
      )
  ~CondLim ();                                                                 //
      Destructeur

  CondLim(const CondLim & condition) ;                                         //
      Constructeur par copie
  CondLim & operator=(const CondLim & condition);                              //
      Operateur d'affectation

  // Accesseurs

  int getNbToDel () ;
  int getLabel () ;
  void getIndToDel (int *cp_IndToDel) ;
```

```
private :

  int label ;
  int ddl [21];
  int nbToDel ;
  int *indToDel;

};

// lien avec les fonction fortran
extern "C" void count_indices_to_del_(Shape *aShape,int *nbPatch,int *label,int *ddl,
    int *nbToDel) ;
extern "C" void find_indices_to_del_(Shape *aShape,int *nbPatch,int *label,int *ddl,int
     *nbToDel,int *IndToDel);
```

Les routines pour la résolution du système sont codées en Fortran et sont issues des documentations [Mumps, 2016] et [Lehoucq *et al.*, 1997].

A.3 Les classes pour l'optimisation

L'implémentation d'un problème d'optimisation repose sur trois classes. Les deux première,s `OptimShape` et `OptimInfos`, sont des classes permettant de faire le lien entre OptSurf et ipopt. La troisième classe nécessaire au fonctionnement de l'optimisation est une classe spécifique au critère. Par définition, la librairie ipopt requiert la création d'une classe pour le type de problème étudié [Vigerske et Wächter, 2015]. Nous présentons dans les suites `OptimShape` et `OptimInfos` ainsi que `Ipopt_Compliance` la classe pour la définition du critère de compliance.

```
/*****************************************************************************/
// Definition de la classe OptimShape : Cette classe permet la mise a jour des
    informations
// de la forme a chaque iteration de l'algorithme d'optimisation
//
// Attributs : o theShape        = La forme a optimiser
//             o nbPoles         = nombre de point de controle
//             o nbPolesTab      = tableau contenant le nombre de points de controles
    par patch
//             o poles           = tableau de  contenant les points de controles
//             o nbPolesToIgnore= nombre de points de controles qui ne doivent pas etre
     modifies
//             o polesToIgnore   = liste des indices des points qui ne doivent pas etre
    modifies
//             o eqty            = tableau contenant  les points de controles identiques
//             o nbVar           = nombre de points de controles qui seront modifies
//             o varInd          = liste des indices des points qui seront modifies
//
/*****************************************************************************/



class OptimShape {

public :
  OptimShape(const char* stepFile,const char *directoryName,int nbPti,int *pti);
  ~OptimShape() ;

  int getNbPoles();
  void getPoles(gp_Pnt *Pts);
  void getWeights(double *Wgts) ;
  int getNbPolesToIgnore();
  void getPolesToIgnore(int *listOfPoles);
```

```
  void getEqualities(int *polesEqty) ;
  int getNbVar();
  void getVarInd(int *listOfVar);

  void updateOptimVar(bool *xyzw,double *var,int size);
  void updateOptimGrad(bool *xyzw,double *grad,int size, double *dPx,double *dPy,double
      *dPz, double *dPw);
  void updatePoles(bool *xyzw,const double *var,int size) ;

  void createCondLim(CondLim & cond,int label, const char*type);
  void  applyPonctualForce(Force & f,double *p) ;
  void exportStep(const char *filename);

  // fonctions objectif :

  void computeCompliance(MultipleCondLim *condLim,MultipleForces *forces,double thick,
      double young,double nu, double rho,double & compliance,double & error,int nbproc)
      ;
  void computeGradCompliance(MultipleCondLim *condLim,MultipleForces *forces,double
      thick,double young,double nu, double rho,double & compliance, double *
      dPx_compliance,double *dPy_compliance, double *dPz_compliance, double *
      dW_compliance,double & error,int nbproc);
  void computeMeanSquareVelocity(MultipleCondLim *condLim,MultipleForces *forces,double
       thick,double young,double nu, double rho,double damp,double fmin, double fmax,
      int nef,double & velocity, int nbproc);
  void computeGradMeanSquareVelocity(MultipleCondLim *condLim,MultipleForces *forces,
      double thick,double young,double nu, double rho,double damp,double fmin, double
      fmax, int nef,double & velocity,double *dPx_velocity,double *dPy_velocity,double
      *dPz_velocity,double *dW_velocity, int nbproc);
  void computeSurface(double & surface);
  void computeGradSurface(double & surface, double *dPx_surface, double *dPy_surface,
      double *dPz_surface, double *dW_surface);


// private :
  Shape *theShape ;
  int nbPoles     ;
  gp_Pnt *poles   ;
  double *weights ;
  int *nbPolesTab ;
  int nbPolesToIgnore;
  int *polesToIgnore;
  int *eqty;
  int nbVar ;
  int *varInd;


  OptimShape (const OptimShape & OptimaShape) ;
  OptimShape & operator =(const OptimShape & );

};

// Lien avec les fonctions fortran

extern "C" void get_dim_all_cond_raccord_(Shape *aShape, int *nbJct,int *dimAll, int *
    dimIndAll);


extern "C" void koiter_compute_compliance_(Shape *aShape,int *nbPatchs,double *thick,
    double *young,double *nu,double *rho,COOMatrix *K,COOMatrix *M,int *dimKM,int *
    dimIndKM,COOMatrix *KR,int *dimKR,int *dimIndKR,MultipleCondLim *condLim,
    MultipleForces *forces,double *compliance,double *error,int *nbproc);
```

```
extern "C" void koiter_compute_grad_compliance_(Shape *aShape,int *nbPatchs,double *
    thick,double *young,double *nu,double *rho,COOMatrix *K,COOMatrix *M,int *dimKM,int
     *dimIndKM,COOMatrix *KR,int *dimKR,int *dimIndKR, COOMatrix **dPx_K,COOMatrix **
    dPy_K,COOMatrix **dPz_K, COOMatrix **dW_K, COOMatrix **dPx_M,COOMatrix **dPy_M,
    COOMatrix **dPz_M, COOMatrix **dW_M, COOMatrix **dPx_KR,COOMatrix **dPy_KR,
    COOMatrix **dPz_KR, COOMatrix **dW_KR,MultipleCondLim *condLim,MultipleForces *
    forces, int *nbPoles,double *compliance,double *dPx_compliance,double *
    dPy_compliance, double *dPz_compliance, double *dW_compliance,double *error,int *
    nbproc);



extern "C" void koiter_compute_mean_square_normal_velocity_(Shape *aShape,int *nbPatch,
    double *thick,double *young,double *nu,double *rho,COOMatrix *K,COOMatrix *M,int *
    dimKM,int *dimIndKM,COOMatrix *KR,int *dimKR,int *dimIndKR, COOMatrix *QtKQ,
    COOMatrix *QtMQ,MultipleCondLim *condLim,MultipleForces *loads,double *fmin,double
    *fmax,int *nef,double *damp,double *velocity,int *nbproc) ;

extern "C" void koiter_compute_grad_mean_square_normal_velocity_(Shape *aShape,int *
    nbPatchs,double *thick,double *young,double *nu,double *rho,COOMatrix *K,COOMatrix
    *M,int *dimKM,int *dimIndKM,COOMatrix *KR,int *dimKR,COOMatrix ** dPx_K,COOMatrix
    **dPy_K,COOMatrix **dPz_K,COOMatrix **dW_K,COOMatrix **dPx_M,COOMatrix **dPy_M,
    COOMatrix **dPz_M,COOMatrix **dW_M,int* dimIndKR,COOMatrix *QtKQ,COOMatrix *QtMQ,
    MultipleCondLim *condLim,MultipleForces *loads,double *fmin,double *fmax,int *nef,
    double *damp,double *velocity,int *nbPoles,double *dPx_velocity,double *
    dPy_velocity,double *dPz_velocity, double *dW_velocity,int *nbproc) ;

extern "C" void compute_surface_(Shape *aShape,int *nbPatchs,double *surface);

extern "C" void compute_grad_surface_(Shape *aShape,int *nbPatchs, int *nbPoles,double
    *surface,double *dPx_surface, double *dPy_surface, double *dPz_surface, double *
    dW_surface);

/*****************************************************************************/
// Definition de la classe OptimInfo : Cette classe permet de recenser les informations
     liees a l'alogirthme au cours de l'optimisation
//
// Attributs : o iter            = iteration
//             o feval           = valeur de la fonction objectif
//             o derivatives     = tableau de booleen idiquant s'il y a les derivees
    selon x,y,z,w
//             o size            = nombre de variable d'optimisation
//             o x               = tableau contenant les valeurs des variables d'
    optimisation
//             o file_obj        = operateur d'ecriture dans un fichier de resultats (
    fonction objective)
//             o file_surf       = operateur d'ecriture dans un fichier de resultats (
    evolution surface)
//             o resultFile      = nom du fichier du fichier step ou est exporte la
    solution
/*****************************************************************************/

class OptimInfos{

public :

  // Attributs :
  int iter ;
  double feval ;
  bool derivatives [4] ;
  int size ;
  double *x;
  ofstream *file_obj ;
  ofstream *file_surf;
  std::string resultFile ;
```

```
  OptimInfos (bool *dxdydzdw,OptimShape *OptSurf,const char *filename_obj,const char *
      filename_surf, const char *exportFile);                          // Constructeur
      avec le nombre de variables a optimiser
  ~OptimInfos () ;

      // Destructeur


};


using namespace Ipopt;


class Ipopt_Compliance : public TNLP

{
public :
  Ipopt_Compliance(const char* stepFile,const char *directoryName,int nbPti,int *pti,
      bool *dxdydzdw,const char *filename_obj,const char *filename_surf,double Thick,
      double Young, double Nu, double Rho,int nbCondLim,int nbForces,int NbProc,const
      char *exportFile,bool Save);
  ~ Ipopt_Compliance();
  bool get_nlp_info(Index& n, Index& m, Index& nnz_jac_g,Index& nnz_h_lag,
      IndexStyleEnum& index_style);
  bool get_bounds_info(Index n, Number* x_l, Number* x_u,Index m, Number* g_l, Number*
      g_u);
  bool get_starting_point(Index n, bool init_x, Number* x,bool init_z, Number* z_L,
      Number* z_U,Index m, bool init_lambda,Number* lambda);
  bool eval_f(Index n, const Number* x, bool new_x, Number& obj_value);
  bool eval_grad_f(Index n, const Number* x, bool new_x, Number* grad_f);
  bool eval_g(Index n, const Number* x, bool new_x, Index m, Number* g);
  bool eval_jac_g(Index n, const Number* x, bool new_x,Index m, Index nele_jac, Index*
      iRow, Index *jCol,Number* values);
  bool eval_h(Index n, const Number* x, bool new_x, Number obj_factor, Index m, const
      Number* lambda,bool new_lambda, Index nele_hess, Index* iRow,Index* jCol, Number*
       values);
  void finalize_solution(SolverReturn status,Index n, const Number* x, const Number*
      z_L, const Number* z_U, Index m, const Number* g, const Number* lambda,Number
      obj_value, const IpoptData* ip_data,IpoptCalculatedQuantities* ip_cq);

  void setIpoptBounds(double *lx, double *ux, double *ly, double *uy, double *lz,
      double *uz,double delta_surf);
  void addCondLim(int label,const char*type,int num);
  void addForce(Force & f,int num);
  void addPonctualForce(Force & f, double *p,int num);
  void applyCondLim();

private :

  Ipopt_Compliance(const Ipopt_Compliance&);
  Ipopt_Compliance & operator=(const Ipopt_Compliance&);

  // attributs :


  OptimShape *optShape;
  OptimInfos *optInfos;
  double thick;
  double young;
  double nu;
  double rho;
  double damp ;
  MultipleCondLim *condLim;
```

```
  MultipleForces *forces;
  int nbProc ;
  bool saveIter ;
  double originalSurface ;
  double *dPx_compliance;
  double *dPy_compliance;
  double *dPz_compliance;
  double *dW_compliance;
  double *dPx_surface;
  double *dPy_surface;
  double *dPz_surface;
  double *dW_surface;
  double *lx;
  double *ux;
  double *ly;
  double *uy;
  double *lz;
  double *uz;
  double delta_surface;
};
```

A.4 Exemple de fichier de commande pour OptSurf

La plupart des calculs avec OptSurf sont réalisés à partir de la lecture d'un fichier de commande de la forme suivante :

```
##############################################################################
# Fichier de commande pour le cas test du paraboloide hyperbolique
##############################################################################

##############################################################################
# Computation : mechanics or modal analysis
##############################################################################

Computation = Static

##############################################################################
# STEP file to import
##############################################################################

StepFile = ../../CAO/Macros/stepFiles/hyperbolic_paraboloid.step

##############################################################################
# Mesh informations
##############################################################################

MeshFolder = hyperbloic_paraboloid_patchs_meshes
GenerateMeshes = yes
Discretization = 10

##############################################################################
# Mechanical parameter
##############################################################################

Thickness = 0.8
YoungModulus = 2.85e4
PoissonRatio = 0.4
Density = 7300

##############################################################################
# Boundary Conditions
##############################################################################
```

```
BoundaryConditions = 4
1   encastrement
2   encastrement
3   encastrement
4   encastrement


##############################################################################
# Loads
##############################################################################

Loads = 1
pression 0 0.01

##############################################################################
# PostProcessing
##############################################################################

Strain = yes
Filename = deplacement_paraboloid.resu
Constraints = no
Filename = .
LaunchPostProcessing = yes
NumberOfDisplacements = 1
A 0.0 0.0 0.0
```

Bibliographie

[Abballe *et al.*, 2015] Abballe, T., Albertelli, M., Allaire, G., Caron, A. *et al.* (2015). Rodin project, topology optimization 2.0 ? Actes du congrès "Simulation" de la Société des Ingénieurs de l'Automobile (SIA), Montigny le Bretonneux, 18-19 mars 2015. (cité page 9)

[Adam *et al.*, 2015] Adam, C., Bouabdallah, S., Zarroug, M. et Maitournam, H. (2015). Improved numerical integration for locking treatment in isogeometric structural elements. part ii : Plates and shells. *Computer Methods in Applied Mechanics and Engineering*, 284:106 – 137. (cité page 32)

[Adhikari, 2006] Adhikari, S. (2006). Damping modelling using generalized proportional damping. *Journal of Sound and Vibration*, 293(1):156–170. (cité page 92)

[Allaire, 2007] Allaire, G. (2007). *Conception Optimale des Structures*. Springer-Verlag and Heidlberg. (cité pages 5 et 9)

[Allaire *et al.*, 2004] Allaire, G., Jouve, F. et Toader, A.-M. (2004). Structural optimization using sensitivity analysis and a level-set method. *Journal of Computational Physics*, 194:363–393. (cité page 8)

[Auricchio *et al.*, 2010] Auricchio, F., Da Veiga, L. B., Hughes, T., Reali, A. et Sangalli, G. (2010). Isogeometric collocation methods. *Mathematical Models and Methods in Applied Sciences*, 20(11):2075–2107. (cité page 28)

[Banichuk, 2010] Banichuk, N. (2010). *Introduction to optimization of structures*. Springer Verlag. (cité page 6)

[Bazilevs *et al.*, 2006a] Bazilevs, Y., Beirao da Veiga, L., Cottrell, J. A., Hughes, T. J. et Sangalli, G. (2006a). Isogeometric analysis : approximation, stability and error estimates for h-refined meshes. *Mathematical Models and Methods in Applied Sciences*, 16(07):1031–1090. (cité page 28)

[Bazilevs *et al.*, 2006b] Bazilevs, Y., Calo, V., Zhang, Y. et Hughes, T. J. (2006b). Isogeometric fluid–structure interaction analysis with applications to arterial blood flow. *Computational Mechanics*, 38(4-5):310–322. (cité page 28)

[Bazilevs *et al.*, 2010] Bazilevs, Y., Calo, V. M., Cottrell, J. A., Evans, J. A., Hughes, T., Lipton, S., Scott, M. A. et Sederberg, T. W. (2010). Isogeometric analysis using t-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5):229–263. (cité page 62)

[Bendsøe et Sigmund, 2004] Bendsøe, M. P. et Sigmund, O. (2004). *Topology Optimization. Theory, Methods and Applications.* Springer Verlag. (cité page 8)

[Bernadou, 1994] Bernadou, M. (1994). Méthodes des éléments finis pour les problèmes de coques minces. (cité pages 32, 38, 41, 45, 46 et 50)

[Bernadou et Cubier, 1996a] Bernadou, M. et Cubier, A. (1996a). Numerical analysis of junctions between thin shells, part 1 : continuous problems. Rapport technique, INRIA. (cité page 38)

[Bernadou et Cubier, 1996b] Bernadou, M. et Cubier, A. (1996b). Numerical analysis of junctions between thin shells, part 2 : Approximation by finite elements methods. Rapport technique, INRIA. (cité page 38)

[Bernadou *et al.*, 1991] Bernadou, M., Palma, F. et Rousselet, B. (1991). Shape optimization of an elastic thin shell under various criteria. *Structural optimization*, 3(1):7–21. (cité pages 71, 73 et 98)

[Bischoff *et al.*, 2004] Bischoff, M., Bletzinger, K.-U., Wall, W. et Ramm, E. (2004). Models and finite elements for thin-walled structures. *Encyclopedia of computational mechanics.* (cité page 31)

[Bletzinger *et al.*, 2010] Bletzinger, K.-U., Firl, M., Linhard, J. et Wüchner, R. (2010). Optimal shapes of mechanically motivated surfaces. *Computer methods in applied mechanics and engineering*, 199(5):324–333. (cité page 81)

[Blouza *et al.*, 2006] Blouza, A., Hecht, F. et Le Dret, H. (2006). Two finite element approximations of naghdi's shell model in cartesian coordinates. *SIAM journal on numerical analysis*, 44(2):636–654. (cité page 59)

[Braibant et Morelle, 1990] Braibant, V. et Morelle, P. (1990). Shape optimal design and free mesh generation. *Structural optimization*, 2(4):223–231. (cité page 11)

[Brujic *et al.*, 2010] Brujic, D., Ristic, Mihailoand Mattone, M., Maggiore, P. et De Poli, G. P. (2010). CAD based shape optimization for gas turbine component design. *Structural and Multidisciplinary Optimization*, 41(4):647–659. (cité page 10)

[Bruyneel *et al.*, 2014] Bruyneel, M., Craveur, J.-C. et Gourmelen, P. (2014). *Optimisation des structures mécaniques. Méthodes numériques et éléments finis.* Dunond. (cité page 11)

[Bézier, 1977] Bézier, P. (1977). *Essai de définition numérique des courbes et des surfaces expérimentales : contribution à l'étude des propriétés des courbes et des surfaces paramétriques polynomiales à coefficients vectoriels.* Thèse de doctorat, Université Paris VI. (cité page 20)

[Céa, 1986] Céa, J. (1986). Conception optimale ou identification de formes, calcul rapide de la dérivée directionnelle de la fonction coût. *RAIRO-Modélisation mathématique et analyse numérique*, 20(3):371–402. (cité page 72)

[Chambolle, 2003] Chambolle, A. (2003). A density result in two-dimensional linearized elasticity, and applications. *Archive for Rational Mechanics and Analysis*, 167(3):211–233. (cité page 9)

[Chenais, 1975] Chenais, D. (1975). On the existence of a solution in a domain identification problem. *Journal of Mathematical Analysis and Applications*, 52(2):189 – 219. (cité page 9)

[Choi et Kim, 2006] Choi, K. et Kim, N. (2006). *Structural Sensitivity Analysis and Optimization 1 : Linear Systems.* Mechanical Engineering Series. Springer New York. (cité page 90)

[Ciarlet, 2005] Ciarlet, P. G. (2005). An introduction to differential geometry with applications to elasticity. *Journal of Elasticity*, 78(1):1–215. (cité pages 30 et 32)

[Ciarlet, 2007] Ciarlet, P. G. (2007). *Introduction à l'analyse numérique matricielle et à l'optimisation.* Dunod, 5ème édition. (cité page 4)

[Cottrell *et al.*, 2009] Cottrell, J. A., Hughes, T. J. R. et Bazilevs, Y. (2009). *Isogeometric Analysis. Toward integration of CAD and FEA.* Wiley. (cité pages 12, 26 et 28)

[Cottrell *et al.*, 2006] Cottrell, J. A., Reali, A., Bazilevs, Y. et Hughes, T. J. (2006). Isogeometric analysis of structural vibrations. *Computer methods in applied mechanics and engineering*, 195(41):5257–5296. (cité page 28)

[Culioli, 2012] Culioli, J.-C. (2012). *Introduction à l'optimisation.* Ellipses, 2ème édition. (cité page 4)

[Dapogny, 2013] Dapogny, C. (2013). *Shape optimization, level set methods on unstructured meshes and mesh evolution.* Thèse de doctorat, Université Pierre et Marie Curie. (cité page 11)

[de Boor, 1962] de Boor, C. (1962). Bicubic spline interpolation. *Journal of Mathematics and Physics*, 41:212–218. (cité page 22)

[de Casteljau, 1985] de Casteljau, P. (1985). *Mathématiques et CAO 2, Formes à pôles.* Hermès. (cité page 20)

[de Nazelle, 2013] de Nazelle, P. (2013). *Paramétrage de formes surfaciques pour l'optimisation.* Thèse de doctorat, École Centrale de Lyon. (cité pages xiii, 9, 12, 32 et 33)

[Delfour et Zolésio, 2001] Delfour, M. C. et Zolésio, J.-P. (2001). *Shape and Geometries. Analysis, Differential Calculus and Optimization.* SIAM. (cité page 4)

[Dervieux et Palmerio, 1976] Dervieux, A. et Palmerio, B. (1976). *Une Formule De Hadamard Dans Des Problemes d'Optimal Design*, pages 63–7. Springer Berlin Heidelberg. (cité page 7)

[Fourcade, 2015] Fourcade, C. (2015). Conception mécanique des structures - optimisation de structure sur critère de fatigue. Cours de l'Université Paris-Saclay. (cité pages 16 et 106)

[Froment, 2014] Froment, P. (2014). *Optimisation de formes paramétriques en grande dimension.* Thèse de doctorat, École centrale de Lyon. (cité page 14)

[Genest, 2016] Genest, L. (2016). *Optimisation de formes par gradient en dynamique rapide.* Thèse de doctorat, École Centrale de Lyon. (cité page 17)

[Géradin et Rixen, 1996] Géradin, M. et Rixen, D. (1996). *Théorie des vibrations : application à la dynamique des structures.* Collection Physique fondamentale et appliquée. Masson. (cité pages 91, 92 et 103)

[Golub, 1973] Golub, G. H. (1973). Some modified matrix eigenvalue problems. *Siam Review*, 15(2): 318–334. (cité page 54)

[Goupy, 2013] Goupy, J. (2013). *Introduction aux plans d'expériences.* Dunod, 5ème édition. (cité page 12)

[Guezaine et Remacle, 2014] Guezaine, C. et Remacle, J.-F. (2014). *Gmsh Reference Manual - The documentation for Gmsh 2.8. A finite element mesh generator with built-in pre- and post-processing facilities.* (cité page 50)

[Hadamard, 1908] Hadamard, J. (1908). *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées.* (cité page 7)

[Hardee *et al.*, 1999] Hardee, E., Chang, K.-H., Tu, J., Choi, K. K., Grindeanu, I. et Yu, X. (1999). A CAD-based design parameterization for shape optimization of elastic solids. *Advances in Engineering Software*, 30(3):185 – 199. (cité page 10)

[Haslinger et Mäkinen, 2003] Haslinger, J. et Mäkinen, R. (2003). *Introduction to shape optimization. Theory, approximation and computation.* SIAM. (cité page 6)

[Henn, 1998] Henn, H.-W. (1998). Crash tests and head injury criterion. *Teaching Mathematics and its Applications*, 17(4):162–170. (cité page 16)

[Henrot et Pierre, 2005] Henrot, A. et Pierre, M. (2005). *Variation et optimisation de formes.* Springer Verlag. (cité page 4)

[Hughes *et al.*, 2010] Hughes, T. J., Reali, A. et Sangalli, G. (2010). Efficient quadrature for nurbs-based isogeometric analysis. *Computer methods in applied mechanics and engineering*, 199(5):301–313. (cité page 28)

[Hughes *et al.*, 2005] Hughes, T. J. R., Cottrell, J. A. et Bazilevs, Y. (2005). Isogeometric analysis : CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering*, 194:4135–4195. (cité page 20)

[Häußler et Albers, 2005] Häussler, P. et Albers, A. (2005). Shape optimization of structural parts in dynamic mechanical systems based on fatigue calculations. *Structural and Multidisciplinary Optimization*, 29(5):361–373. (cité page 15)

[Kiendl *et al.*, 2009] Kiendl, J., Bletzinger, K.-U., Linhard, J. et Wüchner, R. (2009). Isogeometric shell analysis wit Kirchhoff–Love elements. *Computer Methods in Applied Mechanics and Engineering*, 198(49–52):3902 – 3914. (cité page 28)

[Kiendl, 2010] Kiendl, J. M. (2010). *Isogeometric Analysis and Shape Optimization Design of Shell Structures.* Thèse de doctorat, Technische Universität München. (cité pages 36 et 38)

[Koiter, 1966] Koiter, W. T. (1966). On the nonlinear theory of thin elastic shells. I- introductory sections. II- basic shell equations. III- simplified shell equations(nonlinear theory of thin elastic shells, discussing surface geometry and deformation, equations of equilibrium and boundary conditions and stress functions). *Koninklijke Nederlandse Akademie van Wetenschappen, Proceedings, Series B*, 69(1):1–54. (cité page 31)

[Kurtaran *et al.*, 2002] Kurtaran, H., Eskandarian, A., Marzougui, D. et Bedewi, N. E. (2002). Crashworthiness design optimization using successive response surface approximations. *Computational Mechanics*, 29(4):409–421. (cité page 17)

[Lamancusa, 1993] Lamancusa, J. (1993). Numerical optimization techniques for structural-acoustic design of rectangular panels. *Computers & structures*, 48(4):661–675. (cité page 95)

[Laporte et Le Tallec, 2003] Laporte, E. et Le Tallec, P. (2003). *Numerical Methods in Sensitivity Analysis and Shape Optimization.* Springer Verlag. (cité page 7)

[Lehoucq *et al.*, 1997] Lehoucq, R. B., Sorensen, D. et Yang, C. (1997). *ARPACK Users' Guide : Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.* (cité pages 55 et 117)

[Ma *et al.*, 1994] Ma, Z.-D., Cheng, H.-C. et Kikuchi, N. (1994). Structural design for obtaining desired eigenfrequencies by using topology and shape optimization method. *Computing Systems in Engineering*, 5(1):77–89. (cité page 15)

[Ma *et al.*, 1993] Ma, Z.-D., Kikuchi, N. et Hagiwara, I. (1993). Structural topology and shape optimization for a frequency response problem. *Computational mechanics*, 13(3):157–174. (cité page 95)

[Macneal et Harder, 1985] Macneal, R. H. et Harder, R. L. (1985). A proposed standard set of problems to test finite element accuracy. *Finite elements in Analysis and Design*, 1(1):3–20. (cité page 60)

[Maguire *et al.*, 1989] Maguire, J., Dawswell, D. et Gould, L. (1989). *Selected Benchmarks For Forced Vibration*. NAFEMS. (cité pages 93 et 104)

[Mallikarjuna *et al.*, 1992] Mallikarjuna, Kant, T. et Fafard, M. (1992). Transient response of isotropic, orthotropic and anisotropic composite-sandwich shells with the superparametric element. *Finite Elem. Anal. Des.*, 12(1):63–73. (cité page 105)

[Marburg et Hardtke, 2002] Marburg, S. et Hardtke, H.-J. (2002). An optimization technique in structural-acoustic design of sedan body panels. *In UTAM Symposium on Designing for Quietness*, pages 279–297. Spring Netherlands. (cité page 15)

[Mohammadi et Pironneau, 2009] Mohammadi, B. et Pironneau, O. (2009). *Applied Shape Optimization for Fluids, 2nd Edition*. Oxford University Press. (cité page 6)

[Mumps, 2016] Mumps (2016). *MUltifrontal Massively Parallel Solver (MUMPS 5.0.0) Users' guide*. (cité pages 52 et 117)

[Murat et Simon, 1976] Murat, F. et Simon, J. (1976). Étude de problèmes d'optimal design. *In Optimization Techniques Modeling and Optimization in the Service of Man Part 2 : Proceedings, 7th IFIP Conference Nice, September 8–12, 1975*, pages 54–62. Springer Berlin Heidelberg. (cité page 9)

[Nikolova *et al.*, 2004] Nikolova, N. K., Bandler, J. W. et Bakr, M. H. (2004). Adjoint techniques for sensitivity analysis in high-frequency structure cad. *IEEE transactions on microwave theory and techniques*, 52(1):403–419. (cité page 98)

[Ohayon et Soize, 1997] Ohayon, R. et Soize, C. (1997). *Structural Acoustics and Vibration : Mechanical Models, Variational Formulations and Discretization*. Elsevier Science. (cité page 90)

[Olhoff et Du, 2014] Olhoff, N. et Du, J. (2014). *Topological Design for Minimum Dynamic Compliance of Structures under Forced Vibration*, pages 325–339. Springer Vienna. (cité page 95)

[OpenCascade, 2015] OpenCascade (2015). *OpenCascade Technology Documentation*. (cité page 50)

[Osher et Santosa, 1988] Osher, S. et Santosa, S. (1988). Front propagating with curvature dependant speed : algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 78:12–49. (cité page 8)

[Owen et Hinton, 1980] Owen, D. et Hinton, E. (1980). *Finite elements in plasticity : theory and practice*. Pineridge Press. (cité page 105)

[Pellet, 2011] Pellet, J. (2011). *Dualisation des conditions aux limites*. (cité page 52)

[Petyt, 2010] Petyt, M. (2010). *Introduction to Finite Element Vibration Analysis*. Cambridge University Press. (cité page 91)

[Piegl et Tiller, 1995] Piegl, L. et Tiller, W. (1995). *The NURBS Book*. Springer. (cité page 20)

[Rousselet, 1986] Rousselet, B. (1986). *Principes d'analyse de sensitivité, utilisation pour la conception optimale*. INRIA. (cité page 71)

[Rousselet et Chenais, 1990] Rousselet, B. et Chenais, D. (1990). Continuité et différentiabilité d'éléments propres : Application à l'optimisation de structures. *Applied Mathematics and Optimization*, 22(1):27–59. (cité page 99)

[Schnack et Weikl, 2001] Schnack, E. et Weikl, W. (2001). Shape optimization under fatigue using criteria continuum damage mechanics. *Computer-Aided Design*, 34. (cité page 15)

[Sederberg *et al.*, 2003] Sederberg, T. W., Zheng, J., Bakenov, A. et Nasri, A. (2003). T-splines and t-nurccs. *In ACM transactions on graphics (TOG)*, volume 22, pages 477–484. ACM. (cité page 62)

[Sinha, 2007] Sinha, K. (2007). Reliability-based multiobjective optimization for automotive crashworthiness and occupant safety. *Structural and Multidisciplinary Optimization*, 33(3):255–268. (cité page 16)

[Sokolowski et Zolésio, 1992] Sokolowski, J. et Zolésio, J.-P. (1992). *Introduction to shape optimisation. Shape sensitivity analysis.* Springer Verlag. (cité page 7)

[Timošenko et Woinowsky-Krieger, 1993] Timošenko, S. et Woinowsky-Krieger, S. (1993). *Theory of plates and shells.* Engineering Societies monographs. McGraw-Hill. (cité page 59)

[Vigerske et Wächter, 2015] Vigerske, S. et Wächter, A. (2015). *Introduction to IPOPT : A tutorial for downloading, installing, and using IPOPT.* (cité page 117)

[Wächter et Biegler, 2006] Wächter, A. et Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57. (cité pages 77 et 79)

[Yoon, 2010] Yoon, G. H. (2010). Structural topology optimization for frequency response problem using model reduction schemes. *Computer Methods in Applied Mechanics and Engineering*, 199(25–28):1744 – 1763. (cité page 95)

[Zimmer *et al.*, 2012] Zimmer, H., Gross-Thebing, A., Prabhu, M. et Duddeck, F. (2012). A new approach for vibro-acoustic optimization using discrete and continuous shape variables applied to a car body. *In Proceedings of the FISITA 2012 World Automotive Congress*, pages 425–434. Spring Verlag Heidelberg. (cité page 15)