Master Mathématiques Analyse Numérique Calcul Scientifique

Benchmarking de solveurs d'optimisation pour des problèmes d'optimisation topologique 3D

G. O. AGYEKUM

9 septembre 2019

Benchmarking de solveurs d'optimisation pour des problèmes d'optimisation topologique 3D

G. O. AGYEKUM

9 septembre 2019

Le but du stage est d'évaluer et comparer les performances de plusieurs solveurs d'optimisation, appliqués à divers problèmes d'optimisation topologique de structures mécaniques sous contraintes dans le cadre de l'élasticité linéaire. Une bibliothèque complète, représentative d'une classe de problèmes de compliance minimale et volume minimal d'une structure élastique pour différentes tailles d'éléments finis (E.F) est devéloppée pour cette analyse comparative.

La compliance minimale ou le volume minimal est en fait un problème "mal posé", c'est à dire qu'en général il n'existe pas de solution optimale et les solutions calculées par des méthodes numériques classiques sont dépendantes d'un choix initial de mise à l'échelle [1] et du maillage. Mais la méthode SIMP (Solid Isotropic Material with Penalization) [2], combinée à des méthodes de Filtrage [3] basées sur la théorie de l'homogénéisation permettent de rendre ce problème bien posé.

Plusieurs solveurs classiques y compris MMA, GCMMA, OC [4], et des solveurs des bibliothèques IPOPT, NLOPT sont répertoriés dans la bibliothèque réalisée et leurs performances sont évaluées et comparées à l'aide d'une courbe de performance (Profils de performance et "Data profiles").

Quand c'est possible les solveurs sont comparés quand appliqués à des problèmes aux formulations Nested et/ou SAND (Simultaneous ANalysis and Design). Tous les solveurs sont branchés dans le code TopOpt PETSc [5] et les tests numériques sont effectués sur HPC [6] (High Performance Computing).

Table des matières

1. Quotient de E_1/E_v, où E_1 est le module de Young du matériau solide et E_v du matériau proche du vide
2. Méthode d'interpolation du matériau
3. Méthodes de restriction qui permettent d'assurer l'existence d'une solution optimale
4. Méthodes numériques classiques d'ordre 1 courrament utilisés en optimisation topologique
5. Bibliothèque en C++ dévelopée par N. Aage, E. Andreassen, B. Lazarov
6. Calcul à haute performance

1 INTRODUCTION

1.1 Motivations

Le besoin d'optimiser une fonction objectif est fréquent dans les tâches de conception industrielle et de gestion. Dans la phase de conception d'un nouveau produit, par exemple, l'ingénieur veut non seulement minimiser les coûts, mais aussi probablement maximiser les performances, minimiser le poids ou maximiser la durée de vie du produit. En pratique, la fonction objectif n'est souvent pas donnée sous une forme explicite, mais elle est définie de manière implicite, par exemple comme le résultat d'une simulation. Mais le résultat d'une simulation est souvent trés couteux, d'où le besoin de faire appelle à des algorithmes d'optimisation afin de réduire le coût d'exploration de l'espace de conception ou design (ie, on utilise les algorithmes d'optimisation pour nous guider dans l'espace de conception).

Dans un tel cas, on peut utiliser des méthodes classiques comme la méthode OC, MMA et GCMMA pour résoudre le problème. Cependant, de nombreux algorithmes existants sont disponibles et ce n'est pas simple pour un utilisateur de choisir un algorithme à appliquer à un nouveau problème inconnu.

C'est là que l'analyse des performances des algorithmes rentre en jeu. Si les difficultés observées dans les problèmes pratiques sont couvertes par un ensemble de problèmes tests bien choisis, les algorithmes peuvent être comparés afin de déterminer quels algorithmes fonctionnent mieux que d'autres. C'est pourquoi, on visera à évaluer des combinaisons de méthodes d'optimisation et de formulations dans les problèmes d'optimisation topologique à l'aide d'une courbe de performances pour déterminer si elles sont efficaces, robustes et fiables pour cette classe de problèmes.

Les solveurs plus généraux dans IPOPT [7] et NLOPT [8] sont comparés aux méthodes d'optimisation topologique plus couramment utilisées (classiques) comme OC, MMA et GCMMA.

1.2 État de l'art

1.2.1 Optimisation topologique des structures mécaniques

Pour guider le choix des solutions techniques ou dimensionner des sous-ensembles vers une solution moins mauvaise que les autres, les bureaux d'études structures emploient de plus en plus des méthodes d'optimisation, et en particulier l'optimisation topologique.

Les outils d'optimisation sont utilisés dans de trés nombreux domaines dont la gestion de production, la finance, les transports aériens et ferroviaires entre autres. Ces méthodes sont utilisées en particulier

7. contient des méthodes de points intérieurs
8. contient des méthodes séquentielles quadratiques

dans les bureaux d'études mécaniques en génie civil, construction navale, aéronautiques et automobile entre autres. Certains outils d'optimisation sont spécifiques à un type de problème, par exemple la méthode OC [9] et ont été dévéloppés pour répondre à un besoin précis, d'autres sont plus généraux.

Mais les techniques d'optimisation reposent sur un ensemble de définitions d'objets mathématiques importantes et sont souvent difficiles à comprendre et à utiliser. Les routines d'optimisation nécessitent la formulation analytique du problème et celui-ci doit se tourner vers le calcul numérique.

Grâce à l'avènement de l'ordinateur, l'idée de conception optimale (ou vers une solution meilleure) des structures mécaniques est possible. Sur la base d'une conception initiale définie par un certain nombre de paramètres de valeurs variables, appelés variables de conception, l'optimisation des structures vise à déterminer de manière algorithmique (ou automatique) la conception qui est la meilleure au regard des critères liés à des performances structurales. La conception optimale est trouvée par un procédé itératif altenant analyse structurale et application d'une technique d'optimisation.

Mais la physique de certains problèmes à optimiser est souvent traduite par des équations aux dérivées partielles dont la solution, si elle existe, est très souvent inconnue. Utiliser diverses techniques numériques comme par exemple la méthode des élements finis permettent d'approcher cette solution. Dans ce cadre, on considèrera la méthode des éléments finis (M.E.F) pour l'analyse structurale afin de déterminer une conception optimale. On ne peut pas faire de l'optimisation sans la maitrise de la M.E.F, de la modélisation et du maillage.

Le problème d'optimisation topologique est formulé comme étant la recherche de la distribution optimale des propriétés matérielles dans un domaine de conception initiale prescrit. Ce type d'optimisation s'applique aux structures discrétisées par des éléments de barres, de poutres (M.E.F discrets), de membranes, de plaques ou de coques, et de volumes (M.E.F continues).

Le but de l'optimisation topologique, par exemple dans le cas M.E.F discrets est de déterminer quelles barres ou poutres vont conserver les propriétés mécaniques du matériau les constituant à la solution, les autres éléments structuraux étants, eux affectés de propriétés mécaniques proches du vide. Le maillage est donc figé, seules les propriétés mécaniques de la conception changent et ne subissent aucune évolution au fur et mesure des itérations. Donc la topologie des éléments et le raffinement du maillage ont une influence sur la conception (ou solution) optimale.

On considèrera la discrétisation par M.E.F continues, dont chaque élément fini e, on définit une variable de conception, notée $\rho_e \in \{0,1\}$ [10] et appelée pseudo-densité. Le problème est de déterminer l'absence de matière ou non dans chaque élément e à la solution optimale. Mais dans le cas pratique, on travaille en variables continues qui varient dans l'intervalle]0,1]. on évite de prendre des valeurs nulles pour éviter la singularité numérique liée à la suppression complète d'un membre structural. Donc la matrice de raideur obtenue par M.E.F continues est toujours symétrique, définie positive.

Sa mise en œuvre est décrite dans la figure 1

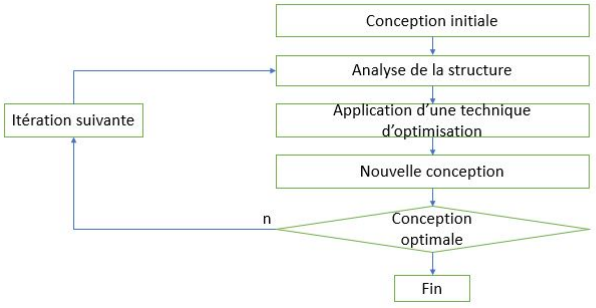


FIGURE 1 – Mise en œuvre de l'optimisation des structures

9. Optimal criteria : méthodes empirique
10. on parle de Black and white

(voir. [23])

1.2.2 Problèmes d'optimisation topologique

Comme nous l'avons dit précedement, l'optimisation topologique des structures mécaniques est la recherche de la distribution optimale de la matière dans un domaine de conception prescrit, dont la fonction objectif et les contraintes correspondent à des réponses structurales représentant le poids de la structure, et sa raideur. Les variables de conception, appelées pseudo-densités sont, des densités de matières variant entre 0 et 1, les contraintes sur les variables de conception définissent la géométrie de la pièce à concevoir. Celles-ci permettent de paramétrer avec plus ou moins de libérté la structure étudiée. La validité de la solution est fixée par les performances structurales et les restrictions à la conception (ou les contraintes) retenues dans le problème d'optimisation. En d'autres termes, selon la manière avec laquelle on va paramétrer le problème, on pourra définir différents problèmes d'optimisation.

Comme premier exemple de fonction objectif et contraintes dans un problème d'optimisation, supposons qu'on veuille résoudre un problème sous une forme standard pour la programmation non linéaire suivant :

$$min : f_0(x) \tag{1}$$

$$\text{sujet à } : f_i(x) \leq 0, i = 1, .m \tag{2}$$

$$x \in X. \tag{3}$$

Où $x = (x_1, ., x_n)^T \in \mathbb{R}^n$, sont les variables de conception. Ici $X = \{x \in \mathbb{R}^n | x_j^{min} \leq x_j \leq x_j^{max}, j = 1, .., n\}$, où x_j^{min} et x_j^{max} sont des réels données, qui satisfont $x_j^{min} \leq x_j^{max}$, $\forall j$, $f_0, f_1, ., f_m$ sont des fonctions données, continument différentiable, à valeurs réelles, définies sur X. Ici f_0 est la fonction objectif, f_i, i=1,..,m définissent les contraintes du problème d'optimisation, et x_j^{min} et x_j^{max} définissent les contraintes de bornes.

Par la suite on considèrera deux classes de problèmes d'optimisation topologique des structures mécaniques : compliance minimale et volume minimal

1.2.3 Compliance minimale

La fonction objectif classique en optimisation topologique est la fonction énergie associée à la force appliquée à la structure mécanique. On maximise la raideur de la structure dont on connait l'encombrement, les conditions limites et les chargements. C'est-à dire qu'on minimise l'énergie de déformation associée à la structure. Cette énergie est appelée compliance. En d'autres termes, on minimise le déplacement sous la charge. La contrainte du problème de minimisation de la compliance est liée au volume de la matière (ou à la masse) à conserver dans le domaine de conception initial. Ainsi le modèle mathématiques du problème continue d'optimisation topologique peut être formulé comme suit :

Soit Ω_0, un domaine de conception et x, le vecteur des variables de conception, ie généralement la pseudo-densité de chaque élément. Dans le cadre de l'élasticité linéaire, on cherche le minimum de l'énergie de déformation :

$$\min_{\tilde{E} \in E_{AD}} l(u(\tilde{E})), \tag{4}$$

$$\text{avec } : a(u,v) = l(v), v \in V, \tag{5}$$

où u(E) est solution du problème variationnelle (PV) : $u \in V$ tel que a(u,v) = l(v), $\forall v \in V$,

avec :

$$a(u,v) = \int_{\Omega_0} E_{ijkl}(x)\epsilon_{ij}(u)\epsilon_{kl}(v)dx,$$

$$\epsilon_{ij}(u) = \frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i})$$

$$l(v) = \int_{\Omega_0} \Omega f v dx + \int_{\Omega} \Gamma_N t u dx,$$

où f : = le chargement, t := est la traction sur la surface Γ_N

Autrement dit, on cherche une conception optimale Ω dans un domaine de conception prescrit Ω_0 dont la contrainte est liée au volume (ie, $|\Omega|$) de la matière à conserver dans le domaine de conception intiale et qui minimise la compliance (ie, fonction objectif).

L'approche classique d'optimisation topologique des structures mécaniques est souvent faite à l'aide de la M.E.F. On considèrera parmis les techniques les plus répandues, la méthode d'homogénéisation (Bendsøe et Kikuchi,1988). Dans cette approche, on associe à chaque élément une densité indiquant la présence ou non de matière. Cette densité, $\rho(x_e)$ est traduite numériquement par une valeur entre 0 et 1 pouvant évoluer de manière discrète (ie, 0-1) ou continue (ie, [0,1]). Autrement dit la densité $\rho(x)$ est constante par morceaux, ie $\rho(x) = \chi_\Omega(x)$, $\Omega \subset \Omega_0$ et l'on fait l'hypothèse que l'espace admissible des tenseurs des déformations peut s'écrire sous la forme :

$$E_{AD} = \{E_{ijkl}(x) = E^0_{ijkl}\chi_\Omega(x), |\Omega| \leq V_f\}$$

Cependant cette technique a quelque inconvénient dans le cas où la densité est modélisée de manière continue. Tout d'abord, la présence de la densité intermédiare dans la solution peut être difficile à interpréter pour la réalisation de la pièce (ie, la physique d'une solution peut être difficile à interpréter). On peut alors pénaliser les densités intermédiaires en considérant que la raideur évolue selon une loi avec un exposant p>1 pour la densité. Ainsi l'espace admissible des tenseurs des déformations peut se reécrire sous la forme (Approche SIMP, [5, 38, 35]) :

$$E_{AD} = \{E_{ijkl}(x) = E^0_{ijkl}\rho(x)^p, \int_{\Omega} \rho \leq V_f\}$$

Et il a été prouvé que l'exposant p est lié au coefficient de Poisson ν^0 (qui a son tour est lié au tenseur E^0) par la relation géometrique :

$$p \geq max\left(15\frac{1-\nu^0}{7-5\nu^0}, 1.5\frac{1-\nu^0}{1-2\nu^0}\right)$$

Cette approche permet ainsi à une densité intermédiare de prendre une valeur discrète entre 0-1. Empiriquement, la valeur p=3 est souvent utilisée pour assurer l'existence d'une solution. Mais la solution peut présenter des problèmes d'instabilités numériques : problème à damier, de dépendance aux maillages, de minimum local.

Ces problèmes numériques communs apparaissant en optimisation topologique des structures mécaniques et sont divisé en trois classe de problèmes :

- Problème à damier se réfère à, l'apparition d'une formation de régions, alternant des éléments solide dont la densité est égale 1 et des éléments dont la densité est égale ϵ [11] (et, dont la propriété mécanique est proche du vide)
- Problème de dépendance aux maillages se réfère à, l'obtention de différentes solutions pour différentes tailles de maillages ou discrétisations

11. une petite valeur proche de zéro

— Problème de minimum local se réfère à, l'obtention de solutions différentes au même problème discrétisé, pour différents choix de paramètres algorithmiques.

Un filtrage inspiré du traitement d'images fut proposé par Sigmund en 1999 [32] pour éviter l'apparition d'une structure ayant un aspect de damier (ou plus géneralement de trous en dessous d'une certaine échelle). Cette technique d'homogénéisation, appelée méthode SIMP (Solid Isotropic Material with Penalization, [5, 38, 35]), combinée à une méthode de filtrage, permet de rendre le problème d'optimisation topologique des structures mécaniques bien posé. La pénalisation et le filtrage seront décrits plus en détails dans les sections 1.2.5 et 1.2.6.

En pratique, le problème d'optimisation topologique est formulé comme suit : c'est la formulation equivalente basée sur la M.E.F

Le système equivalent associé à la formulation : a(u,v) = l(v) est donné par l'équation d'équilibre d'élasticité linéaire suite à la discrétisation par M.E.F :

$$K(x)U - F = 0,$$

où $U \in \mathbb{R}^d$ est la variable d'états (ou le vecteur des déplacements nodaux) et $x \in \mathbb{R}^n$ est la variable de conception (on dit aussi de design). La variable x, représente la densité du matériau dans chaque élément du maillage. De plus $F \in \mathbb{R}^d$ est le vecteur de chargement, d est les dégrés de liberté, et n le nombre d'éléments dans le maillage. La matrice de raideur $K(x) : \mathbb{R}^n \to \mathbb{R}^{d\times d}$, et on supose K(x) symétrique, définie positive pour toute variable de conception $x \in [0, 1]^n$. Une petite valeur E_{min} est inclue dans la définition de la matrice de raideur pour une densité $\rho(x_e)$ de valeur nulle pour éviter la singularité de la matrice de raideur liée à la généralisation de méchanisme.

Il existe différentes manières de modéliser le problème d'optimisation topologique. Dans la première approche, la compliance est minimisé sur x et U. Elles sont considérées comme des variables indépendantes.

En outre les équations d'équilibre K(x)U - F = 0 sont explicitement inclus en tant que contraintes d'égalité. La formulation (P_S^c), communément appelée SAND (Simultaneous Analysis and Design) est définie sous la forme :

$$\begin{cases} \min\limits_{x,U} : F^T U \\ K(x)U - F = 0 \qquad (P_S^c) \\ a^T x \leq V_f; 0 \leq x \leq 1 \end{cases}$$

est une modélisation du problème de compliance minimale. Le volume relatif à chaque variable de conception x, est défini par $a \in \mathbb{R}^n$, avec $a_e > 0$, $\forall$ e=1,.n. Finalement $0 < V_f \leq 1$ est la fraction volumique [12]. Le problème à la formulation (P_S^c) ainsi défini, est linéaire avec contraintes d'égalité (non linéaire) et d'inégalité (linéaire). Un Tel problème d'optimisation topologique est géneralement caractérisé comme un problème non convexe ou un problème de programation non linéaire.

Si l'on considére la fonction objectif, comme une fonction non linéaire et que l'on exprime le chargement F, comme la solution des équations d'équilibre K(x)U - F = 0. Alors, le problème de compliance minimale peut également se formuler sous la forme :

$$\begin{cases} \min\limits_{x,U} : U^T K(x) U \\ K(x)U - F = 0 \qquad (P_{SNL}^c) \\ a^T x \leq V_f; 0 \leq x \leq 1 \end{cases}$$

En pratique on n'utilise pas la formulation (P_{SNL}^c) car la performance des solveurs à la formulation SAND est très similaire à cette formulation. Le nombre de contraintes et de variables peuvent être réduits dans le problème de compliance minimale si le problème est formulé en utilisant uniquement la variable de conception. Le déplacement provoqué par la force est déterminé par l'équation d'équilibre :

$$U(x) = K(x)^{-1} F$$

12. Contrainte de volume relative à x

Par conséquent l'équation d'états $U(x) = K(x)^{-1}F$ est ainsi résolue lors de l'évaluation de la fonction objectif. Et la formulation (P_N^c) communément appelée Nested, permet de modéliser le problème de compliance minimale sous la forme :

$$\begin{cases} \underset{x}{min} : U(x)^T K(x) U(x) \\ a^T x \leq V_f; 0 \leq x \leq 1 \qquad (P_N^c) \end{cases}$$

C'est la formulation classique en optimisation topologique des structures mécaniques. Cependant l'évaluation de la fonction objectif, du gradient et de la hessienne est plus coûteuse.

On considèrera la formulation Nested pour modéliser le problème d'optimisation topologique par la suite.

1.2.4 Volume minimal

Les problèmes d'optimisation topologique peuvent également être formulés sous la forme d'une minimisation de la masse de la structure avec contrainte sur la compliance. Similaire au problème de compliance minimale (P_S^c), la formulation SAND au problème de volume minimal (P_S^w) est formulée :

$$\begin{cases} \underset{x}{min} : a^T x \\ K(x)U - F = 0; \quad F^T U \leq C \ (P_S^w) \\ 0 \leq x \leq 1 \end{cases}$$

où C > 0 est une constante donnée, c'est la contrainte sur la compliance. Dans la formulation Nested (P_N^w), l'équation d'équilibre est satisfaite dans la contrainte d'inégalité non linéaire (ie, contrainte sur la compliance) et (P_N^w) est formulé sous la forme :

$$\begin{cases} \underset{x}{min} : a^T x \\ U(x)^T K(x) U(x) \leq C \quad (P_N^w) \\ 0 \leq x \leq 1 \end{cases}$$

Finalement dans la formulation SANDNL (P_{SNL}^w), la contrainte d'égalité et la contrainte d'inégalité sont non linéaires :

$$\begin{cases} \underset{x}{min} : a^T x \\ K(x)U - F = 0; \quad U(x)^T K(x) U(x) \leq C \ (P_{SNL}^w) \\ 0 \leq x \leq 1 \end{cases}$$

Pour les problèmes de volume minimal, la fonction objectif est linéaire mais la non-linéarité apparaît dans les contraintes. (voir. [50], [58])

1.2.5 Méthode SIMP : une approche numérique d'optimisation topologique

Les techniques d'optimisation topologique des structures mécanique sont actuellement dominées par des méthodes qui utilisent la distribution des propriétés matérielles dans un domaine de conception prescrit. Ces méthodes sont basées sur la théorie de l'homogénéisation. Une Première approche d'optimisation toplogique fut la méthode SIMP de Bendsøe et Kikuchi (1988) et depuis le concept a évolué dans différentes directions, notamment : la méthode de densité de (Bendsøe 1989 ; Zhou et Rozvany 1991 ; Mlejnek 1992), la dérivée topologique (Sokołowski et Zochowski 1999) [13], la méthode level set (Allaire et al. 2002, 2004 ; Wang et al.2003), les approches évolutives (Xie et Steven 1993) et plusieurs autres. On considèrera pour le présent travail la méthode SIMP.

13. Outil récent pour l'optimisation de formes

Le processus d'optimisation topologique a souvent pour objectif de déterminer quels éléments de la structure vont conserver les propriétés mécaniques du matériau les constituant à la solution, les autres éléments structuraux étants, eux affectés des propriétés mécaniques proches du vide. En d'autres termes on associe à chaque élément une densité $\rho(x_e)$, indiquant la présence ou non de matière. Cette densité est traduite numériquement par une valeur entre 0 et 1.

Les modèles d'interpolation du matériau sont très populaires dans le milieu d'optimisation topologique pour convertir le problème 0-1 (ie, un problème discret) en un problème continue dans l'intervalle [0,1]. Ces modèles permettent ainsi à une densité intermédiare de prendre une valeur discrète entre 0-1. (voir [7])

La méthode SIMP est l'une des approches les plus courantes. Dans ce modèle, la densité x est remplacée par une loi de puissance. Par contre, la solution du problème continue peut présenter une topologie en "damier" sur les éléments structuraux, que l'approche SIMP ne résout pas. Cependant l'utilisation d'un filtrage inspiré du traitement d'images permet d'éviter l'apparition d'une structure à damier, d'assurer la régularité de la matrice de raideur K(x) et l'existence d'une solution. La matrice de raideur modifiée utilisant la méthode SIMP et le filtre de densité par exemple est sous la forme :

$$K(x) = \sum_{e=1}^{n} (E_v + (E_1 - E_v)\tilde{x}_e^p) K_e,$$

où la variable $\tilde{x}_e$ est le filtre de densité (qui sera définie par la suite), $p \geq 1$ et E_1 et E_v sont les modules de Young du matériau solide et du matériau proche du vide respectivement. Le paramètre E_v est inclu dans la définition de K(x) pour éviter la singularité numérique liée à la suppression complète d'un membre structural (c'est le cas lorque la densité x_e prends une valeur nulle). Cette formulation s'est révélée particulièrement efficace dans de nombreux cas.

Empiriquement, la valeur p=3 est souvent utilisée pour assurer l'existence d'une solution. En pratique cette valeur est souvent utilisée (dans les expériences numériques). Comme illustré dans la figure 2, une alternance rapide de valeurs ϵ et 1 des pseudo-densités $\rho(x_e)$ crée une structure ayant à "damier" et plus le maillage de la structure est fin, plus l'effet damier est important.

1.2.6 Méthodes de filtrage : filtre de densité et de sensibilité

Dans l'optimisation topologique des structures discrétisées par des E.F continus (Bendsøe et Kikuchi 1988 ; Bendsøe 1989), deux problèmes numériques appelés : topologie en "damier" et dépendance aux maillages (Cheng et Olhoff 1981) ont attiré les intérêts de nombreux chercheurs.

Le problème de la dépendance aux maillages, également appelé : non-existence de solutions se réfère à l'obtention de différentes solutions pour différentes tailles de maillages ou discrétisations. Ces deux problèmes artificiels entraîneront des instabilités numériques dans l'optimisation (Sigmund et Petersson 1998) et conduisent à des solutions non admissibles (ou non acceptables). Pour éviter de tels problèmes et obtenir une solution au problème continue acceptables, des méthodes de restriction doivent être appliquées à l'optimisation (Sigmund et Petersson 1998).

Divers types de méthodes de restriction ont été proposés dans l'optimisation topologique afin d'éviter les problèmes de structures ayant un aspect damiers et un problème de dépendance au maillage. Parmi toutes les méthodes de restriction, les méthodes de filtrage sont les plus courrament utilisées en optimisation topologique car elles sont très simples et efficaces. Après le filtre de sensibilité (Sigmund 1994) et le filtre de densité (Bruns 2001 ; Bourdin 2001), de nombreuses autres méthodes de filtrage basées sur ces deux filtres ont été proposées. Mais récemment, Sigmund (2007) a introduit une nouvelle classe de schémas de restriction qui fonctionnent comme des filtres de densité qui permet d'obtenir des solutions admissibles aux problèmes 0-1 avec des contraintes minimales sur la taille des maillages ou discrétisations de la structure par E.F continus. On notera également la nouvelle méthode de filtrage qui consiste à préserver le volume (Guest et al.2004). On considèrera pour le présent travail la nouvelle méthode de filtrage (Guest et al.2004).

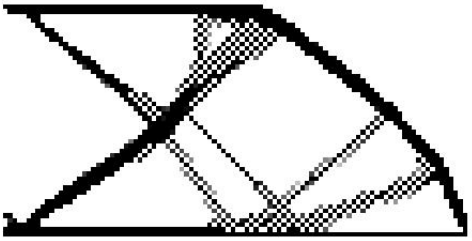


FIGURE 2 – Image de Top88 : Solution avec damier

FIGURE 3 – Image de Top88 : Solution sans damier

1.2.6.1 Filtre de sensibilité

Le filtrage de sensibilité modifie les sensibilités de la fonction objectif aux variables de design $\frac{\partial c}{\partial_{x_e}}$ comme suit :

$$\frac{\hat{\partial c}}{\partial_{x_e}} = \frac{1}{max(\gamma, x_e)\sum_{i\in N_e} H_{ei}} \sum_{i\in N_e} H_{ei} x_i \frac{\partial c}{\partial x_i} \tag{6}$$

Où N_e est l'ensemble des éléments pour lequels $\Delta(e,i)$ est inférieure au rayon de filtre r_{min} et H_{ei} est un facteur de pondération défini comme :

$$H_{ei} = max(0, r_{min} - \Delta(e,i)) \tag{7}$$

Le terme γ $(= 10^{-3})$ dans (6) est un petit nombre positif introduit afin d'éviter la division par zéro. Dans l'approche classique de la méthode SIMP la variable de densité x_e est dans (0,1] et donc le terme γ n'est pas requis.

1.2.6.2 Filtre de densité

Le filtre de densité est défini de la manière suivante :

Étant donné un élément fini e, est définie une variable de conception $\tilde{x}_e$ et appeléé densité physique. La variable $\tilde{x}_e$ dépend d'une moyenne pondérée dans un voisinage du rayon de filtre r_{min} de centre e :

$$\tilde{x}_e = \frac{1}{\sum_{i\in N_e}\overline{H}_{ei}} \sum_{i\in N_e} \overline{H}_{ei} x_i \tag{8}$$

$$\overline{H}_{ei} = max(0, r_{min} - \Delta(e,i)) \tag{9}$$

où $\tilde{x}_e$ est la transformée de la variable de densité x_e de l'élément e, N_e est l'ensemble des éléments pour lesquelles la distance à l'élément e (définie par $\Delta(e,i)$) est plus petit que le rayon de filtre r_{min}. En pratique cette valeur est typiquement fixée à $r_{min} = 0.04L_x$. Où L_x est la longueur du domaine de conception dans la direction des x (dans le repère cartésien).

1.2.6.3 Filtre d'Helmholtz

Afin de traiter éfficacement, des problèmes de grande taille ou d'échelle et des problèmes aux maillages non uniforme, une technique de filtrage a été proposée comme une alternative au filtre de sensibilité dans laquelle une équation différentielle partielle de type Helmholtz est résolue, au lieu de l'habituel filtrage inspiré du traitement d'images (Lazarov et Sigmund 2010). Cette technique de filtrage proposée n'a besoin que d'informations du maillage nécessaires à la discrétisation du problème par éléments finis. L'idée principale est de définir implicitement la variable filtrée comme une solution d'une équation différentielle de type Helmholtz, avec conditions aux limites homogènes de Neumann.

Le filtre de densité donné par (8) peut être implicitement représenté par la solution d'une équation différentielle de type Helmholtz, avec conditions limites homogènes de Neumann :

$$-R_{min}^2 \nabla^2 \tilde{\Psi} + \tilde{\Psi} = \Psi \tag{10}$$

$$\frac{\partial \tilde{\Psi}}{\partial n} = 0 \tag{11}$$

Où Ψ est une réprésentation continue du domaine de conception non filtré et $\tilde{\Psi}$ est le filtré de Ψ. La solution de l'EDP peut s'écrire sous la forme d'un produit de convolution semblable au filtrage classique. Le paramètre R_{min} dans (10) joue un rôle similaire à celui de r_{min} dans (9). Il existe une relation approximative entre les échelles de longueur pour le filtre classique et le filtre d'Helmholtz (Lazarov et Sigmund 2010) :

$$R_{min} = r_{min}/2\sqrt{3} \tag{12}$$

Cette technique préserve le volume de Ψ. La même idée peut-être appliquée pour le filtre de sensibilité avec Ψ dans (10) remplacée par $\Psi = x\frac{\partial c}{\partial x}$ et $\tilde{\Psi} = x\frac{\partial \tilde{c}}{\partial x}$ (Lazarov et Sigmund 2009). Comme nous l'avons dit précedement, cette approche n'a besoin que d'informations du maillage utilisé pour les équations d'équilibre (ie, l'équation d'états) et ne nécessite aucune information supplémentaire, ce qui évite une utilisation excessive de la mémoire. De plus, le coût de calcul dépend linéairement du paramètre R_{min} dans (10) si la solution à l'équation (10) est obtenue par une méthode itérative. Par conséquent, pour un large rayon de filtre, en particulier en trois dimensions, l'approche de filtre d'Helmholz doit être le choix préféré.

1.2.7 Exemple classique d'optimisation topologique : top88 et top3d

Top88 est un code Matlab dont les auteurs sont Erik Andreassen, Anders Clausen, Mattias Schevenels, Boyan S. Lazarov et Ole Sigmund. C'est une version améliorée (en termes de performance de calculs) du code Top99 dont l'auteur est Ole Sigmund. (voir. [53])

L'algorithme est implanté à l'aide de 88 lignes de code [14] destiné à résoudre efficacement un problème classique d'optimisation topologique en 2d. Et top3d est la version réadaptée de top88 pour 3d dont les auteurs sont LIU et TOVAR. (voir. [31])

Top88 génére et résout des problèmes classique de compliance minimale. On considèrera des exemples d'optimisation de structures de type Michell, "MMB beam" [15] et "Cantilever" [16]

Problème 2d : Michell, Cantilever et MBB beam

Le domain de conception prescrit, les conditions limites et le chargement extérieure de la structure de type Michell, Cantilever et MMB beam est mis en évidence par la figure 4.

L'Algorithme top88

(1) La méthode SIMP modifiée [5, 38]

Le domain de conception est discrétisé par élément fini carré. A chaque élement e, est associé une densité x_e et à chaque x_e est associée un module de Young E_e définit comme suit :

$$E_e(x_e) = E_{min} + {x_e}^p(E_0 - E_{min})$$

14. contrairement top99 qui lui est implanté à l'aide de 99 lignes de code
15. Problèmes de MMB encastrée
16. Problèmes de poutre encastrée

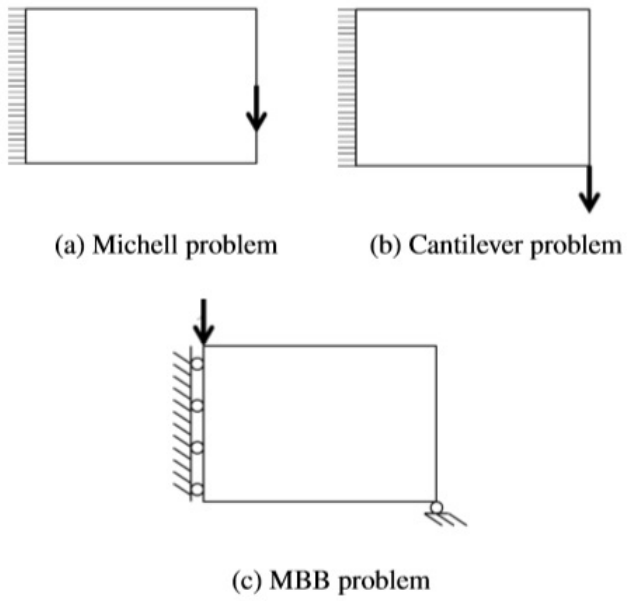


FIGURE 4 – Ensemble tests

Avec $x_e \in [0,1]$, E_0 la raideur du matériau, E_{min} une trés petite valeur de raideur associée aux éléments dont la propriété physique est proche du vide. On veut éviter une singularité dans la matrice de raideur. Et p, la pénalisation (où la valeur p=3 est souvent utilisée) pour convertir le problème 0-1 en un problème continue.

(2) Formulation :

$$\begin{aligned} &\min_{x,U}\ c(x) = U^T K(x) U = \textstyle\sum_{e=1}^{N} E_e(x_e) u_e^T k_0 u_e \\ &\text{sujet à : } V(x)/V_0 = f\,;\ K(x)U = F\,;\ 0 \le x \le 1, \quad (P_S^c) \end{aligned}$$

où c est la compliance, U et F sont les vecteurs de déplacement (globale) et de chargement respectivement, K(x) est la matrice raideur globale, u_e est le vecteur local de déplacement. k_0 est la matrice élémentaire de raideur avec un module de Young égal à 1 et c'est la même pour tous les éléments, x est la variable conception (ou de densité), N est le nombre d'éléments dans le maillage par éléments carré du domain de conception initial, V(x) et V_0 sont respectivement le volume du matériau relatif à la variable de conception x et le volume du domain de conception initial, et f est la fraction volumique prescrite.

(3) Sensibiltés c et V par rapport à la densité x_e

La sensibilité de la fonction objectif c et le volume V par rapport à la densité x_e sont donnés par :

$$\frac{\partial c}{\partial_{x_e}} = -p x_e^{p-1} (E_0 - E_{min}) u_e^T k_0 u_e \tag{13}$$

$$\frac{\partial V}{\partial_{x_e}} = 1 \tag{14}$$

L'équation (14) repose sur l'hypothèse que chaque élément e est associé une densité x_e. C'est à dire V(x) = $a^T x$, avec a=(1,.,1).

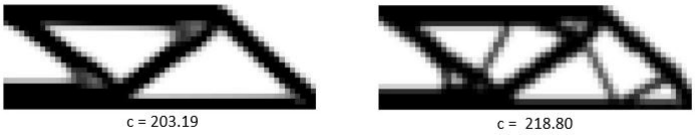


FIGURE 5 – Structure MMB-beam optimisée

(4) Filtrage

Afin de garantir l'existence de solutions au problème d'optimisation topologique et d'éviter l'apparition d'une structure à damier, certaines restrictions sur la conception doivent être imposées. L'une des approches courantes consiste à appliquer un filtre soit de sensibilité ou soit de densité.

Le filtrage de sensibilité modifie les sensibilités de la fonction objectif $\frac{\partial c}{\partial_{x_e}}$ comme dans (6), (7).

Le filtre de densité transforme les denstités x_e comme suit :

$$\tilde{x_e} = \frac{1}{\sum_{i \in N_e} H_{ei}} \sum_{i \in N_e} H_{ei} x_i \tag{15}$$

Dans ce qui suit, les densités x_e sont appélées variables de conception. Le filtre de densité $\tilde{x_e}$ sont appelés densités physiques. On considèrera $\tilde{x_e}$ plutôt que x_e comme solution au problème d'optimisation topologique. Dans le cas où un filtre de densité est appliqué, les sensibilités $\frac{\partial c}{\partial x_e}$ (de la fonction objectif c) et $\frac{\partial V}{\partial x_e}$ (du volume V du matériau) sont toujours donnés par (13) et (14), à condition que la variable x_e est remplacée par la variable $\tilde{x_e}$.

Les sensibilités vis-à-vis des variables de conception x_j sont obtenues au moyen de la règle de dérivation en chaîne ou théorème de dérivation des fonctions composées :

$$\frac{\partial \psi}{\partial x_j} = \sum_{e \in N_j} \frac{\partial \psi}{\partial \tilde{x_e}} \frac{\partial \tilde{x_e}}{\partial x_j} = \sum_{e \in N_j} \frac{1}{\sum_{i \in N_e} H_{ei}} H_{je} \frac{\partial \psi}{\partial \tilde{x_e}} \tag{16}$$

Où ψ représente soit la fonction objectif c, soit le volume V du matériau.

(5) Implémentation Matlab

Le code est appelé à partir de l'invite MATLAB : **top88(nelx,nely,volfrac,penal,rmin,ft)**

Où nelx et nely sont le nombre d'éléments dans les directions horizontale et verticale, respectivement, volfrac est la fraction volumique f prescrite, penal est la puissance de pénalisation p, rmin est le rayon du filtre r_{min} et ft spécifie si l'on veut un filtre de sensibilité (ft=1) ou si l'on veut un filtre de densité (ft=2).

Un exemple de problème d'optimisation topologique en 2d : la "MMB beam" peut être reproduite à l'aide de l'appel de fonction : top88 (60,20,0.5,3,1.5,1)

La structure MMB beam optimisée est montrée par la FIGURE 5, la poutre encastrée par la FIGURE 6, avec ft=1,2.

FIGURE 6 – Structure Cantilever optimisée

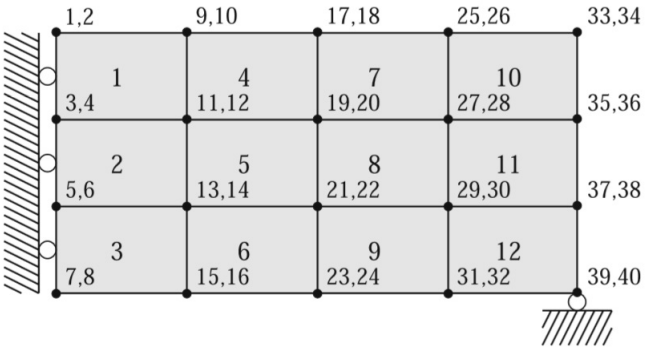


FIGURE 7 – Un exemple de maillage grossier

(5.1) Méthode d'éléments finis

Le domaine de conception est supposé rectangulaire, discrétisé par éléments carrés. Un exemple de maillage grossier composé de 12 éléments avec quatre nœuds par élément et deux degrés de liberté (DDL) par nœud est présenté à l'aide de la figure 7 :

Les nœuds et les éléments sont numérotés par colonne, de gauche à droite, et les DDL 2n-1 et 2n correspondent aux déplacements horizontal et vertical du noeud n, respectivement.

La partie prétraitement par la M.E.F commence par la définition des propriétés du matériau : E0 est le module de Young E_0 du matériau, Emin est le module de Young 'artificiel' E_{min} des éléments dont les propriétés caractérisques du matériau est proche du vide et nu est le coefficient de Poisson ν. Ensuite, la matrice élémentaire k_0 pour un élément e avec un module de Young E0=1 est calculée. Cette matrice est notée KE. En raison de la régularité du maillage, cette matrice est identique pour tous les éléments e, du maillage.

Afin de permettre un assemblage efficace de la matrice de raideur globale dans la boucle de l'optimiseur, une matrice edofMat est construite. La i-ème ligne de cette matrice contient les huit indices DDL correspondant au i-ème élément. La matrice edofMat est construite en trois étapes :

Tout d'abord, une matrice (nely + 1) × (nelx + 1) avec les numéros des nœuds est définie. La fonction MATLAB Reshape est utilisée ; cette fonction retourne une matrice de la taille spécifiée par les deuxième et troisième arguments d'entrée, dont les éléments sont pris à partir du premier argument d'entrée (qui est dans ce cas un vecteur contenant les numéros des nœuds du maillage). Ensuite, la matrice nodenrs est utilisée pour déterminer le premier indice DDL pour tous les éléments du maillage par E.F et sont stockés dans un vecteur edofVec. Enfin, le vecteur edofVec est utilisé pour déterminer les huit indices DDL pour chaque élément e. Pour cela, la fonction MATLAB repmat est appelée deux fois. Cette fonction copie une matrice en un nombre de fois spécifié(s) dans les sens vertical et horizontal.

Le premier appel à la fonction repmat renvoie une matrice à huit colonnes, qui sont toutes des copies du vecteur edofVec. Le deuxième appel à la fonction repmat renvoie une matrice de même

$$\texttt{edofMat} = \begin{bmatrix} 3 & 4 & 11 & 12 & 9 & 10 & 1 & 2 \\ 5 & 6 & 13 & 14 & 11 & 12 & 3 & 4 \\ 7 & 8 & 15 & 16 & 13 & 14 & 5 & 6 \\ 11 & 12 & 19 & 20 & 17 & 18 & 9 & 10 \\ \vdots & \vdots & \vdots & \vdots & & & & \\ 31 & 32 & 39 & 40 & 37 & 38 & 29 & 30 \end{bmatrix} \begin{matrix} \leftarrow \text{Element 1} \\ \leftarrow \text{Element 2} \\ \leftarrow \text{Element 3} \\ \leftarrow \text{Element 4} \\ \\ \leftarrow \text{Element 12} \end{matrix}$$

FIGURE 8 – Matrice edofMat

taille où toutes les lignes sont identiques ; cette matrice relie les indices des huit DDL d'un élément e, à l'indice de son premier DDL stocké dans le vecteur edofVec. Les résultats sont additionnés et rassemblés dans la matrice edofMat.

Pour l'exemple du maillage grossier illustré par figure 7, cette procédure donne le résultat illustré par la figure 8.

A chaque itération de la boucle de l'optimiseur, l'assemblage de la matrice de raideur globale K est effectué à l'aide de la fonction MATLAB sparse. La procédure suivie ici est inspirée de l'approche décrite par Davis (2007). La fonction sparse prend comme arguments d'entrée trois vecteurs : les premier et second contiennent les indices de ligne et de colonne des entrées de matrice non nulles, qui sont collectés dans le troisième vecteur. Les vecteurs d'indices de ligne et de colonne (iK et jK, respectivement) sont créés à partir de la matrice edofMat. On utilise un produit matriciel (à l'aide de la fonction Matlab Kronecker) avec un vecteur unitaire de taille 8 suivi d'un "resharpe" (à l'aide de la fonction Matlab reshape).

Les vecteurs iK et jK sont structurés de sorte que les indices iK (k) et jK (k) correspondent à la (i,j) ème entrée de la matrice de raideur pour l'élément e, où k = i +8 (j -1) +64 (e -1). Le troisième vecteur, contenant les entrées de la matrice de raideur (matrice creuse), est calculé dans la boucle de l'optimiseur, car cela dépend des densités physiques $\tilde{x_e}$. Ce vecteur sK, est obtenu via un "reshape" de la matrice élémentaire KE pour obtenir un vecteur colonne. En multipliant ce vecteur avec le module de Young $E_e(\tilde{x_e})$ pour chaque élément e, suivi d'une opération de concaténation des résultats pour tous les éléments. La multiplication et la concaténation sont mises en œuvre sous forme de produit matriciel suivi d'un "reshape" (à l'aide de fonction Matlab reshape). On fait en sorte que la matrice de raideur obtenue soit symétrique via l'opération K = (K + K')/2, où K' est la transposée de K. Finalement les conditions aux limites et le vecteur de chargement sont calculés et le système d'équations d'équilibre est enfin résolu par une méthode directe (opérateur \ dans Matlab).

(5.2) Filtrage

L'application d'un filtre de sensibilité selon (6) implique une moyenne pondérée sur différents éléments. C'est une opération linéaire ; elle peut donc être mise en œuvre en tant que produit matriciel d'une matrice de coefficients et d'un vecteur contenant les sensibilités $\frac{\partial c}{\partial x_i}$ (multiplié par les variables de conception x_i). Diviser le résultat par un facteur $max(\gamma, x_e) \sum_{i \in N_e} H_{ei}$ donne $\frac{\hat{\partial} c}{\partial_{x_e}}$. La matrice H et le vecteur Hs contiennent les coefficients H_{ei} et les constantes de normalisation $\sum_{i \in N_e} H_{ei}$, repectivement. L'utilisation d'un filtre de densité n'implique pas seulement le filtrage des densités selon (15) mais aussi une modification de la règle de dérivation en chaîne des sensibilités de la fonction objectif et de la contrainte de volume selon (16). Les deux opérations impliquent une moyenne pondérée sur différents éléments du maillage E.F.

La matrice H et le vecteur Hs restent invariants lors de l'optimisation et sont calculés a priori. La matrice H de taille (nelx × nely) × (nelx × nely) établit une relation entre tous les éléments.

FIGURE 9 – Domain de conception initial

Cependant, comme défini dans (7), le noyau du filtre a un support borné, donc seuls les éléments voisins sont concernés. En conséquence, la majorité des coefficients est nulle et donc H est une matrice creuse. Elle est construite à l'aide de la fonction MATLAB sparse. Les vecteurs d'indices de lignes et de colonnes iH et jH ainsi que le vecteur SH avec des entrées non nulles sont assemblés au moyen de quatre boucles imbriquées.

(5.3) Boucle de l'optimiseur

La partie principale du code top88 est la boucle de l'optimiseur. La boucle est initialisée, toutes les variables de conception x_e sont initialement définies égales à la fraction volumique f prescrite. Les densités physiques $\tilde{x_e}$ sont identiques aux variables x_e. Si un filtrage de sensibilité est utilsé, cette égalité est toujours valable, alors que si un filtre de densité est utilisé, elle est valable tant que les variables x_e sont homogénéisés. Chaque itération de la boucle de l'optimiseur commence par l'analyse par E.F décrite dans (5.1). Ensuite, la fonction objectif c est calculée, ainsi que les sensibilités dc et dv respectivement de c et V par rapport à $\tilde{x_e}$.

La matrice edofMat est utilisée efficacement pour calculer simultanément la compliance c de tous les éléments : elle est utilisée sous forme d'indices pour le vecteur de déplacement. Il en résulte une matrice de taille n (= size(edofMat)), contenant les déplacements correspondant aux DDL énumérés dans edofMat. Les sensibilités sont ensuite filtrées (si le filtre de sensibilité est utilisé) ou modifiées (si le filtre de densité est utilisé) comme expliqué dans (4).

(5.4) Le Solveur d'optimisation

La structure MMB beam en exemple a été optimisée avec le solveur empirique OC (optimality criteria). Les images des structures optimisées sont présentés à l'aide de la figure 10. La méthode OC est utilisée pour mettre à jour les variables de conception. Enfin, les résultats intermédiaires sont tracés.

Optimisation topologique des structures en 2d dans Matlab à l'aide de Top88

La boucle de l'optimiseur est terminée lorsque la norme L_∞ de la différence relative entre deux conceptions consécutives (en termes de variables de conception : x^{k+1} et x^k) est inférieure à la valeur ch = 0.01.

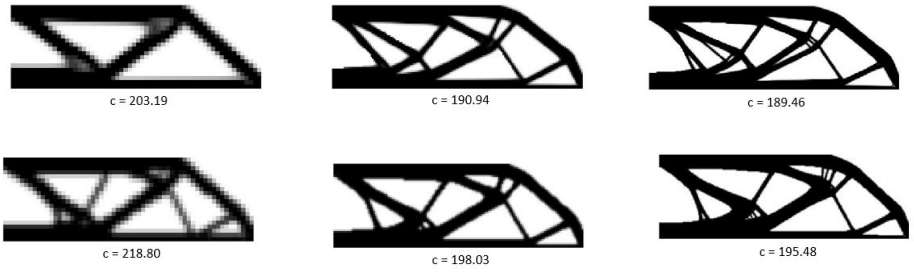


FIGURE 10 – Structure "MBB beam" optimisée et la compliance correspondante c obtenues avec Top88 utilisant le filtre de sensibilité (en haut) et le filtre de densité (en bas). Un maillage avec 60 × 20 éléments (à gauche), 150 × 50 éléments (au centre) et 300 × 100 éléments (à droite) a été utilisé

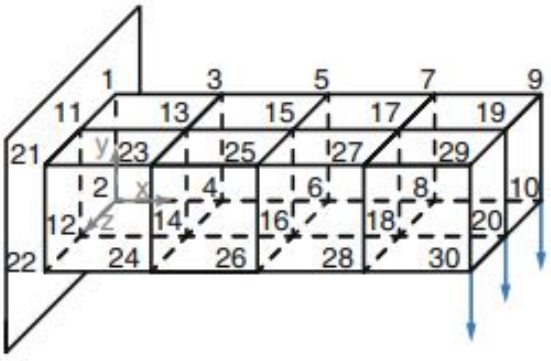


FIGURE 11 – Un exemple de maillage grossier en 3d

Optimisation topologique des structures en 3d dans Matlab à l'aide de Top3d

Le domaine de conception est supposé parallélépipède rectangulaire, discrétisé par éléments cubique. Un exemple de maillage grossier composé de 8 éléments avec huits nœuds par élément et trois degrés de liberté (DDL) par nœud est présenté à l'aide de la figure 11 :

Les noeuds sont numérotés et ordonnés par colonnes de haut en bas, de gauche à droite et de bas en haut. La position de chaque nœud est définie par rapport au système de coordonnée cartésienne avec origine, le coin gauche en bas à l'arrière. Au sein de chaque élément e, les huit nœuds N1, ., N8 sont ordonnés dans le sens anti-horaire comme indiqué par la figure 12. Notez que la numérotation "locale" des noeuds (Ni) ne suit pas la même règle que la numérotation «globale» des noeuds. Étant donné le volume (nelx × nely × nelz) du conception initial et les coordonnées globales du noeud N1 (x1, y1, z1), on peut identifier les coordonnées du noeud global et la numérotation globales des sept autres nœuds dans cet élément par cartographie comme résumé dans le tableau : figure 13.

Chaque nœud de la structure a trois degrés de liberté correspondant aux déplacements linéaires dans les directions x-y-z (un élément a 24 DDL). Les degrés de liberté s'organisent dans le vecteur de déplacement nodal U de la façon suivante : $U = [U_{1x}, U_{1y}, U1z, ., U_{8\times nz}]$. Enfin, le système d'équations d'équilibre est résolu par une méthode directe.

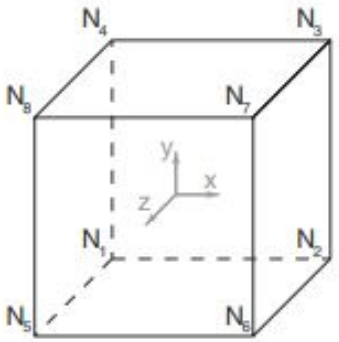


FIGURE 12 – Numérotation local

Node	ξ_1	ξ_2	ξ_3
1	−1	−1	−1
2	+1	−1	−1
3	+1	+1	−1
4	−1	+1	−1
5	−1	−1	+1
6	+1	−1	+1
7	+1	+1	+1
8	−1	+1	+1

FIGURE 13 – tableau des coordonnées des nœuds globales

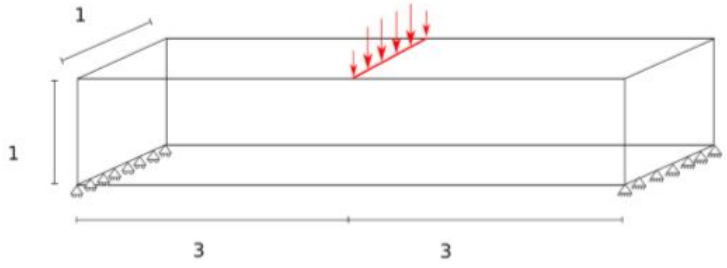


FIGURE 14 – Exemple 3D MBB beam

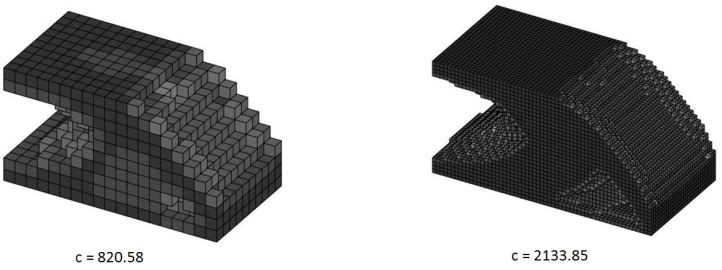


FIGURE 15 – Structure "MBB beam" optimisée et la compliance correspondante c obtenues avec Top3d utilisant le filtre d'Helmholz. Un maillage avec 19×10x10 éléments (à gauche), et 59×30x30 éléments (à droite) a été utilisé

1.2.8 Analyse structurale : techniques d'approximation

Cette section est inspirée de la thèse de Simone CONIGLIO, [14].

Résoudre l'équation d'équilibre : $KU - F = 0$ pour la variable d'états U est une tâche exigeante, en particulier pour des problèmes en 3d, de grande taille comme celui que nous essayons d'aborder dans ce travail. La matrice de raideur est de grande taille, creuse, symétrique, et définie positive. Pour cette raison, on peut soit employer des méthodes directes (Davis, 2006,[1]) ou soit des méthodes itératives (Saad, 2003,[51]).

Quand on traite des petits problèmes en 2d, les solveurs directs sont préférables en raison de leur efficacité et de leur robustesse. Dans ces approches, la matrice de raideur est décomposée en un produit d'une matrice triangulaire inférieure et d'une matrice triangulaire supérieure (ie, Décomposition LU ou Décompositions de Cholesky). Ensuite, l'inversion de ces deux matrices est réalisée en deux étapes sous la forme d'une opération de substitution de descente-remontée. Pour les problèmes en 3d à grande échelle, ces méthodes deviennent trop coûteuses et inefficaces. Dans ces cas, la faible densité de la matrice creuse conduit à une opération matrice-vecteur peu couteuse, qui peut-être utilisée efficacement par des méthodes itératives. On considèrera en particulier une méthode itérative préconditionnée [14], appélée multi-grid CG.

1.2.8.1 Algorithme du gradient conjugué : CG

Pour des matrices symétriques, définies positives, un choix classique est l'algorithme CG (Hestenes et Stiefel, 1952). Nous décrivons ici une version dérivée de cet algorithme en suivant la procédure illustrée dans (Saad, 2003,[51]). A chaque itération k de l'algorithme CG, le vecteur de solution U est mis à jour comme suit :

$$U_{k+1} = U_k + \alpha_k p_k \tag{17}$$

Où U_{k+1} est la solution à l'itération k+1, fonction de U_k, α_k est un scalaire et p_k vecteur de pertubation de la direction à l'itération k. A chaque itération, on peut définir le vecteur résidu r :

$$r_k = F - [K]U_k \tag{18}$$

Le résidu à l'itération k+1 peut être alors écrit comme suit :

$$r_{k+1} = F - [K]U_k - \alpha_k[K]p_k = r_k - \alpha_k[K]p_k \tag{19}$$

On veut aussi que le résidu l'itération k+1 soit orthogonal au résidu à l'itération k :

$$r_{k+1}^T r_k = 0 \tag{20}$$

Cela n'est vrai que si :

$$\alpha_k = \frac{r_k^T r_k}{r_k^T [K] p_k} \tag{21}$$

Pour la mise à jour du vecteur de direction, on veut qu'il appartient au plan engendré par r_{k+1} et p_k :

$$p_{k+1} = r_{k+1} + \beta_k p_k \tag{22}$$

Pour déterminer β_k, on veut que p_{k+1} soit orthogonale à $[K]p_k$:

$$p_k^T [K] p_{k+1} = 0 \tag{23}$$

Et cela n'est vrai si :

$$\beta_k = -\frac{p_k^T [K] r_{k+1}}{p_k^T [K] p_k} \tag{24}$$

Sachant que :

$$[K] p_k = -\frac{1}{\alpha_k}(r_{k+1} - r_k) \tag{25}$$

Ainsi :

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{\alpha_k p_k^T [K] p_k} \tag{26}$$

En utilisant les équations (22) et (23) :

$$p_k^T [K] p_k = p_k^T [K](r_k + \beta_{k-1} p_{k-1}) = p_k^T [K] r_k \tag{27}$$

Alors :

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \tag{28}$$

L'algorithme CG est presenté ci-dessous dans la figure 16 :

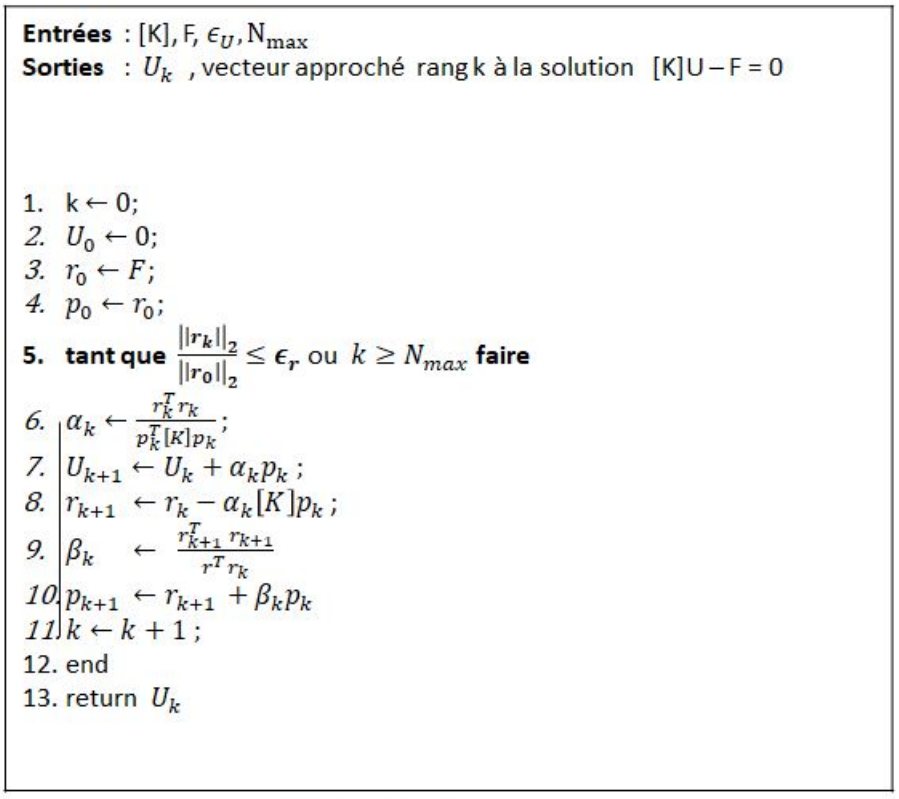


FIGURE 16 – Gradient conjugué

1.2.8.2 Algorithme du gradient conjugué préconditionné : PCG

Il peut être montré dans (Saad, 2003,[51]) que, la méthode CG atteint le vecteur solution U^* en moins de n [17] itérations, et le taux de convergence est donné par :

$$||U_k - U^*||_{[K]} =\leq 2(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^k ||U_0 - U^*||_{[K]} \tag{29}$$

Où la [K]-norme d'un vecteur x, $||x||_{[K]}$ est définie comme suit :

$$||x||_{[K]} = \sqrt{x^T [K] x}, \tag{30}$$

k est l'itération où l'algorithme CG s'est arrêté, κ est le conditionnement de la matrice de raideur [K]. L'équation (29) montre que la méthode CG a une convergence superlinéaire et, si $\kappa \gg 1$, la convergence de CG est trés lente. En pratique, on utilise des techniques de préconditionnement pour accélérer et assurer la convergence. L'idée de base de ces techniques est de remplacer le système d'équations d'origine par un autre qui a la même solution et qui est plus facile à résoudre à l'aide des méthodes itératives.

Un opérateur $[M]^{-1}$ doit être construit de telle sorte que : $\kappa([M]^{-1}[K]) \ll \kappa([K])$. En utilisant ces techniques, l'algorithme PCG est presenté ci-dessous dans la figure (17) :

17. n := taille du vecteur de chargement F, ie nombre de DDL

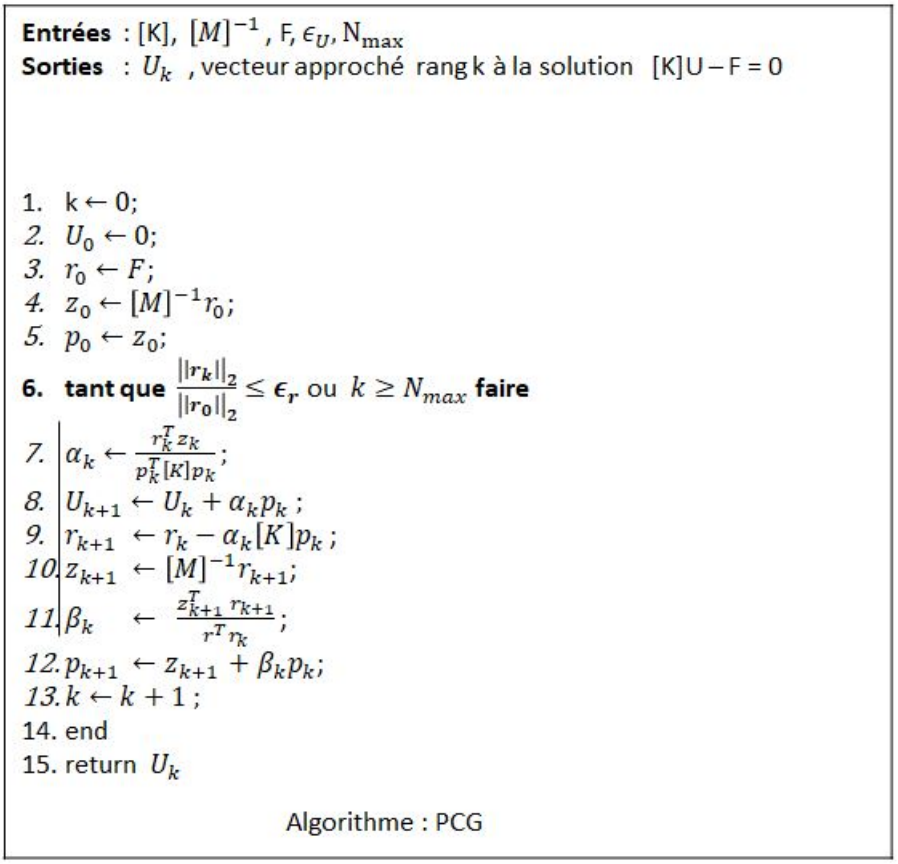


FIGURE 17 – Gradient conjugué préconditionné

On peut observer que l'application de l'opérateur $[M]^{-1}$ est nécessaire qu'une seule fois par itération. Pour cette raison, un bon opérateur de préconditionnement devrait être :

- peu couteux : ne devrait pas impliquer trop d'opérations
- Efficace : $\kappa([M]^{-1}[K])$ devrait être aussi proche que possible de 1.
- Scalable : $\kappa([M]^{-1}[K])$ et son temps de calcul ne devrait pas dépendre de la taille de [K]
- Robuste : L'efficacité de $[M]^{-1}$ ne devrait pas être problème dépendant.
- Simple : La complexité devrait être justifiée par un gain de calcul

Parmi les techniques les plus courantes, on peut trouver la factorisation incomplète de Cholesky (Kershaw, 1978,[28]), la compensation diagonale, la méthode FSAI (Factorized Sparse Approximate Inverses). Toutes ces techniques sont bien adaptées lorsque la matrice n'est pas liée à un problème physique particulier. On considèrera une méthode PCG appélée : Multi-grid CG, qui a l'avantage du fait que la matrice de raideur [K] provient d'une discrétisation d'une EDP.

1.2.8.3 Algorithme du Multi-grid CG : MGCG [43]

La vistesse de convergence de la méthode PCG pour la résolution de systèmes linéaires issus d'une EDP dépend de la taille du système. Les méthodes MGCG (Tatebe et Oyanagi, 1994,[45],Ashby et Falgout, 1996,[57]) ont été initialement conçues spécialement pour résoudre ces problèmes. D'autre part, les succès de ses méthodes dépendent fortement du problème et elles ont besoin de beaucoup

plus d'attention pour obtenir de meilleures performances. Elles nécessitent des corrections plus lisses et des corrections sur des maillages grossiers.

En choisissant le vecteur résidu associé à l'erreur relative à chaque itération, plutôt que la base des vecteurs propres de la matrice [K], on peut montrer que les composantes du vecteur résidu associées à des hautes fréquences, sont amortis en quelques itérations par des techniques simples de lissage comme la méthode de Jacobi, de Gauss Seidel, ou de Red-Black (Saad, 2003,[51]). Mais, les composantes du vecteur résidu associées à des basses fréquences survivent à l'étape de lissage et doivent subir des corrections sur un maillage grossier. Les preuves de ces résultats peuvent être trouvées dans (Saad, 2003,[51]), pour une discrétisation uniforme en 1d et 2d. Dans le présent travail, on se concentre sur des E.F discrétisés par éléments cubique. Nous avons considéré le maillage fin comme une succéssion de raffinements de maillages grossiers. Chaque raffinement de maillage est considéré comme un niveau dans la méthode GMG (Geometric Multigrid). De cette manière, il est relativement simple de construire des opérateurs d'interpolation et de prolongation pour transférer un vecteur à travers les niveaux. Étant donné un maillage grossier, son raffinement peut être obtenu en coupant chaque élément fini en 8 comme illustré dans la figure 18 ([14], Simone CONIGLIO).

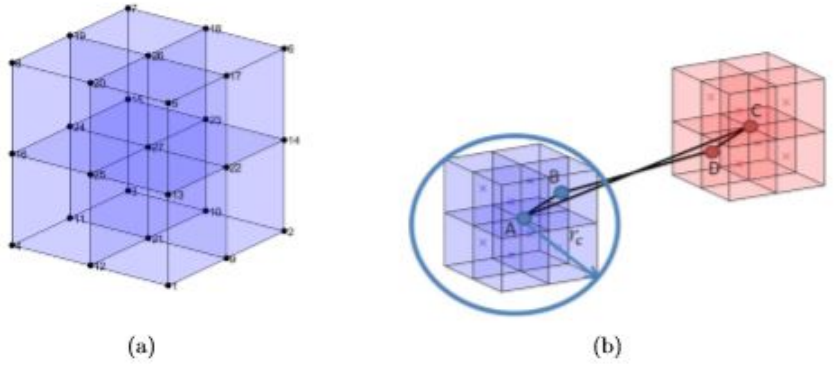


FIGURE 18 – Relation entre maillage grossier et maillage fin. (a) partition 3D utilisée pour le raffinement du maillage, les nœuds 1 à 8 appartiennent à l'élément du maillage grossier d'origine. La partition détermine les nœuds 9 à 27 et 8 éléments du maillage plus fin. (b) Schéma des deux éléments du maillage d'origine après partition

(Simone Coniglio, Christian Gogu, Remi Amargier, Joseph Morlier [56])

Pour chaque élément du maillage grossier on peut définir un transfert de données entre le niveau du maillage grossier u_i^c, i=1,..,8 et le niveau du maillage plus fin u_j^f, j=1,2,..,27, qui peut être utilisé pour construire l'opérateur d'interpolation. Nœuds 1 à 8 du maillage fin ont leur noeud correspondant dans le niveau grossier de sorte que : $u_i^c = u_i^f$, i=1,..8. Nœuds 9 à 20 sont au milieu du bord de l'élément grossier. Pour cette raison, le domaine restreint peut être interpolé en utilisant les extrémités des arêtes. Par exemple $u_9^f = \frac{u_1^c + u_2^c}{2}$. Nœuds 21 à 26 correspondent aux points milieux des faces de l'élément grossier. Le composant correspondant peut être interpolé par le sommet des faces correspondant à l'élément grossier. Par exemple $u_{21}^f = \frac{u_1^c + u_2^c + u_5^c + u_6^c}{4}$. Enfin, le noeud 27 se situe au centre de l'élément grossier. Sa variable nodale peut être interpolée en utilisant toutes les variables des nœuds du maillage grossier : $u_{27}^f = \frac{u_1^c + u_2^c + u_3^c + u_4^c + u_5^c + u_6^c + u_7^c + u_8^c}{8}$.

Toutes les relations précédentes peuvent être écrites sous forme matriciel :

$$\{u_f^{el}\} = [I_{fc}^{el}]\{u_c^{el}\} \tag{31}$$

Lors d'une communication entre différents niveaux grossiers, de domaines restreints indépendants, on peut utiliser le même opérateur d'interpolation pour chaque domaine restreint :

$$\{U_f^{el}\} = [I_{fc}^{el} \otimes I_d]\{U_c^{el}\} \tag{32}$$

Enfin, il est possible de combiner les équations issues de chaque élément de niveau grossier pour construire l'opérateur d'interpolation global. En faisant cela, on doit considérer chaque DDL du maillage fin une seule fois. De manière générale, l'interpolation entre l-ième raffinement et le niveau grossier correspondant est donné par :

$$\{U_l\} = [I_{l,l-1}]\{U_{l-1}\} \tag{33}$$

L'opérateur de prolongation permet la communication inverse :

$$\{U_{l-1}\} = [P_{l,l-1}]\{U_l\} \tag{34}$$

Pour garder la matrice de raideur à chaque raffinement, symétrique, on considère que l'opérateur de prolongation est :

$$[P_{l-1,l}] = [I_{l,l-1}]^T \tag{35}$$

De cette façon, étant donné l'équation d'état au l-ième raffinement :

$$[K_l]\{U_l\} = \{F_l\} \tag{36}$$

on peut obtenir l'équation d'état au niveau grossier :

$$[I_{l,l-1}]^T[K_l]\{U_l\} = [K_{l-1}]\{U_{l-1}\} = [I_{l,l-1}]^T\{F_l\} = \{F_{l-1}\} \tag{37}$$

Cette notation sera utilisée pour définir l'étape de correction du maillage grossier nécessaire dans l'algorithme multi-grid CG. L'autre ingrédient important qui joue un rôle fondamental dans la convergence est le lissage. On notera :

$$\{U_{l,v}\} = smooth^v([K_l], \{U_0\}, \{F_{l-1}\}) \tag{38}$$

indiquant l'application de l'étape v :

$$\{U_{l,j+1}\} = [B_l]\{U_{l,j}\} + [C_l]\{F_l\} \tag{39}$$

Où la matrice $[B_l]$ et $[C_l]$ dépend de l'approche particulière choisie pour le lissage et sera détaillé plus tard. On considèrera l'algorithme MGCG à 4 niveaux, appelé cycle-V. L'algorithme MGCG à 4 niveaux est presenté ci-dessous dans la figure 19 : [14], Simone CONIGLIO

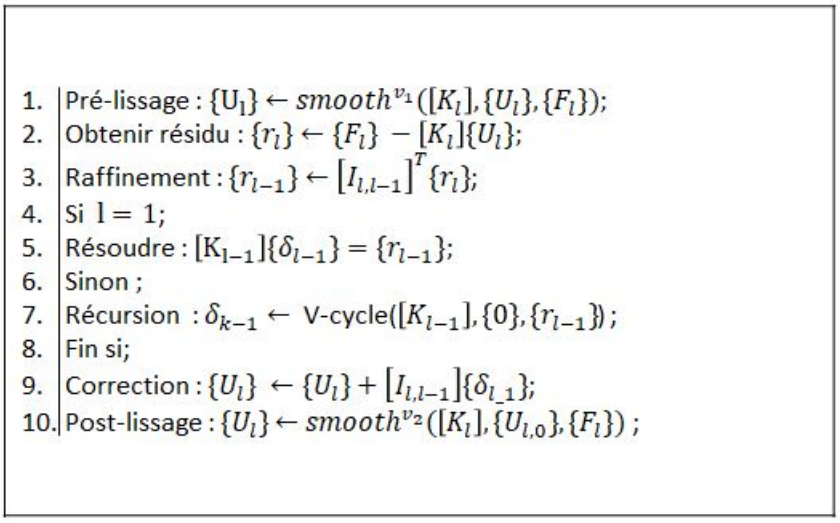


FIGURE 19 – MGCG V-cyle

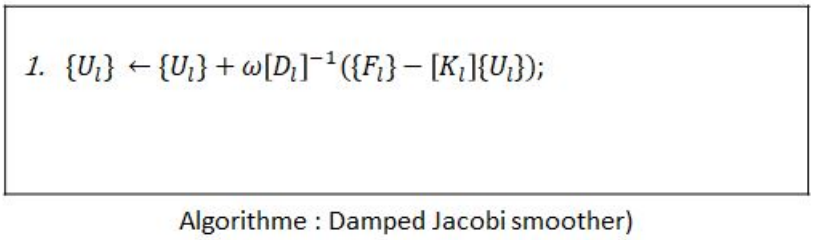


FIGURE 20 – Technique lissage de Jacobi

On peut observer dans le V-cylce, une seule opération matrice fois vecteur à chaque étape de prolongation et d'interpolation. Comme mentionné ci-dessus, le choix de lissage doit être complémentaire aux corrections des maillages grossiers (en termes d'erreurs du vecteur résidu). La technique de lissage "Damped" Jacobi (Damped Jacobi smoother) est la plus efficace, en particulier pour des problèmes de grande taille. Cette technique est presentée ci-dessous dans la figure 20.

Où $[D_l]$ est la matrice diagonale de $[K_l]$. Un autre problème concerne le choix du paramètre d'amortissement ω. Des considérations théoriques peuvent être appliquées pour les maillages uniformes comme expliqué dans (Saad, 2003,[51])

1.2.9 Optimiseurs : techniques d'optimisation

Comme expliqué précédemment, l'analyse de la structure est effectuée par des E.F et le paramétrage s'effectue dans ce cadre. Les différents problèmes d'optimisation des structures y sont liés. Certaines techniques d'optimisation proviennent du domaine des mathématiques pures, où des algorithmes généraux ont été développés (par ex. les solveurs dans IPOPT, NLOPT) sans que le but initial ait été de les appliquer au cas des structures. D'autres méthodes viennent du domaine de la gestion.

Nous ne les utiliserons pas dans ce cas, car elles ne sont pas réllement adaptées à l'optimisation des structures. Enfin la classe de méthodes directement liée aux structures, mise au point par des spécialistes (comme la méthode OC, MMA, GCMMA). Cette classe de méthodes dépend fortement du problème d'optimisation traité. Cependant dans la classe des méthode SCP (Sequantial Convex Programming), il y a aussi des méthodes d'ordre 2, qui permettent de reformuler les méthodes SCP sous la forme d'une série de sous-problèmes QP (Quadratic Programming) de Lagrange-Newton [30]. Mais pour le présent travail, on considèrera que, la classe de méthodes SCP d'ordre 1.

On considèrera trois grandes classes d'optimisation : les critères d'optimalité, les méthodes de programmation mathématiques avec gradient, et la classe de méthodes qui dérive des méthodes de programmation mathématiques : la programmation séquentielle convexe.

1.2.7.1 Solveurs d'optimisation topologique classiques

Comme expliqué précédemment, certaines méthodes sont directement liées aux structures et sont mises au point par des spécialistes. Trés spécifiques et dépendant fortement du problème traité, on peut les classer dans la catégorie des solveurs classiques d'optimisation topologique. Dans le présent travail, on considèrera trois des solveurs les plus courrament utilisés en optimisation topologique.

1.2.7.1.1. Optimiseur OC

Les critères d'optimalité sont des méthodes d'optimisation dédiées à la solution de problèmes spécifiques. Autrement dit, ils sont mis au point au cas par cas. En particulier la méthode OC (Optimality Criteria), dévéloppée pour l'optimisation topologique.

L'avantage de la méthode OC est sa simplicité. Mais bien que la méthode OC a permis de résoudre de nombreux problèmes d'optimisation topologique, il faut constater que la communauté scientifique et des utilisateurs industriels tendent à se détourner d'elle, probablement à cause de son manque de généralité et sa faible éfficacité.

Pour simplifier, nous allons utiliser ici une méthode OC standard. Selon Bendsøe ([6]), un schéma heuristique de mise à jour des variables de conception peut être formulé comme suit :

$$x_e^{new} = \begin{cases} max(x_{min}, x_e - m) & \text{si } x_e B_e^{\eta} \leq max(x_{min}, x_e - m) \\ x_e B_e^{\eta} & \text{si } max(x_{min}, x_e - m) < x_e B_e^{\eta} \leq min(1, x_e + m) \\ min(1, x_e + m) & \text{si } min(1, x_e + m) \leq x_e B_e^{\eta} \end{cases} \tag{40}$$

Où m est un coefficent positif de déplacement limite d'asymptote (positive move-limit), η (= 1/2) est un coefficient d'amortissement et B_e est trouvé à partir de la condition d'optimalité :

$$B_e = \frac{-\frac{\partial c}{\partial x_e}}{\lambda \frac{\partial V}{\partial x_e}} \tag{41}$$

Où λ est le multiplicateur de Lagrange liée à la contrainte volumique. Cela peut être trouvé par un algorithme de dichotomie.

1.2.7.1.2. Méthode de programmation séquentielle convexe

Les méthodes de programmation mathématiques (qui seront présentées ci-aprés) sont des méthodes génériques mais qui peuvent parfois manquer en éfficacité sur les problèmes d'optimisations comparés aux algorithmes spécifiques. Telles quelles, ces méthodes sont souvent trop couteuses, en particulier pour des problèmes de grande taille. Ainsi des méthodes (datant de 1987) inspirées des méthodes de programmation mathématiques ont été développées comme la méthode MMA, GCMMA. On décrit ici les algorithmes de (Krister Svanberg, 2007,[59]), mises en œuvre par les auteurs dans Matlab.

1.2.7.1.2.1 Optimiseur MMA selon (Krister Svanberg, 2007,[59])

On considère le problème d'optimisation suivant :

$$min : f_0(x) + a_0 z + \sum_{i=1}^{m}(c_i y_i + \frac{1}{2} d_i y_i^2) \tag{42}$$

$$\text{sujet à } : f_i(x) - a_i z - y_i \leq 0, i = 1, .m \tag{43}$$

$$x \in X, y \leq 0, z \leq 0. \tag{44}$$

Où $x = (x_1, ., x_n)^T \in \mathbb{R}^n$, $y = (y_1, ., y_m)^T \in \mathbb{R}^m$, et $z \in \mathbb{R}$. Ici $X = \{x \in \mathbb{R}^n | x_j^{min} \leq x_j \leq x_j^{max}, j = 1, .., n\}$, où x_j^{min} et x_j^{max} sont des réels données, qui satisfont $x_j^{min} \leq x_j^{max}$, $\forall j$, $f_0, f_1, ., f_m$ sont des fonctions données, continument différentiable, à valeurs réelles, définies sur X, a_0, a_i, c_i et d_i sont des constantes réelles données, qui satisfont $a_0 > 0$, $a_i \geq 0$, $c_i \geq 0$, $d_i \geq 0$ et $c_i + d_i > 0$ pour tous i, et $a_i c_i > a_0$ pour tous i, avec $a_i > 0$.

Dans (42) les "vraies" variables de conception (ou d'optimisation) sont $x_1, ., x_n$, tandis que $y_1, .., y_m$ et z sont des variables "artificielles" d'optimisation, qui devront permettre de formuler et de résoudre plus facilement certaines sous-classes de problèmes, tels que les problèmes de moindres carrés et les problèmes de minmax.

Comme premier exemple, supposons que l'on veuille résoudre un problème sous une forme «standard» pour la programmation non linéaire suivant :

$$min : f_0(x) \tag{45}$$

$$\text{sujet à } : f_i(x) \leq 0, i = 1, .m \tag{46}$$

$$x \in X, \tag{47}$$

Où $f_0, f_1, ., f_m$ sont des fonctions différentiables et X est comme ci-dessus. Pour que le problème (42) soit (presque) équivalent au problème (45), d'abord il faut $a_0 = 1, a_i = 0$ pour tous i. Alors z = 0 pour toute solution optimale de (42). En outre, pour chaque i, il faut $d_i = 1$ et $c_i =$ "une trés grande valeur réélle", de sorte que les variables y_i deviennent «chères». Alors y = 0 pour toute solution optimale de (42), et le x correspondant est une solution optimale du problème (45).

Il convient de noter que le problème (42) a toujours des solutions admissible, et a au moins une solution optimale. Ceci est valable même si le problème (45) n'a aucune solution admissible, dans ce cas, certaines $y_i > 0$ sont positives dans la solution optimale du problème (42).

Dans le présent travail, les contraintes sont sous la forme $g_i(x) \leq g_i^{max}$, où $g_i(x)$ est une contrainte volumique, tandis que g_i^{max} est la fraction volumique prescrite, cela signifie que $f_i(x) = g_i - g_i^{max}$ dans (42), aussi bien que dans (45). Dans ce cas, on doit redimensionner les contraintes de manière à ce que $1 \leq g_i^{max} \leq 100$ pour chaque i (et non $g_i^{max} = 10^{10}$). La fonction objectif doit alors être redimensionnée de manière à ce que $1 \leq f_0(x) \leq 100$ pour des valeurs raisonnables sur les variables. Les variables x_j doivent aussi être redimensionnées de manière à ce que $0.1 \leq x_j^{max} - x_j^{min} \leq 100$, pour tous j.

En ce qui concerne les «grands nombres» c_i (mentionnés ci-dessus), pour des raisons numériques, on évite de prendre des valeurs «extrêmement grandes» (comme par ex. 10^{10}). Il est préférable de commencer avec des valeurs «raisonnablement grandes» et ensuite, s'il s'avère que les $y_i = 0$ dans

la solution optimale (42), ne sont pas tous nuls, on augmente alors les valeurs correspondantes de c_i, par ex. par un facteur 100 et on résoud de nouveau le problème, etc.

MMA est une méthode de résolution de problèmes de type (42), utilisant l'approche suivante : À chaque itération, le point d'itération courant $(x^{(k)}, y^{(k)}, z^{(k)})$ est donné. Puis un sous problème approché, dans lequel les fonctions $f_i(x)$ sont remplacées par certaines fonctions convexes $\tilde{f_i}^k(x)$ sont générées. Le choix de ces fonctions approchées repose principalement sur les informations du gradient des f_i au point d'itération courant, mais aussi sur certains paramètres $u_j^{(k)}, l_j^{(k)}$ (les asymptotes mobiles), qui sont mis à jour à chaque itération en fonction des données du point d'itération précedente. Le sous-problème est résolu et l'unique solution optimale devient le prochain point d'itération $(x^{(k+1)}, y^{(k+1)}, z^{(k+1)})$. Ensuite, un nouveau sous-problème est généré, etc.

Le sous-problème MMA se présente comme suit :

$$\min : \tilde{f_0}^{(k)}(x) + a_0 z + \sum_{i=1}^{m}(c_i y_i + \frac{1}{2} d_i y_i^2) \tag{48}$$

$$\text{sujet à } : \tilde{f_i}^{(k)}(x) - a_i z - y_i \leq 0, i = 1, ., m \tag{49}$$

$$\alpha_j^{(k)} \leq x_j \leq \beta_j^{(k)}, j = 1, ., n \tag{50}$$

$$y_i \geq 0, i = 1, ., m \tag{51}$$

$$z \geq 0. \tag{52}$$

Dans ce sous-problème (48), les fonctions approchéees $\tilde{f_i}^{(k)}(x)$ sont choisies comme suit :

$$\tilde{f_i}^{(k)}(x) = \sum_{j=1}^{n} \Big(\frac{p_{ij}^{(k)}}{u_j^{(k)} - x_j} + \frac{q_{ij}^{(k)}}{x_j - l_j^{(k)}} \Big) + r_i^{(k)}, i = 0, ., m, \tag{53}$$

Où :

$$p_{ij}^{(k)} = (u_j^{(k)} - x_j^{(k)})^2 \bigg(1.001 \Big(\frac{\partial f_i(x^k)}{\partial x_j} \Big)^+ + 0.001 \Big(\frac{\partial f_i(x^k)}{\partial x_j} \Big)^- + \frac{10^{-5}}{x_j^{max} - x_j^{min}} \bigg), \tag{54}$$

$$q_{ij}^{(k)} = (x_j^{(k)} - l_j^{(k)})^2 \bigg(0.001 \Big(\frac{\partial f_i(x^k)}{\partial x_j} \Big)^+ + 1.001 \Big(\frac{\partial f_i(x^k)}{\partial x_j} \Big)^- + \frac{10^{-5}}{x_j^{max} - x_j^{min}} \bigg), \tag{55}$$

$$r_i^{(k)} = f_i(x^{(k)}) - \sum_{j=1}^{n} \Big(\frac{p_{ij}^{(k)}}{u_j^{(k)} - x_j^{(k)}} + \frac{q_{ij}^{(k)}}{x_j^{(k)} - l_j^{(k)}} \Big). \tag{56}$$

Où, $\Big(\frac{\partial f_i(x^{(k)})}{\partial x_j} \Big)^+$ désigne le plus grand des deux nombres, $\frac{\partial f_i(x^{(k)})}{\partial x_j}$ et 0, tandis que $\Big(\frac{\partial f_i(x^{(k)})}{\partial x_j} \Big)^-$ désigne le plus grand des deux nombres, $-\frac{\partial f_i(x^{(k)})}{\partial x_j}$ et 0. Les limites $\alpha_j^{(k)}$, $\beta_j^{(k)}$ dans (50) sont choisies comme :

$$\alpha_j^{(k)} = max\{x_j^{min}, l_j^{(k)} + 0.1(x_j^{(k)} - l_j^{(k)}), x_j^{(k)} - 0.5(x_j^{max} - x_j^{min})\}, \tag{57}$$

$$\beta_j^{(k)} = min\{x_j^{max}, u_j^{(k)} - 0.1(u_j^{(k)} - x_j^{(k)}), x_j^{(k)} + 0.5(x_j^{max} - x_j^{min})\}, \tag{58}$$

ce qui signifie que les contraintes $\alpha_j^{(k)} \leq \beta_j^{(k)}$ sont équivalentes aux trois ensembles de contraintes suivantes :

$$x_j^{min} \leq x_j \leq x_j^{max} \tag{59}$$

$$-0.9(x_j^{(k)} - l_j^{(k)}) \leq x_j - x_j^{(k)} \leq 0.9(u_j^{(k)} - x_j^{(k)}) \tag{60}$$

$$-0.5(x_j^{max} - x_j^{min}) \leq x_j - x_j^{(k)} \leq 0.5(x_j^{max} - x_j^{min}) \tag{61}$$

Les règles par défaut pour la mise à jour des asymptotes inférieures $l_j^{(k)}$ et supérieures $u_j^{(k)}$ sont les suivantes. Les deux premières itérations, quand k = 1 et k = 2,

$$l_j^{(k)} = x_j^{(k)} - 0.5(x_j^{max} - x_j^{min}) \tag{62}$$
$$u_j^{(k)} = x_j^{(k)} + 0.5(x_j^{max} - x_j^{min}) \tag{63}$$

Pour les itérations suivantes, lorsque $k \geq 3$:

$$l_j^{(k)} = x_j^{(k)} - \gamma_j^{(k)}(x_j^{k-1} - l_j^{k-1}) \tag{64}$$
$$u_j^{(k)} = x_j^{(k)} + \gamma_j^{(k)}(u_j^{k-1} - x_j^{k-1}) \tag{65}$$

Où :

$$\gamma_j^{(k)} = \begin{cases} 0.7 & \text{si } (x_j^k - x_j^{k-1})(x_j^k - x_j^{k-2}) \leq 0, \\ 1.2 & \text{si } (x_j^k - x_j^{k-1})(x_j^k - x_j^{k-2}) \geq 0, \\ 1 & \text{si } (x_j^k - x_j^{k-1})(x_j^k - x_j^{k-2}) = 0, \end{cases} \tag{66}$$

à condition que cela conduise à des valeurs qui satisfont :

$$l_j^{(k)} \leq x_j^{(k)} - 0.01(x_j^{max} - x_j^{min}) \tag{67}$$
$$l_j^{(k)} \geq x_j^{(k)} - 10(x_j^{max} - x_j^{min}) \tag{68}$$
$$u_j^{(k)} \geq x_j^{(k)} + 0.01(x_j^{max} - x_j^{min}) \tag{69}$$
$$l_j^{(k)} \leq x_j^{(k)} + 10(x_j^{max} - x_j^{min}) \tag{70}$$

Si l'une de ces limites $l_j^{(k)}, u_j^{(k)}$ est non vérifiée, la correspondante est placée à droite de l'inégalité non vérifiée. Notez que la plupart des nombres explicites dans les expressions ci-dessus ne sont que des valeurs par défaut.

1.2.7.1.2.2 Optimiseur GCMMA

L'optimiseur GCMMA est la version MMA d'une méthode globalement convergente. Pour résoudre des problèmes de type (42), GCMMA se compose d'itérations «externes» et «internes». L'indice k est utilisé pour désigner le numéro d'itération externe, et ν pour désigner le numéro d'itération interne. Dans chaque itération externe, il peut y avoir zéro, une ou plusieurs itérations internes. Les double indices (k, ν) sont utilisés pour désigner la ν ième itération interne dans la k ième itération externe. Le premier point d'itération est obtenu en choisissant d'abord $x^{(1)} \in X$, puis en choisissant $y^{(1)}$ et $z^{(1)}$ tel que $(x^{(1)}, y^{(1)}, z^{(1)})$ devient une solution admissible de (42). Une

itération externe de la méthode, partant du k ème point d'itération $(x^{(k)}, y^{(k)}, z^{(k)})$ au (k +1) ème point d'itération $(x^{(k+1)}, y^{(k+1)}, z^{(k+1)})$, peut être décrit comme suit : Étant donné $(x^{(k)}, y^{(k)}, z^{(k)})$, un sous-problème approché est généré et résolu. Dans ce sous-problème, les fonctions $f_i(x)$ sont remplacées par certaines fonctions convexes $\tilde{f}_i^{(k,0)}(x)$. La solution optimale de ce sous-problème est notée $(\hat{x}^{(k,0)}, \hat{y}^{(k,0)}, \hat{z}^{(k,0)})$. Si $\tilde{f}_i^{(k,0)}(\hat{x}^{(k,0)}) \geq f_i(\hat{x}^{(k,0)})$, pour tous i=0,..,m, le prochain point d'itération devient $(x^{(k+1)}, y^{(k+1)}, z^{(k+1)}) = (\hat{x}^{(k,0)}, \hat{y}^{(k,0)}, \hat{z}^{(k,0)})$, et l'itération externe est terminée (sans aucune itération interne nécessaire). Sinon, une itération interne est faite, ce qui signifie qu'un nouveau sous-problème est généré et résolu au point $x^{(k)}$, avec de nouvelles fonctions approchées $\tilde{f}_i^{(k,1)}(x)$, qui sont plus conservatives que $\tilde{f}_i^{(k,0)}(x)$, pour les indices i pour lesquels l'inégalité ci-dessus n'est pas vérifié. Nous entendons par là $\tilde{f}_i^{(k,1)}(x) > \tilde{f}_i^{(k,0)}(x)$, pour tout $x \in X^{(k)}$, sauf $x = x^{(k)}$ où $\tilde{f}_i^{(k,1)}(x) = \tilde{f}_i^{(k,0)}(x)$. La solution optimale de ce nouveau sous-problème est notée $(\hat{x}^{(k,1)}, \hat{y}^{(k,1)}, \hat{z}^{(k,1)})$. Si $\tilde{f}_i^{(k,1)}(\hat{x}^{(k,1)}) \geq f_i(\hat{x}^{(k,1)})$, pour tous i=0,..,m, le prochain point d'itération devient $(x^{(k+1)}, y^{(k+1)}, z^{(k+1)}) = (\hat{x}^{(k,1)}, \hat{y}^{(k,1)}, \hat{z}^{(k,1)})$ et l'itération externe est terminée (avec une itération interne nécessaire). Sinon, une autre itération interne est effectuée, ce qui signifie qu'un nouveau sous-problème est généré et résolu en $x^{(k)}$, avec de nouvelles fonctions approchées $\tilde{f}_i^{(k,2)}(x)$, etc. Ces itérations internes sont répétées jusqu'à ce que $\tilde{f}_i^{(k,\nu)}(\hat{x}^{(k,\nu)}) \geq f_i(\hat{x}^{(k,\nu)})$, pour tous i=0,.,m, ce qui arrive toujours après un nombre fini (généralement petit) d'itérations internes. Alors le prochain point d'itération devient $(x^{(k+1)}, y^{(k+1)}, z^{(k+1)}) = (\hat{x}^{(k,\nu)}, \hat{y}^{(k,\nu)}, \hat{z}^{(k,\nu)})$, et l'itération externe est terminée (avec ν itérations internes nécessaires).

Il convient de noter que dans chaque itération interne, il n'est pas nécessaire de recalculer les gradients $\nabla f_i(x^{(k)})$, puisque $x^{(k)}$ n'a pas changé. Les gradients des fonctions f_i ne sont calculés qu'une fois par itération externe. C'est une remarque importante car le calcul des gradients est généralement la partie la plus coûteuse en optimisation topologique des structures mécaniques.

Le sous-problème GCMMA se présente comme suit, pour $k \in \{1, 2, 3, ., \}$ et $\nu \in \{1, 2, ., \}$:

$$\min : \tilde{f}_0^{(k,\nu)}(x) + a_0 z + \sum_{i=1}^{m}(c_i y_i + \frac{1}{2} d_i y_i^2) \tag{71}$$

$$\text{sujet à } : \tilde{f}_i^{(k,\nu)}(x) - a_i z - y_i \leq 0, i = 1, ., m \tag{72}$$

$$\alpha_j^{(k)} \leq x_j \leq \beta_j^{(k)}, j = 1, ., n \tag{73}$$

$$y_i \geq 0, i = 1, ., m \tag{74}$$

$$z \geq 0. \tag{75}$$

Dans ce sous-problème (71), les fonctions approchées $\tilde{f}_i^{(k,\nu)}(x)$ sont choisies comme suit :

$$\tilde{f}_i^{(k,\nu)}(x) = \sum_{j=1}^{n} \Big(\frac{p_{ij}^{(k,\nu)}}{u_j^{(k)} - x_j} + \frac{q_{ij}^{(k,\nu)}}{x_j - l_j^{(k)}} \Big) + r_i^{(k,\nu)}, i = 0, ., m, \tag{76}$$

Où :

$$p_{ij}^{(k,\nu)} = (u_j^{(k)} - x_j^{(k)})^2 \left(1.001 \Big(\frac{\partial f_i(x^k)}{\partial x_j} \Big)^+ + 0.001 \Big(\frac{\partial f_i(x^k)}{\partial x_j} \Big)^- + \frac{\rho_i^{(k,\nu)}}{x_j^{max} - x_j^{min}} \right), \tag{77}$$

$$q_{ij}^{(k)} = (x_j^{(k)} - l_j^{(k)})^2 \Big(0.001 \Big(\frac{\partial f_i(x^k)}{\partial x_j} \Big)^+ + 1.001 \Big(\frac{\partial f_i(x^k)}{\partial x_j} \Big)^- + \frac{\rho_i^{(k,\nu)}}{x_j^{max} - x_j^{min}} \Big), \tag{78}$$

$$r_i^{(k,\nu)} = f_i(x^{(k)}) - \sum_{j=1}^{n} \Big(\frac{p_{ij}^{(k,\nu)}}{u_j^{(k)} - x_j^{(k)}} + \frac{q_{ij}^{(k,\nu)}}{x_j^{(k)} - l_j^{(k)}} \Big). \tag{79}$$

Si : $\rho_i^{(k,\nu+1)} > \rho_i^{(k,\nu)}$ alors $\tilde{f}_i^{(k,\nu+1)}(x)$ est une approximation plus conservative que $\tilde{f}_i^{(k,\nu)}(x)$. Entre chaque itération externe, les limites $\alpha_j^{(k)}$ et $\beta_j^{(k)}$ et les asymptotes mobiles $l_j^{(k)}$ et $u_j^{(k)}$ sont mises à jour comme dans MMA, les formules (57),(58) tiennent toujours. Les paramètres $\rho_i^{(k,\nu)}$, dans (77) et (78) sont strictement positifs et mis à jour comme suit. Dans une itération externe k donnée, les seules différences entre deux itérations internes sont les valeurs de certains de ces paramètres. Au début de chaque itération externe, lorsque $\nu = 0$, les valeurs par défaut suivantes sont utilisées :

$$\rho_i^{(k,0)} = \frac{0.1}{n} \sum_{j=1}^{n} \Big| \frac{\partial f_i(x^{(k)})}{\partial x_j} \Big| (x_j^{max} - x_j^{min}), i = 0, ., m. \tag{80}$$

Si l'un des membres de droite dans (80) $< 10^{-6}$, alors le $\rho_i^{(k,0)}$ correspondant reçoit 10^{-6}. A chaque nouvelle itération interne, la mise à jour de $\rho_i^{(k,0)}$ est basé sur la solution du sous-problème le plus récent. Notez que $\tilde{f}_i^{(k,\nu)}(x)$ peut être écrit sous la forme :

$$\tilde{f}_i^{(k,\nu)}(x) = h_i^{(k)}(x) + \rho_i^{(k,\nu)} d^{(k)}(x), \tag{81}$$

Où $h_i^{(k)}(x)$ et $d^{(k)}(x)$ ne dépend pas de $\rho_i^{(k,\nu)}$. Aprés calculs, on obtient :

$$d^{(k)}(x) = \sum_{j=1}^{n} \frac{(u_j^{(k)} - l_j^{(k)})(x_j - x_j^{(k)})^2}{(u_j^{(k)} - x_j)(x_j - l_j^{(k)})(x_j^{max} - x_j^{min})} \tag{82}$$

Supposons :

$$\delta_i^{(k,\nu)} = \frac{f_i^{(k,\nu)}(\hat{x}^{(k,\nu)}) - \tilde{f}_i^{(k,\nu)}(\hat{x}^{(k,\nu)})}{d^{(k)}(\hat{x}^{(k,\nu)})} \tag{83}$$

Alors $h_i^{(k)}(\hat{x}^{(k,\nu)}) + (\rho_i^{(k,\nu)} + \delta_i^{(k,\nu)}) d^{(k)}(\hat{x}^{(k,\nu)}) = f_i(\hat{x}^{(k,\nu)})$, ce qui montre que $\rho_i^{(k,\nu)} + \delta_i^{(k,\nu)}$ pourrait être une valeur normale de $\rho_i^{(k,\nu+1)}$. Pour obtenir une méthode globalement convergente, cette valeur normale est modifiée comme suit :

$$\rho_i^{(k,\nu+1)} = \begin{cases} \min\{1.1(\rho_i^{(k,\nu)} + \delta_i^{(k,\nu)}), 10\rho_i^{(k,\nu)}\} & \text{si } \delta_i^{(k,\nu)} > 0 \\ \rho_i^{(k,\nu)} & \text{si } \delta_i^{(k,\nu)} \leq 0 \end{cases} \tag{84}$$

Il découle des formules (76) - (79) que les fonctions $\tilde{f}_i^{(k,\nu)}$ sont toujours des approximations du premier ordre des fonctions f_i à l'itération courante, ie :

$$\tilde{f}_i^{(k,\nu)}(x^{(k)}) = f_i(x^{(k)}), \frac{\partial \tilde{f}_i^{(k,\nu)}}{\partial x_j}(x^{(k)}) = \frac{\partial f_i}{\partial x_j}(x^{(k)}) \tag{85}$$

Puisque les paramètres $\rho_i^{(k,\nu)}$ sont toujours strictement positifs, les fonctions $\tilde{f}_i^{(k,\nu)}$ sont strictement convexes. Cela implique qu'il existe toujours une solution optimale unique du sous-problème GCMMA.

On considèrera deux approches dans (Krister Svanberg, 2007) pour résoudre les sous-problèmes MMA et GCMMA, l'approche duale et l'approche point intérieur-primal-dual. Dans l'approche point intérieur-primal-dual, une suite de conditions KKT relaxées, sont résolues par la méthode de Newton.

1.2.7.2 Solveurs d'optimisation plus généraux

On considère le problème d'optimisation : {(45), (46), (47)}. En raison du fait que les formulations de problèmes plus larges permettent une représentation plus détaillée et plus précise de la réalité, des cas de plus en plus importants ont été pris en charge, facilités par les progrès réalisés en informatique. Des méthodes d'optimisation plus généraux ont été développées, comme par exemple les méthodes SQP (Sequential Quadratic Programming). De plus, de nouveaux outils mathématiques ont été développés comme la différentiation automatique des fonctions.

On décrit ci-dessous, les méthodes SQP et de points intérieurs décrites dans (A. Wächter, janvier 2002,[62]).

1.2.7.2.1 Méthodes SQP

Définitions

définition 1.1. *L'indépendance linéaire, appélée LICQ (The Linear Independence Constraint Qualification) est satisfait au point x_*, si le gradient des contraintes d'égalités et contraintes de bornes actives sont linéairement indépendants, ie si les lignes de la matrice*

$$[A(x_*)e_{j1}....e_{jK}], \tag{86}$$

avec

$$\mathcal{A}(x_*) := \{j : x_*^j = 0\} = \{j_1, .., j_K\}, \tag{87}$$

sont linéairement indépendantes.

Avec cela, nous pouvons énoncer les conditions d'optimalité nécessaires, du premier ordre, aussi appelées, conditions KKT (Karush-Kuhn-Tucker)

définition 1.2. *(Conditions KKT) Supposons que LICQ est satisfait au point x_*. Alors x_* satisfait les conditions KKT du problème (47), s'il existe des vecteurs $\lambda_* \in \mathbb{R}^m$ et $v_* \in \mathbb{R}^n$, avec $v_* \geq 0$ et $v_*^i = 0$, $i \notin \mathcal{I}$, tels que*

$$\nabla_x \mathcal{L}(x_*, \lambda_*, v_*) = g(x_*) + A(x_*)\lambda_* - v_* = 0 \tag{88}$$

$$c(x_*) = 0 \tag{89}$$

$$x_* \geq 0 \tag{90}$$

$$x_*^T v_* = 0, \tag{91}$$

où (88) est la condition dual de faisabilté, (89)-(90) est la condition primal de faisabilité, et (91) est la condition de complémentarité.

définition 1.3. *(Conditions SOS) Supposons LICQ est satisfait au x_*. Alors, x_* satisfait les conditions SOS (Second Order Sufficient) pour le problème (47), si,*

i) x_* satisfait la condition KKT du problème (47),

ii) x_* *satisfait la condition stricte de complementarité, ie*

$$v^i > 0, \forall i \in \mathcal{I}, \text{avec } x_*^i = 0,$$

iii) la Hessienne $\nabla^2_{xx}\mathcal{L}(x_*, \lambda_*, v_*)$ *du Lagrangien est définie positive sur le noyau des contraintes actives, ie si* $\mathcal{Z}_*$ *est la matrice dont les vecteurs colonnes forment une base du noyau de la matrice (86), alors*

$$\mathcal{Z}_*^T \nabla^2_{xx}\mathcal{L}(x_*, \lambda_*, v_*)\mathcal{Z}_* \tag{92}$$

est définie positive.

Beaucoup d'algorithmes couramment utilisés pour la résolution de problèmes d'optimisation : {(45), (46), (47)}, appartiennent à la classe de méthodes de programmation séquentielle quadratique (SQP). Bien qu'il y ait une grande variété de ces algorithmes dans (NLOPT), leur idée de base peut être décrite comme suit :

Supposons que (47), n'a pas de contraintes de bornes, ie dans X, $x_j^{min} = -\infty$ et $x_j^{max} = +\infty$. Alors, les conditions KKT sont :

$$g(x_*) + A(x_*)\lambda_* = 0 \tag{93}$$
$$c(x_*) = 0 \tag{94}$$

Où $g(x) = \nabla f_0(x)$, est le gradient de la fonction objectif, et la transposée $A^T(x) = \nabla c(x)$, est la transposée de la jacobienne (ie, des contraintes f_i dans (46), i=1,..,m). Par conséquent, afin de trouver un point critique x_* du problème (45), avec des multiplicateurs optimaux λ_*, on peut appliquer la méthode de Newton au système d'équations non linéaire {(93), (94)} : Étant donné une estimation initiale (x_0, λ_0) du système {(93), (94)}, nous générons une suite $\{(x_k, \lambda_k)\}$ définie comme suit :

$$(x_{k+1}, \lambda_{k+1}) = (x_k, \lambda_k) + (d_k, d_k^\lambda), \tag{95}$$

où les pas de directions (d_k, d_k^λ) sont obtenues comme la solution de la linéarisation des conditions KKT du problème {(45), (46), (47)} :

$$\begin{pmatrix} W_k & A_k \\ A_k^T & 0 \end{pmatrix} \begin{pmatrix} d_k \\ d_k^\lambda \end{pmatrix} = -\begin{pmatrix} g_k + A_k\lambda_k \\ c_k \end{pmatrix} \tag{96}$$

Où $c_k := c(x_k)$, $g_k := g(x_k)$, $A_k := A(x_k)$, et $W_k := \nabla^2_{xx}\mathcal{L}(x_k, \lambda_k)$. On sait que si (x_0, λ_0) est suffisamment proche d'une solution locale stricte satisfaisant les conditions SOS ([62]), alors le système linéaire (96) a une solution unique pour tout k, et la suite (x_k, λ_k) converge quadratiquement vers (x_*, λ_*) (Theorem 15.2.1, (A. Wächter, [62])).

L'obtention des pas (d_k, d_k^λ), du système linéaire (96) est équivalent à l'obtention du point critique d_k, solution du problème quadratique (QP) :

$$\min_{d \in \mathbb{R}^n} : g_k^T d + \frac{1}{2} d^T W_k d \tag{97}$$
$$\textbf{t.q} : A_k^T d + c_k = 0, \tag{98}$$

avec des multiplicateurs correspondants, λ_k^+, et une mise à jour :

$$d_k^{\lambda} := \lambda_k^{+} - \lambda_k \tag{99}$$

Si la matrice Hessienne W_k, du Lagrangien est définie positive sur $Ker(A_k^T)$, de l'opérateur A_k^T, par exemple dans un voisinage d'une solution locale stricte x, satisfaisant les conditions SOS, alors d_k est solution au problème (QP) de {(96),(98)}. Le problème (QP) peut être interprété comme un modèle local du problème {(45), (46), (47)} au point x_k, consistant en une approximation quadratique de la fonction objectif f_0 et une approximation linéaire des contraintes $f_i, i = 1, ..m$. Si l'on veut traiter les problèmes du type {(45), (46), (47)} avec des contraintes d'inégalités, nous pouvons inclure une approximation linéaire des contraintes d'inégalités dans le modèle local et utiliser le problème (QP) résultant :

$$\min_{d \in \mathbb{R}^n} : g_k^T d + \frac{1}{2} d^T W_k d \tag{100}$$

$$\mathbf{t.q} : A_k^T d + c_k = 0, \tag{101}$$

$$x \in X, d_i^{(k)}, i = 1, ., n, \tag{102}$$

obtenir les pas de directions $(d_{(k)}, d_{(k)}^{\lambda}, d_{(k)}^{v})$, où la suite $d_{(k)}^{v}$ sont les pas des directions pour le vecteur multiplicateur lié aux contraintes de bornes. Si x_* est une solution locale stricte satisfaisant les conditions SOS (A. Wächter,[62]), avec multiplicateurs optimaux λ_*, v_*, et si (x_0, λ_0, v_0) est suffisamment proche de (x_*, λ_*, v_*) et si la suite $\{(x_k, \lambda_k, v_k)\}$ est générée comme dans (95), alors pour une taille k suffisamment grande, le problème (QP) de {(100), (101), (102)} coïncide avec $\mathcal{A}(x_*)$ dans (Definition 1.1, Theorem 18.1, [41]). L'algorithme se comporte alors comme si toutes les bornes actives étaient incluses dans les contraintes d'égalité et les limites inactives sont ignorées. Par conséquent, les pas de directions se comportent encore une fois comme les pas de Newton pour le système (96) et une convergence locale rapide est atteinte.

Jusqu'ici, nous n'avons abordé que le comportement de convergence locale de la méthode SQP. Mais, choisir un bon point de départ $(x_0; \lambda_0; v_0)$ n'est pas une chose facile et donc des stratégies doivent être ajoutées pour garantir la convergence globale, ie convergence vers une solution (ou au moins un point critique) du problème (45), (46), (47) à partir de n'importe quel point de départ. Et justement, les algorithmes SQP se distinctent selon la technique utilisée pour atteindre cet objectif. D'autres différences consistent dans la façon dont le problème {(100), (101), (102)} est résolu et dans le choix de la matrice Hessienne W_k dans (100), qui par exemple, pourrait être une approximation quasi-newton de $\nabla^2_{xx}\mathcal{L}(x_k, \lambda_k, v_k)$.

Certaines de ces options sont réexaminées dans (2.4 Global Convergence, A. Wächter, [62])

1.2.7.2.2 Méthodes de points intérieurs

Depuis quelques années, plusieurs méthodes de points intérieurs ont été développées (voir par ex. dans, A. Wächter, [62],[47],[49],[2],[21],[16],[39],[3],[33],[60],[64],[27]), telles que les méthodes primal ou primal-dual, méthodes de recherche ou à régions de confiance, avec différentes fonctions de mérite (A. Wächter, [62], 2.4 Global Convergence), différentes stratégies de mises à jour du paramètre barrière (A. Wächter, [62], 2.7 Barrier Methods), etc. On décrit ci-dessous l'une de ces méthodes : la méthode Knitro (connu aussi sous le nom "NITRO"), de Byrd, Gilbert, Hribar, Liu, Nocedal, et Waltz dans (A. Wächter, [62], [21, 22, 23, 24]). On peut aussi voir la méthode Loqo, de Benson, Shanno, et Vanderbei dans (A. Wächter, [62],[47, 49, 46, 48, 26, 25]).

La méthode Knitro

On aborde les problèmes non-linèaires dont la formulation générale est :

$$\min_{x\in\mathbb{R}^n} : f(x) \tag{103}$$

$$\textbf{t.q} : c^{\mathcal{E}} = 0, \tag{104}$$

$$x \in X, c^{\mathcal{I}} \geq 0, \tag{105}$$

en introduisant des variables d'ajout s [18] pour les contraintes d'inégalité (105). Une suite de problèmes similaire à (2.38), 2.7 Barrier Methods, dans (A. Wächter,[62]) est résolu pour des valeurs monotones décroissantes du paramètre barrière μ (2.7 Barrier Methods, A. Wächter,[62],), qui sont ensuite, maintenus constants pour plusieurs itérations jusqu'à ce que les méthodes de mises à jour du paramètre barrière (2.7 Barrier Methods, A. Wächter, [62]) soit suffisamment bien résolu. La convergence globale est assurée par une approche à régions de confiance (vr. Section 2.4,A. Wächter, [62]) en utilisant la pénalité-l_2 de la fonction exacte ((2.14)[62], ie "f(x)" est remplacée par "$\varphi_\mu(x)$"). Les pas sont générés par l'algorithme SQP à régions de confiance de Byrd-Omojokun ([44]) dans, (A. Wächter,[62]) appliqué au problème barrière : à une itération (x_k, s_k) avec $s_k > 0$, un pas d'essai (d_k, d_k^s) est calculé comme solution approchée du sous-problème :

$$\min_{d,d^s\in\mathbb{R}^n} : g_k^T d + \frac{1}{2} d^T W_k d - \mu e^T S_k^{-1} d^s + \frac{1}{2}(d^s)^T \Sigma_k d^s \tag{106}$$

$$\textbf{t.q} : (A_k^{\mathcal{E}})^T d + c^{\mathcal{I}}(x_k) - s_k = r^{\mathcal{I}}, \tag{107}$$

$$(A_k^{\mathcal{I}})^T d - c^{\mathcal{E}}(x_k) = r^{\mathcal{E}} \tag{108}$$

$$d^s \geq \tau s_k \tag{109}$$

$$||(d, S_k^{-1} d^s)||_2 \leq \Delta_k, \tag{110}$$

Où $e = (1, ., 1)^T$, $A_k^{\mathcal{E}} := \nabla c^{\mathcal{E}}(x_k)$, $A_k^{\mathcal{I}} := \nabla c^{\mathcal{I}}(x_k)$, $S_k := diag(s_k)$, et $\Sigma_k := S_k^{-1} V_k$, avec $V_k := diag(v_k)$, où v_k est une estimation positive des multiplicateurs de Lagrange associée à (105). W_k est la matrice Hessienne (exacte) du Lagrangien associée à (105). L'inégalité (109) pour une constante $\tau \in (0, 1)$ proche de 1 permet aussi qu'à la prochaine itération, on ait $s_{k+1} > 0$. À noter que, contrairement aux méthodes SQP décrites précédemment, la linéarisation des contraintes (107) et (108) ne doivent pas être strictement satisfait par les pas d'essai (ie, nous pourrions avoir $r^{\mathcal{E}}, r^{\mathcal{I}} \neq 0$), ce qui permet de traiter le cas où le problème (QP) est inadmissible, ie "stricte", avec $r^{\mathcal{E}}, r^{\mathcal{I}} = 0$

La solution approchée au sous problème {(106),(107), (108), (109) , (110)} est obtenu par décomposition générale des pas d'essai (d_k, d_k^s) en une composante normale (q_k, q_k^s) et une composante tangentielle (p_k, p_k^s) (anolgue à (2.20) dans A. Wächter, [62]). La composante normale est une solution (approchée) au problème normal :

$$\min_{q,q^s\in\mathbb{R}^n} : ||(A_k^{\mathcal{E}})^T q + c^{\mathcal{E}(x_k)}||_2^2 + ||(A_k^{\mathcal{I}})^T q - q^s c^{\mathcal{I}(x_k)} + s_k||_2^2 \tag{111}$$

$$\textbf{t.q} : q^s \geq -\tau s_k/2, \tag{112}$$

$$||(q, S_k^{-1} q^s)||_2 \leq 0.8\Delta_k, , \tag{113}$$

qui est obtenu en utilisant la méthode Dogleg [19]. Le pas normal résultant est une combinaison linéaire de la plus profonde descente $(q_k^{sd}, q_k^{s,sd})$ et la solution des moindres carrés $(q_k^{ls}, q_k^{s,ls})$ au problème (QP) de (111) sans les contraintes {(112), (113)}. Un point important ici est que les deux composantes de la méthode Dogleg sont obtenus après la mise à l'échelle par S_k^{-1} de l'espace de la variable d'ajout (comme dans (66)). Par exemple $(q_k^{ls}, q_k^{s,ls})$ est calculé en résolvant en premier le système augmenté (mis à l'échelle) :

18. variable qui est ajoutée à une contrainte d'inégalité pour la transformer en une égalité
19. Méthode de recherche dans la région de confiance

$$\begin{pmatrix} I & 0 & A_k^{\mathcal{E}} & A_k^{\mathcal{I}} \\ 0 & I & 0 & -S_k \\ (A_k^{\mathcal{E}})^T & 0 & 0 & 0 \\ (A_k^{\mathcal{I}})^T & -S_k & 0 & 0 \end{pmatrix} \begin{pmatrix} y_k \\ y_k^s \\ z_k \\ z_k^s \end{pmatrix} = - \begin{pmatrix} 0 \\ 0 \\ c^{\mathcal{E}}(x_k) \\ c^{\mathcal{E}}(x_k) - s_k \end{pmatrix} \tag{114}$$

et puis on définit :

$$\begin{pmatrix} q_k^{ls} \\ q_k^{s,ls} \end{pmatrix} := \begin{pmatrix} A_k^{\mathcal{E}} & A_k^{\mathcal{I}} \\ 0 & S_k^2 \end{pmatrix} \begin{pmatrix} z_k \\ z_k^s \end{pmatrix} \tag{115}$$

La composante horizontale est solution approchée d'un sous-problème réduit obtenue par une méthode PCG (similaire à (2.24), sans les contraintes d'inégalités mais des contraintes similaire à (2.44d) et (2.44e), dans, (A. Wächter,[62])). Les pas de la méthode PCG sont calculés à l'aide des solutions du système augmenté (mis à l'échelle) de (114) avec différents seconds membres. Si la projetée de la matrice Hessienne (106) dans l'espace des noyaux des contraintes (107), (109) n'est pas définie positive, la méthode PCG peut rencontrer des directions de courbure négative, suivies jusqu'à la limite de la région de confiance. Les estimations λ_k et v_k associées respectivement à (104) et (105) sont calculés comme des multiplicateurs de la méthode des moindres carrés, éventuellement corrigés afin d'avoir $v_k > 0$.

La convergence globale de la méthode Knitro a été démontrée dans (A. Wächter, [62], [47]) et la convergence locale, superlinéaire à été démontrée dans (A. Wächter,[62], [46]) en utilisant un taux de diminution superlinéaire pour le paramètre de barrière μ. Cependant, la mise en œuvre actuelle de Knitro ne diminue que d'un facteur 0.2, conduisant donc uniquement à un taux de convergence local linéaire.

1.2.10 Benchmarking

Afin de comparer l'éfficacité de différents algorithmes d'optimisation, nous utiliserons des courbes de performances par le biais de "data profiles" et les profils de performance ([36]), comme outil d'analyse des performances des solveurs d'optimisation, appliqués à divers problèmes d'optimisation topologique de structures mécaniques sous contraintes dans le cadre de l'élasticité linéaire. Une bibliothèque complète, représentative d'une classe de problèmes de compliance minimale et de volume minimal d'une structure élastique pour différentes tailles E.F est devéloppée pour cette analyse comparative. On utilise une courbe de performances, conjointement avec un test de convergence qui mesure la diminution de la valeur d'une fonction donnée, afin d'analyser les performances de deux grandes classes d'optimisation : les méthodes de programmation séquentielle convexe (MMA, GCMMA) et l'approche heuristique, appelée OC.

Nos résultats fournissent des estimations de la différence de performance entre les solveurs classiques : OC, MMA, GCMMA. On considérera dans le présent travail un test de convergence qui mesure la diminution de la valeur d'une fonction donnée :

$$f(x_0) - f(x) \geq (1 - \tau)(f(x_0) - f_L), \tag{116}$$

où $\tau > 0$ est la tolérance, x_0 est le point de départ du problème d'optimisation, et f_L est calculé pour chaque problème comme la plus petite valeur de f obtenue quelque soit le solveur utilisé, pour limite donné, μ_f d'évaluations de la fonction. Ce test de convergence est invariant à la transformation affine $f \to \alpha f + \beta$ ($\alpha > 0$) et mesure la réduction, $f(x_0) - f(x)$, d'une fonction atteinte par x, par rapport à la meilleure réduction possible $f(x_0) - f_L$.

Nous utilisons des profils de performance et "Data profiles" (Jorge J. Moré et Stefan M. Wild, [18]), conjointement au test de convergence (116), pour évaluer les performances des solveurs. Au lieu d'utiliser une valeur fixe de τ, nous utilisons $\tau = 10^{-k}$, avec $k \in \{1, 3, 5, 7\}$ de sorte que l'on puisse évaluer les performances d'un solveur pour différents niveaux de précision. Ces profils de performance sont utiles aux utilisateurs qui doivent choisir un solveur qui offre une réduction donnée de la valeur de la fonction dans une limite donné, μ_f d'évaluations.

Les profils de performance ont été conçus pour comparer les solveurs et ainsi utilisent un ratio de performance au lieu du nombre d'évaluations de fonctions nécessaires pour résoudre un problème. En conséquence, les profils de performance ne fournissent pas le pourcentage de problèmes pouvant être résolus (pour une tolérance donnée τ) dans une limite donné μ_f d'évaluations de la fonction. Cette information est essentielle lorsqu'on est confronté à des problèmes d'optimisation coûteux.

Dans le présent travail, nous utiliserons aussi le "Data profiles" pour étudier le comportement à court terme des optimiseurs et qui fournit justement l'évolution en fonction d'une limite donné, μ_f d'évaluations de la fonction. Le choix des solveurs a été guidé principalement par une volonté d'examiner les performances de solveurs d'optimisation classiques (OC, MMA, GCMMA).

1.2.10.1 Analyse comparative des solveurs classiques et généraux

Les profils de performance, introduits par Dolan et Moré (Jorge J. Moré and Stefan M. Wild, april 2008, [18]), se sont révélés être un outil important pour l'analyse comparative des solveurs d'optimisation. Dolan et Moré, définissent l'analyse comparative comme la donnée d'un ensemble $\mathcal{P}$ de problèmes, d'un ensemble $\mathcal{S}$ de solveurs, et un test de convergence $\mathcal{T}$. Une fois que ces composantes d'évaluations sont définies, des profils de performance peuvent être utilisés pour comparer les performances des solveurs. Nous proposons d'abord un test de convergence pour les solveurs OC, MMA, GCMMA, et ensuite on examine la pertinence des profils de performance quand ces solveurs sont appliqués à divers problèmes d'optimisation topologique de structures mécaniques sous contraintes dans le cadre de l'élasticité linéaire. Une bibliothèque complète, représentative d'une classe de problèmes de compliance minimale et de volume minimal d'une structure élastique pour différentes tailles E.F est devéloppée pour cette analyse comparative.

1.2.10.2 Profils de performance

Les profils de performance sont définis en termes de mesure de performance $t_{p,s} > 0$ obtenu pour chaque problème $p \in \mathcal{P}$ et solveur $s \in \mathcal{S}$. Par exemple, cette mesure pourrait être basée sur le temps de calcul ou le nombre μ_f d'évaluations de fonction, nécessaires au solveur $s \in \mathcal{S}$ pour résoudre le problème $p \in \mathcal{P}$, ie satisfaire le test de convergence. Plus la valeur $t_{p,s}$ est grande, plus le solveur $s \in \mathcal{S}$ est mauvais. Pour tout couple (p,s) de problème $p \in \mathcal{P}$ et de solveur $s \in \mathcal{S}$, le ratio de performance est défini comme suit :

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in \mathcal{S}\}} \tag{117}$$

À Noter que, le meilleur solveur $s \in \mathcal{S}$ pour un problème $p \in \mathcal{P}$, donné, atteint la limite inférieure $r_{p,s} = 1$. Par convention $r_{p,s} = \infty$ est utilisé lorsqu'un solveur s ne réussit pas à satisfaire le test de convergence (pour une tolérance donné τ) pour le problème donné, $p \in \mathcal{P}$, ie ne réussit pas à résoudre le problème $p \in \mathcal{P}$

Le profil de performance d'un solveur $s \in \mathcal{S}$ est défini comme la fraction de problèmes résolus par $s \in \mathcal{S}$, quand le ratio de performance est au plus α :

$$\rho_s(\alpha) = \frac{1}{|\mathcal{P}|} card\{p \in \mathcal{P} : r_{p,s} \leq \alpha\} \tag{118}$$

Où |P| dénote le cardinal de $\mathcal{P}$. Ainsi, un profil de performance est la distribution de probabilité pour le ratio $r_{p,s}$. Les profils de performance cherchent à capturer la performance d'un solveur $s \in \mathcal{S}$ par rapport aux autres solveurs dans $\mathcal{S}$ quand appliqués à divers problèmes dans $\mathcal{P}$. Notez en particulier que $\rho_s(1)$ est la fraction de problèmes pour lesquels le solveur $s \in \mathcal{S}$ arrive en tête, (ie la performance de solveur $s \in \mathcal{S}$, pour le problème $p \in \mathcal{P}$ est la meilleure) et pour α suffisamment

grand, $\rho_s(\alpha)$ est la fraction de problèmes résolus par $s \in \mathcal{S}$. En général, $\rho_s(\alpha)$ est la fraction de problèmes résolus par $s \in \mathcal{S}$ quand le ratio de performance $r_{p,s}$ est plus α. Il est donc préférable quand un solveur $s \in \mathcal{S}$ a une valeur élevée $\rho_s(\alpha)$.

L'analyse comparative de solveur d'ordre 1 [20] comme OC, MMA, GCMMA est assez simple une fois le test de convergence choisi. Pour une tolérance donnéé, $\tau > 0$ et une norme $||\ ||$ donné. Ce test de convergence (116), est complété par un temps limite de calcul ou nombre maximum μ_f d'évaluations de la fonction. L'analyse comparative des solveurs d'optimisation d'ordre 1 est généralement fait avec un choix fixe de tolérance τ, qui donne des solutions assez précises aux problèmes évalués. L'hypothèse sous-jacente est que les performances des solveurs ne changeront pas de manière significative avec les autres choix de tolérance. Dans ce présent travail, on s'intéressera aussi bien aux solutions avec faible précision et haute précision. En pratique, lorsque l'évaluation de f est coûteuse, une solution peu précise est ce qui peut être obtenu selon le budget de calcul μ_f, donné. De plus l'exactitude des données peut ne justifier qu'une solution peu précise. C'est pourquoi, on considèrera pour l'analyse des solveurs, un test de convergence qui ne dépend pas du gradient.

Nous proposons d'utiliser le test de convergence

$$f(x) \leq f_L + \tau(f(x_0) - f_L), \tag{119}$$

où $\tau > 0$ est la tolérance, x_0 est le point de départ du problème, et f_L est calculé pour chaque problème $p \in \mathcal{P}$, comme la plus petite valeur de f obtenue, quelque soit le solveur utilisé, pour une limite donné, μ_f d'évalutions de la fonction f. Le test de convergence (119) peut également être écrit comme

$$f(x_0) - f(x) \geq (1 - \tau)(f(x_0) - f_L), \tag{120}$$

et cela montre que (119) exige que la réduction $f(x_0) - f(x)$ atteinte par x soit au moins $1 - \tau$ fois la meilleure réduction possible $f(x_0) - f_L$. Le test de convergence (120) a été utilisé par Elster et Neumaier (Jorge J. Moré et Stefan M. Wild, avril 2008, [20]), avec f_L fixé à une valeur approchée (précise) du minimum global de f. Ce test a également été utilisé par Marazzi et Nocedal (Jorge J. Moré et Stefan M. Wild, avril 2008, [34]) mais f_L fixé à une valeur approchée du minimum local de f. Fixer f_L à une valuer approchée du minimum de f, n'est pas approprié lorsque l'évaluation de f est coûteuse car il se peut qu'aucun solveur ne soit en mesure de satisfaire (120) selon le budget de calcul μ_f donné. D'autre part, si f_L est la plus petite valeur de f obtenue quelque soit le solveur $s \in \mathcal{S}$ utilisé, alors au moins un des solveurs satisfera (119) quelque soit la tolérance donnéé, $\tau \geq 0$.

L'avantage de (119) est qu'il est invariant à la transformation affine $f \to \alpha f + \beta$, où $\alpha > 0$. Par conséquent, on peut supposer, par exemple, que $f_L = 0$ et $f(x_0) = 1$. Il n'a pas de perte de généralité dans cette hypothèse car les solveurs d'optimisation dans le présent travail sont invariants à la transformation affine $f \to \alpha f + \beta$. En effet, les algorithmes pour les problèmes d'optimisation basés sur les gradients sont invariants par transformation affine. La tolérance $\tau \in [0, 1]$ dans (119) représente le pourcentage de diminution par rapport à la valeur de départ $f(x_0)$. Une valeur $\tau = 0.1$ peut représenter une diminution modeste, une réduction représentant 90% du total possible. Lorsque τ diminue, la précision de f(x) comme valeur approchée de f_L augmente ; la précision de x comme valeur approchée d'un minimum x^* de f dépend de la croissance de f dans un voisinage de x^*.

1.2.10.3 Data profils : "Profils de données"

Nous pouvons utiliser des profils de performance conjointement au test de convergence (119) pour évaluer les solveurs d'optimisation, appliqués à des problèmes avec des évaluations de fonction coûteuses. Dans ce cas, la mesure de performance $t_{p,s}$ est le nombre d'évaluations de la fonction (ie, fonction associée au problème $p \in \mathcal{P}$) par le solveur $s \in \mathcal{S}$. Les profils de performance fournissent une vue précise de la performance relative aux solveurs dans une limite μ_f, donné d'évaluations

20. méthodes d'optimisation avec gradient

de la fonction. Les profils de performance ne fournissent toutefois pas d'informations suffisantes lorsqu'on est confronté à un problème d'optimisation coûteux.

Nous nous intéressons au pourcentage de problèmes pouvant être résolus (pour une tolérance donnéée τ) avec un nombre κ d'évaluations de la fonction. On peut obtenir cette information en laissant $t_{p,s}$ égal à la valeur limite μ_f donné, d'évaluations de la fonction, nécessaires pour satisfaire le test de convergence (119), pour une tolérance donnée τ. Par conséquent :

$$d_s(\kappa) = \frac{1}{|\mathcal{P}|} card\{p \in \mathcal{P} : t_{p,s} \leq \kappa\}$$

est le pourcentage de problèmes pouvant être résolus, quand le nombre d'évalutions de la fonction est au plus κ. Comme d'habitude, il y a une limite μ_f d'évaluations totales de la fonction, et $t_{p,s} = \infty$ si le test de convergence (119) n'est pas satisfait après μ_f évaluations de la fonction.

Griffin et Kolda (Jorge J.Moré et Stefan M. Wild, avril 2008, [24]) étaient également intéressés par la performance en termes de nombre d'évaluations de fonctions et utilise une courbe de performance basée sur le nombre total de problèmes résolus par un solveur $s \in \mathcal{S}$ en un nombre κ, d'évaluations de la fonction, nécessaires pour satisfaire le test de convergence (119), pour une tolérance donnée τ. Leur principale préoccupation était d'évaluer la performance de leur algorithme par rapport à la pénalité $d_s(\kappa)$.

Cette définition de d_s est indépendante du nombre de variables du problème $p \in \mathcal{P}$. Cela n'est pas réaliste car, le nombre d'évaluations de la fonction, nécessaires pour satisfaire à un test de convergence donné est susceptible de croître parallèlement au nombre de variables. On definit donc le "data profile" pour un solveur $s \in \mathcal{S}$ comme suit :

$$d_s(\alpha) = \frac{1}{|\mathcal{P}|} card\{p \in \mathcal{P} : \frac{t_{p,s}}{n_p + 1} \leq \alpha\}, \tag{121}$$

où n_p est le nombre de variables du problème $p \in \mathcal{P}$. On appelle "data profile" le graphe de (121).

Avec cette mise à l'échelle, l'unité de coût est $n_p + 1$ évaluations de la fonction. Cette unité peut être facilement traduite en évaluations de fonction. Un autre avantage de cette unité, est que $d_s(\alpha)$ peut alors être interprété comme le pourcentage de problèmes pouvant être résolus, quand le quotient de $\frac{t_{p,s}}{n_p+1}$ est au plus α (pour un solveur avec gradient).

Profils de performances (118) et "Data profils" (121) sont des fonctions de distribution et donc sont non négatives, continues à droite, monotones croissantes à valeurs dans [0,1]. Cependant, les profils de performance comparent différents solveurs, tandis que "Data profils" affiche les données brutes. En particulier, les Profils de performances ne fournissent pas le nombre μ_f d'évaluations de la fonction, nécessaires pour résoudre un problème p, $\forall p \in \mathcal{P}$. Notez également que "Data profils" pour un solveur $s \in \mathcal{S}$ donné ne dépend pas des autres solveurs ; ce n'est pas le cas pour les profils de performance. Le "Data profiles" sont utiles si pour un couple $(p, s) \in \mathcal{P} \times \mathcal{S}$, on dispose d'un budget μ_f d'évaluations de la fonction f, qui est susceptible d'atteindre une réduction donnée de la valeur de la fonction. On exprime donc le budget μ_f en termes de κ [21] et examine les valeurs de d_s, pour tout $s \in \mathcal{S}$.

On illustre ci-dessous les différences entre les profils de performance et le "Data profiles" avec un cas impliquant deux solveurs. Supposons que le solveur s_1 nécessite $\kappa = k_1$ évaluations de la fonction, pour résoudre chacun des n_1 premiers problèmes, mais ne parvient pas à résoudre n_2 problèmes restants. De même, supposons que le solveur s_2 ne parvient pas à résoudre les n_1 premiers problèmes mais résout chacun des n_2 problèmes restants avec $\kappa = k_2$ évaluations de la fonction. Enfin, supposons que $n_1 < n_2$, et $k_1 < k_2$. Dans ce cas,

$$\rho_1(\alpha) = \frac{n_1}{n_1 + n_2},$$

$$\rho_2(\alpha) = \frac{n_2}{n_1 + n_2},$$

21. c'est la complexité des méthodes d'optimisations avec gradient

pour $\alpha \geq 1$, si le nombre maximal d'évaluations, $\mu_f \geq k_2$. Par conséquent $n_1 < n_2$ implique que $\rho_1 < \rho_2$, et donc le solveur s_2 est préférable. Cela est justifiable, car s_2 résout davantage de problèmes pour tous les ratios de performance r_{p,s_i}, i=1,2. D'autre part,

$$d_1(\alpha) = \begin{cases} 0, & \alpha \in [0, k_1) \\ \frac{n_1}{n_1+n_2}, & \alpha \in [k_1, \infty) \end{cases}$$

$$d_2(\alpha) = \begin{cases} 0, & \alpha \in [0, k_2) \\ \frac{n_2}{n_1+n_2}, & \alpha \in [k_2, \infty) \end{cases}$$

En particulier, $0 = d_2(k) < d_1(k)$ pour tout budget k, où $k \in [k_1, k_2)$, et donc le solveur s_1 est préférable sous ces contraintes budgétaires, car s_2 n'est pas capable de résoudre les problèmes avec moins de k_2 évaluations. Un autre lien à souligner entre les profils de performance et le "Data profiles" est la valeur limite de $\rho_s(\alpha)$, lorsque $\alpha \to \infty$. Ainsi

$$d_s(\tilde{\kappa}) = \lim_{\alpha \to \infty} \rho_s(\alpha), \tag{122}$$

où $\tilde{\kappa}$ est exprime le nombre d'évalutions de la fonction (associéé au problème donné $p \in \mathcal{P}$) divisé par la taille du problème. Ainsi, (122) mesure la fiabilité du solveurs s (pour une tolérance donnée τ), pour une limite donné μ_f, d'évaluations.

1.2.11 Exemple de benchmarking en 2d

Un benchmarking des solveurs classique et généraux d'optimisation topologique des structures mécaniques en 2d, ont été réalisés en 2000 par Susana Rojas-Labanda et Mathias Stolpe. Par le biais des profils de performance, nous présentons ci-aprés les résultats de leur analyse comparative.

Différents solveurs d'optimisation, y compris OC, MMA, GCMMA, méthodes de points intérieurs dans IPOPT et FMINCON, et des méthodes SQP dans SNOPT, sont analysé quand appliqués aux problèmes de compliance minimale et volume minimal en utilisant les profils de performance. Chaque fois que cela est possible, les méthodes sont appliquées à la fois au problème à la formulation Nested et SAND.

Les profils de performance concluent que les solveurs généraux dans (IPOPT, FMINCON, et SNOPT) sont aussi efficaces et fiables que les solveurs classiques (OC, MMA, GCMMA). De plus, l'utilisation des matrices Hessiennes exactes dans la formulation SAND, produisent généralement des variables de conception avec de meilleures valeurs de fonction objectif. Cependant l'analyse des solveurs, appliqués aux problèmes à la formulation SAND consomme plus de temps de calcul que de résoudre des problèmes à la formulation Nested.

Une seule approche pour paramétrer la topologie (M.E.F) et un schéma d'interpolation du matériau (méthode SIMP) avec un filtre de densité. Dans cet ensemble de tests, quelques exemples de tests 2D couramment utilisés dans la littérature sont rassemblés : classe de problèmes de compliance minimale et volume minimal

Les domaines de conception, les conditions aux limites et les chargement externes sont tirés de la littérature telle que (Susana Rojas-Labanda and Mathias Stolpe, 2015, [55],[6],[12],[4],[13], [8], [61], [54],[17], et [52]). L'ensemble de tests utilisés pour cette analyse comparative provient de (CUTE [11] et COPS [10], Rojas-Labanda and Mathias Stolpe, 2015), bibliothéques de problèmes non linéaires avec contraintes d'égalités et d'inégalités. Les profils de performance ont également été utilisés dans ([19], Rojas-Labanda and Mathias Stolpe, 2015) où LANCELOT ([15], Rojas-Labanda and Mathias Stolpe, 2015), MINOS ([37], Rojas-Labanda and Mathias Stolpe, 2015), SNOPT et LOQO sont comparés quand appliqués aux problèmes dans COPS. Enfin, cet analyse se concentre sur les problèmes de taille moyenne car implementés dans MATLAB.

Les solveurs OC, MMA, GCMMA, IPOPT, SNOPT, et FMINCON ont différentes implémentations des conditions d'optimalité ou KKT (différentes mises à l'échelle et normes). Néanmoins, à la fin

du processus d'optimisation, on mesure, le résidu KKT. Les solveurs chercheront à obtenir une conception jusqu'à ce que l'erreur de faisabilité et les conditions d'optimalité soient inférieures à certaines tolérances (conditions KKT ω et primale μ). La valeur de ω dans cette analyse est définie différemment pour les méthodes d'ordre 1 et d'ordre 2, respectivement.

IPOPT, FMINCON, et SNOPT sont généralement capables de satisfaire les conditions de KKT avec une très petite tolérance, tandis que les méthodes d'ordre 1 comme MMA, et GCMMA exigent généralement des tolérances plus grandes. Ceci est mis en évidence dans la figure 21

Solver	Parameter	Description	Value
MMA, GCMMA	`kkt tol`	Euclidean norm of the KKT error	$1e-4$
IPOPT	`tol`	Tolerance of the NLP error	$1e-6$
SNOPT	`Major optimality tolerance`	Final accuracy of the dual variables	$1e-6$
FMINCON	`TolFun`	Tolerance on the function value	$1e-6$
OC	`change`	Difference in the design variable	$1e-4$
OC,MMA, GCMMA	`feas tol`	Euclidean norm of the feasibility error	$1e-8$
IPOPT	`constr viol tol`	Tolerance of the constraint violation	$1e-8$
SNOPT	`Major feasibility tolerance`	Tolerance of the nonlinear constraint violation	$1e-8$
FMINCON	`TolCon`	Tolerance of the constraint violation	$1e-8$

Solver	Parameter	Description	Value
OC	`loop`	Maximum number of iterations	1,000
MMA, GCMMA	`max outer itn`	Maximum number of iterations	1,000
IPOPT	`max iter`	Maximum number of iterations	1,000
SNOPT	`Major iterations limit`	Maximum number of iterations	1,000
FMINCON	`MaxIter`	Maximum number of iterations	1,000
OC,MMA, GCMMA	`max assemblies`	Maximum number of stiffness matrix assemblies	10,000
IPOPT	`max cpu time`	Maximum CPU time	48h
FMINCON	`MaxFunEvals`	Maximum number of function evaluations	10,000

FIGURE 21 – Parameter names, description and values of the convergence criteria

1.2.11.1 Point de départ des problèmes

Pour OC, MMA, GCMMA, et SNOPT, le point de départ $x_0 = Ve$, où $e = (1, ., 1)^T$. Le vecteur de déplacement U est intialisé à $U_0 = 0$ pour les problèmes de compliance minimale. Cependant le point départ dans IPOPT et FMINCON sont toujours initialisés entre les limites inférieure et supérieure des variables de densité, ie $x_0 = 0.5e$ et $U_0 = 0$. D'autres points de départ pourraient être un inconvénient en raison de l'utilisation d'algorithmes de points intérieurs. Pour le volume minimal, le point de départ est initialisé à $x_0 = 0.5e$ et $U_0 = 0$

1.2.11.2 Implémentation de la limite supérieure de la compliance

Pour la classe de volume minimal, la contrainte supérieure sur la compliance est donnée par :

$$C = k(F^T(K^{-1}(x_0)F),$$

pour une constante $k \geq 1$ définie par l'utilisateur.

1.2.11.3 Initialisation des Paramètres dans (IPOPT, FMINCON, SNOPT)

En général, il est souhaitable d'ajuster le moins possible les paramètres du solveur. Toutefois, les performances des solveurs, appliqués à des problèmes d'optimisation topologique peuvent être améliorées si certains paramètres par défaut sont modifiés. La mise à l'échelle automatique du problème est désactivée pour les solveurs dans IPOPT et FMINCON. Cependant, en fonction du problème et sa formulation, SNOPT fonctionne mieux avec différentes valeurs d'option de mise à l'échelle. En pratique, les solveurs dans SNOPT, quand appliqués aux problèmes à la formulation SAND fonctionnent mieux en l'absence de mise à l'échelle automatique. La formulation Nested fonctionne avec une certaine mise à l'échelle et la valeur par défaut est donc utilisée dans cette situation.

De plus, la stratégie de mise à jour du paramètre "barrière" comme suggérée dans ([42], Susana Rojas-Labanda et Mathias Stolpe, 2015) est utilisée dans IPOPT, parce qu'il nécessite moins d'itérations que la stratégie de mise à jour monotone et que les conceptions obtenues sont plus précises. Afin d'améliorer les performances de IPOPT, certaines options sont activées, tels que "fullstep size" pour les multiplicateurs de contraintes et l'utilisation de l'estimation des moindres carrés pour calculer les multiplicateurs de contraintes. FMINCON utilise un algorithme de point intérieur. La méthode du gradient conjugué est choisie pour déterminer le pas d'itération. Enfin, dans SNOPT, la limite "super basics", la limite d'itération, la précision de la fonction, la tolérance de la recherche linéaire et la limite de pas sont modifiées pour pouvoir résoudre des problèmes de grande taille.

Lorsque l'approche Nested est utilisée, IPOPT approche la Hessienne en utilisant une approche BFGS à mémoire limitée. La raison principale de l'utilisation de BFGS est le coût de calcul élevé de la Hessienne exacte. FMINCON a une option dans laquelle une multiplication matrice-vecteur peut être définie. En utilisant cette fonctionnalité, le temps de calculs est beaucoup plus court, et FMINCON peut donc utiliser la Hessienne exacte.

Ceci est mis en évidence dans la figure 22, 23, 24

Parameter	New value	Default	Description
`Algorithm`	interior point	trust region reflective	Determine the optimization algorithm
`Subproblem Algorithm`	cg	ldl factorization	Determines how the iteration step is calculated

FIGURE 23 – Paramètres ajustés dans FMINCON

Parameter	New value	Default	Description
`scale option`	0(SAND)	2	Scaling of the problem (2 LP, 0 no scale)
`Iteration limit`	1e6	1e4	Maximum number of minor iterations allowed in QP subproblem
`Major step limit`	10	2	Limits the change in x during the line-search
`Linesearch tolerance`	.99999	.9	Accuracy which a steplength will be located along the search direction
`New superbasics limit`	1e4	99	Early termination of QP subproblem if the number of free variables has increased since the first feasibility iteration
`Function precision`	1e-4	3.7e-11	Relative function precision

FIGURE 24 – Paramètres ajustés dans SNOPT

Parameter	New value	Default	Description
`mu strategy`	adaptive	monotone	Update strategy for barrier parameter
`limited memory max history`	25	6	Maximum size of history in BFGS
`nlp scaling method`	none	gradient based	Technique for scaling the problem
`alpha for y`	full	primal	Method to determine the step size of constraint multipliers (full step size 1)
`recalc y`	yes	no	Tells the algorithm to recalculate the multipliers as least square estimates

FIGURE 22 – Paramètres ajustés dans IPOPT

1.2.11.4 Mise à l'échelle du problème

Concernant les paramètres du module de Young, la bibliothèque de problèmes pour cette analyse comparative utilise des petits écarts de valeurs entre le matériau solide (E_1) et le vide (E_v). La raison principale en est que la conception optimale est similaire aux résultats obtenus pour les écarts importants, mais que le temps et les itérations nécessaires pour satisfaire le critère d'arrêt sont considérablement réduits. Cet écart est fixé à $E_1/E_v = 10^3$. Ceci est mis en évidence par la figure 25

Et enfin le coefficent de Poisson $\nu = 0.3$ comme dans ([4],Rojas-Labanda and Mathias Stolpe, 2015). Le comportement numérique des solveurs (sauf OC) est meilleur si la contrainte d'inégalité est mise à l'échelle par un facteur de $\frac{1}{\sqrt{n}}$

1.2.11.5 Ensemble de problèmes test

Certains des ensembles tests les plus couramment cités dans la littérature pour les méthodes d'optimisation linéaire et non-linéaire sont (CUTE [11], CUTEr [22], MIPLIB [29], COPS [10], Rojas-Labanda and Mathias Stolpe,2015) et Vanderbei parmi d'autres (utilisés dans [9],[19] et [63], Rojas-Labanda et Mathias Stolpe,2015). Cependant, en optimisation topologique il n'y a pas de grand ensemble de tests de prédilection. Trois types de domaines de conception différents et chargement sont considérés (voir la figure 4) pour la compliance minimale et volume minimal. Ces exemples sont considérés dans ([55], [6], [12], [4], [13], et [8], Susana Rojas-Labanda, Mathias Stolpe, 2015).

Pour chaque domaine, nous considérons différents rapports de longueur. De plus, pour chaque longueur, des tailles différentes de discrétisation, N_l, sont définies. Où N_l est le nombre d'éléments dans le maillage par unité de longueur. On considère un ensemble de tests avec un nombre d'éléments relativement faible pour pouvoir tester tous les solveurs. Enfin, pour chaque domaine de conception dans l'ensemble tests, 5 limites supérieures de fractions volumique différentes sont considérées pour la compliance minimale. La fraction volumique est choisie parmi les valeurs 0.1, 0.2, 0.3, 0.4, et 0.5. Une brève description de l'ensemble tests est mis en évidence par la figure 26, 27, 28. Où L_x et L_y désignent le rapport de longueur dans les directions x et y, respectivement. N_l désigne le nombre d'éléments ou discrétisation par unité de longueur, n le nombre d'éléments et d le nombre de degrés de liberté.

Solver	Problem	E_1	E_0
IPOPT/SNOPT	Minimum compliance	$1e3$	$1e0$
FMINCON	Minimum compliance	$1e2$	$1e-1$
IPOPT/SNOPT	Minimum volume	$1e4$	$1e1$
FMINCON	Minimum volume	$1e1$	$1e-2$
MMA/GCMMA	Minimum compliance/volume	$1e1$	$1e-2$
OC	Minimum compliance	$1e0$	$1e-3$
All	Mechanism design	$1e0$	$1e-2$

FIGURE 25 – Paramètres de mise à l'échelle du problème

Domain	L_x	L_y	N_l	n	d
Michell	1	1	20	400	882
			40	1600	3362
			60	3600	7442
			80	6400	13122
			100	10000	20402
	2	1	20	800	1722
			40	3200	6642
			60	7200	14762
			80	12800	26082
			100	20000	40602
	3	1	20	1200	2562
			40	4800	9922
			60	10800	22082
			80	19200	39042
			100	30000	60802

FIGURE 26 – Ensemble test pour la structure de type Michell

Domain	L_x	L_y	N_l	n	d
Cantilever	2	1	20	800	1722
			40	3200	6642
			60	7200	14762
			80	12800	26082
			100	20000	40602
	4	1	20	1600	3402
			40	6400	13202
			60	14400	29402
			80	25600	52002
			100	40000	81002

FIGURE 27 – Ensemble test pour la structure de type Cantilever

Domain	L_x	L_y	N_l	n	d
MBB	1	2	20	800	1722
			40	3200	6642
			60	7200	14762
			80	12800	26082
			100	20000	40602
	1	4	20	1600	3402
			40	6400	13202
			60	14400	29402
			80	25600	52002
			100	40000	81002
	2	1	20	800	1722
			40	3200	6642
			60	7200	14762
			80	12800	26082
			100	20000	40602
	4	1	20	1600	3402
			40	6400	13202
			60	14400	29402
			80	25600	52002
			100	40000	81002

FIGURE 28 – Ensemble test pour la structure de type MMB

1.2.11.6 Expériences numériques

La bibliothèque pour l'analyse comparative contient 225 problèmes de compliance minimale, 135 problèmes de volume minimal. Cependant, la bibliothèque finale est constituée d'un sous-ensemble utilisant respectivement 121, et 64 car IPOPT, SNOPT et FMINCON dans la formulation SAND ont des problèmes de mémoire ou de temps de calcul (supérieur au maximum autorisé, ie 48h). Tous les calculs ont été effectués sur un processeur Intel Xeon X5650 à 6-cœurs, tournant à 2,66 GHz et avec 4 Go de mémoire pour chaque cœur.

1.2.11.7 Résultats numériques

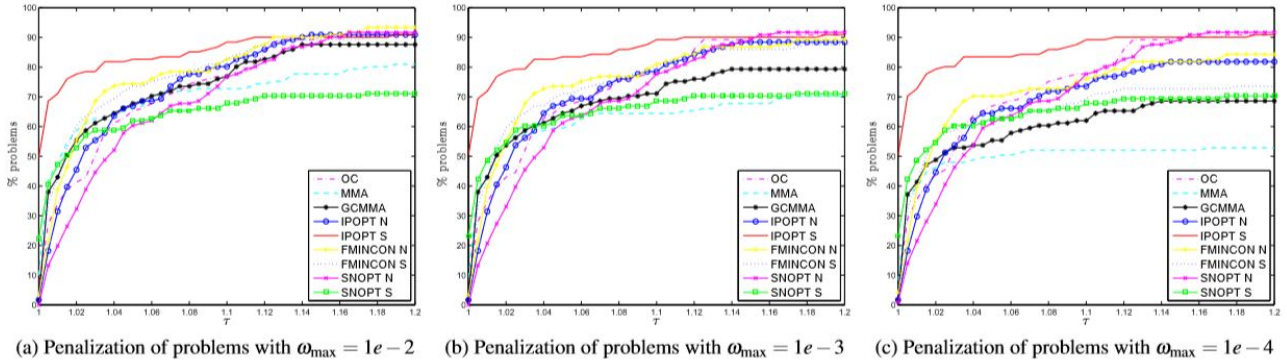
On considère qu'une conception x est incorrecte si la norme euclidienne des conditions KKT est supérieure à un seuil défini par $\omega_{max} = 1e-3$. La figure 29 montre l'impact de cette décision. La performance est mesurée avec la valeur de la fonction objectif. La plupart des solveurs sont capables d'obtenir une conception avec une erreur KKT inférieure à 1e-2. Cependant, pour $\omega_{max} = 1e-4$, il est plus probable que les solveurs échouent. Il est clair que des solveurs tels que MMA et GCMMA sont fortement affectés par ce seuil. Leurs performances diminuent considérablement lorsque nous appliquons la tolérance $\omega_{max} = 1e-4$. Par contre, le pourcentage de succès pour IPOPT et SNOPT n'est pas affecté par le paramètre ω_{max}. IPOPT et SNOPT produisent une conception dans laquelle les conditions d'optimalité sont remplies ou produisent des résultats très médiocres (voir dans la figure 30).

La figure 30a présente le profil de performance pour la valeur de la fonction objectif pour des petites valeurs τ, ce qui donne la performance des solveurs quand ils sont proches de la meilleure conception. IPOPT-S obtient la meilleure valeur de fonction objectif pour 50% des problèmes. Les chances pour le reste des solveurs de gagner sont petites, inférieures à 25%. Cependant, à mesure que τ augmente, IPOPT-N, SNOPT-N, FMINCON-N et FMINCON-S OC deviennent plus compétitifs.

La figure 30b montre, la robustesse ou fiabilité de chaque méthode. Quand $\tau \sim r_{max} = 10^2$, le pourcentage de problèmes est équivalent à la probabilité d'obtenir une conception. MMA ne peut obtenir un design que dans 70% des cas, SNOPT-S dans 75% et GCMMA dans 80%. La figure 30b montre également que FMINCON-N obtient une conception utilisant le plus petit nombre d'itérations suivi de SNOPT-N. Bien que IPOPT-S ait un nombre inférieur de victoires, il devient compétitif lorsque τ augmente.

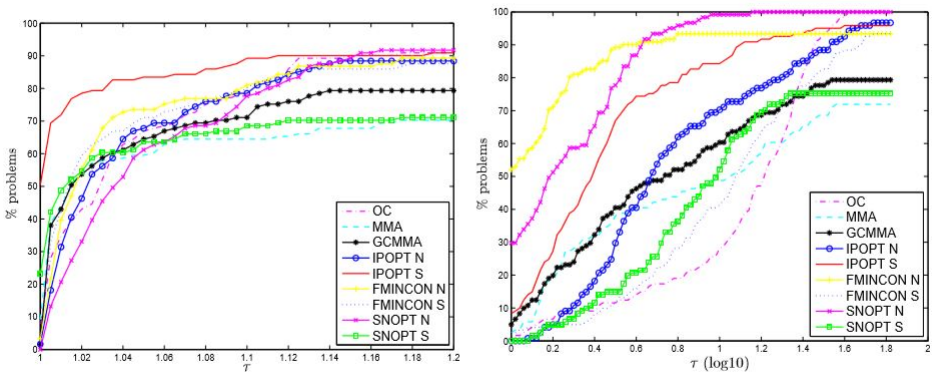
La figure 31 montre les profils de performance pour les 64 problèmes de volume minimal. Il est clair que IPOPT-S surpasse le reste des solveurs par rapport à la valeur de la fonction objectif obtenue, voir figure 31a. La probabilité pour IPOPT-S de gagner est supérieure à 55%. De plus, il a au moins 10% plus de chances que n'importe quel autre solveur d'obtenir une conception finale avec une valeur de fonction objectif à un facteur de $\tau = 1.2$. Contrairement aux problèmes de compliance minimale, FMINCON est incapable de résoudre plus de 70% des problèmes. MMA montre des performances similaires.

Cependant, IPOPT-S consomme plus d'itérations que SNOPT ou FMINCON. Néanmoins, IPOPT-S s'améliore très vite à mesure que τ augmente. Si nous cherchons un solveur capable de résoudre plus de 80% des problèmes avec la plus grande efficacité possible (en nombre d'itérations), le choix devrait alors être soit IPOPT-S, soit SNOPT. Leur performance pour ces 80% est à un facteur de $\tau = 5.6$



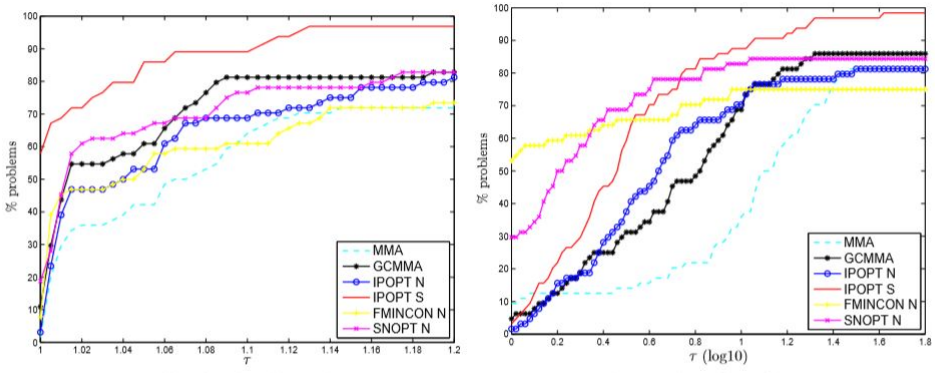
(a) Penalization of problems with $\omega_{\max} = 1e-2$ (b) Penalization of problems with $\omega_{\max} = 1e-3$ (c) Penalization of problems with $\omega_{\max} = 1e-4$

FIGURE 29 – Profils de performance pour l'ensemble test (réduit) de 121 problèmes de compliance minimale (P_N^c) et (P_S^c). La performance mesure la valeur de la fonction objective. Un problème est pénalisé dans les profils de performance si l'erreur KKT est supérieure à 1e-2 (a) 1e-3 (b) 1e-4 (c), respectivement.



(a) Objective function value (b) Number of iterations

FIGURE 30 – Profils de performance pour l'ensemble test (réduit) de 121 problèmes de compliance minimale (P_N^c) et (P_S^c). La performance est mesurée par la valeur de la fonction objectif (a) et nombre d'itérations (b).



(a) Objective function value (b) Number of iterations

FIGURE 31 – Profils de performance pour l'ensemble test (réduit) de 64 problèmes de minimum volume (P_N^w) et (P_S^w). La performance est mesurée par la valeur de la fonction objectif (a) et nombre d'itérations (b).

2 Implémentation en 3d sur HPC

On présente un cadre (ie, Topopt Petsc) flexible pour l'optimisation topologique, parallèle et facile à mettre en œuvre en utilisant PETSc (Portable and Extendable Toolkit for Scientific Computing). Ce cadre est basé sur une bibliothèque standardisée et libre d'accès (17 Août 2014, Niels Aage, Erik Andreassen, Boyan et Stefanov Lazarov), on résout le problème de compliance minimale sur un maillage réguliér, en utilisant la méthode des éléments finis standard, et des techniques de filtrage. Une implémentation parallèle de la méthode MMA, un exemple de problèmes de compliance minimale et d'homogénéisation sont inclus.

Ce cadre montre aussi comment l'optimisation topologique à grande échelle permet l'optimisation de problèmes en 3D avec des discrétisations encore inédites, et comment cela conduit à la découverte de nouveaux effets dans des problèmes d'optimisation des structures mécaniques bien connues. L'utilisation de PETSc réduit la taille de l'implémentation réelle et génère un code compact, qui est facile à lire, utiliser et étendre.

PETSc est une collection de bibliothèques parallélisées (et séquentielles) contenant la plupart des éléments de base nécessaires à l'optimisation topologque à grande échelle, ie matrices creuse, vecteurs, solveurs linéaires itératifs, solveurs non linéaires et schéma de pas à pas. Les principaux avantages de PETSc sont les suivants : a) Les bibliothèques sont testées de manière approfondie et le code est bien maintenu par des spécialistes. L'implémentation est parallèle à des milliers de cœurs et portable sur Linux, UNIX, Mac et Windows. b) Le code est écrit en C, orientée objet, de telle sorte que le code de base est facilement extensible et masque la complexité parallèle. c) La bibliothèque est extrêmement facile à installer et à utiliser.

Le cadre présenté dans ce présent travail est disponible gratuitement et peut être téléchargé à partir de `www.topopt.dtu.dk/PETSc`

2.1 TopOpt Petsc

Topopt Petsc (Niels Aage, Erik Andreassen, Boyan Stefanov Lazarov[40]) est une bibliothèque facile à utiliser et à étendre pour la résolution de divers problèmes d'optimisation topologique à grande échelle. Un diagramme des composants du code est présenté dans la figure 32. La base de TopOpt PETSc est la bibliothèque PETSc, et à partir du diagramme, on peut voir que le code se compose de cinq classes en C++ et d'un court programme principal. Le diagramme indique également quelles classes doivent être modifiées pour résoudre un nouveau problème d'optimisation, ie. : les cases noires signifient que des modifications sont susceptibles d'être requises et les cases rouges indiquent qu'aucune modification ne doit être appliquée. Les cases en pointillé rouge indiquent que des modifications peuvent être nécessaires, en fonction de l'application. Enfin, la complexité du code est indiquée par la grande flèche à gauche qui pointe dans la direction d'un code moins complexe.

2.2 Structure des classes

L'objectif étant d'obtenir un cadre polyvalent, les classes sont divisées en unités mutuellement indépendantes. Cela signifie par exemple que changer la physique du problème ne nécessite que des modifications d'une seule classe, ie. "physics class" qui dans le cas par défaut résout un problème d'élasticité linéaire. Une liste des classes avec description est donnée ci-dessous.

- `TopOpt` Contient des informations sur le problème d'optimisation, taille du maillage, paramètres et réglages généraux.
- `LinearElasticity` La classe des problèmes physique, qui résout le problème d'élasticité linéaire sur un maillage en 3D réguliers à l'aide d'éléments cubique (voir e.g. Zienkiewicz

and Taylor, 2000). Il contient également les méthodes nécessaires au problème de compliance minimale, ie. calculs de la fonction objectif, des contraintes et les sensibilités (Bendsøe et Sigmund, 2004). Le solveur linéaire par défaut est le multigrid CG V-cycle (ie, Galerkin projection multigrid preconditioned flexible GMRES with GMRES/SOR smoothing), qui suit la mise en oeuvre présentée dans (Amir et al., 2014) sauf pour le choix de la méthode Krylov et le lissage.

— `Filter/PDEFilter` Les classes de filtre, qui contient les filtres de sensibilité (Sigmund 1997), de densité (Bruns et Tortorelli 2001 ; Bourdin 2001) et de PDE (Lazarov et Sigmund 2011) via une interface commune.
— `MMA` La classe MMA, contenant une mise en œuvre entièrement parallélisée de la méthode MMA (Svanberg 1987), d'après la description donnée (dans Aage et Lazarov, 2013).
— `MPIIO` Une classe de sortie polyvalente capable de vider des données arbitraires dans un seul fichier binaire.

La distribution inclut également des scripts Python qui convertissent facilement les données de sortie binaires au format VTU (Schroeder et Martin 2003), qui peuvent être visualisées par exemple dans ParaView version 4 ou plus récente (Ahrens et al.2005).

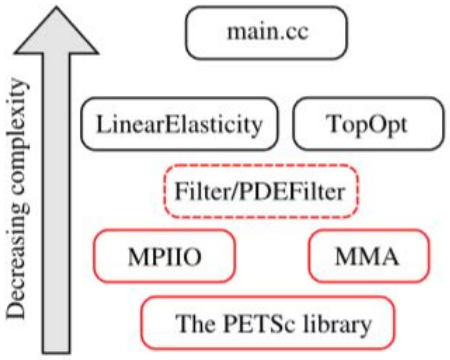


FIGURE 32 – Mise en œuvre d'optimisation topologique dans PETSc. Les noms dans les cases se rapportent au programme principal et aux classes contenant la physique du problème, les paramètres d'optimisation, le filtrage, la sortie, le solveur MMA et la bibliothèque PETSc. Les couleurs des cases indiquent si une classe donnée doit ou non être modifiée lors d'une modification du type de problème, de sorte que le rouge indique qu'il n'est pas nécessaire d'apporter des modifications, le pointillé rouge indique que des modifications mineures peuvent être nécessaires et le noir indique que des modifications sont probablement nécessaires. La flèche indique la complexité des différents composants, ie. plus haut dans le diagramme, plus simple le code.

2.3 Résultats numériques du benchmarking sur HPC

Nous proposons une analyse compartive pour les solveurs OC, MMA, GCMMA, et quelques solveurs basés sur les méthodes de point intérieurs et de SQP dans (IPOPT, NLOPT), et on examine les performances en utilisant les profils performance et le "data profiles", quand ces solveurs sont appliqués à divers problèmes d'optimisation topologique de structures mécaniques sous contraintes dans le cadre de l'élasticité linéaire. Une bibliothèque complète, représentative d'une classe de problèmes de compliance minimale et de volume minimal d'une structure élastique pour différentes tailles E.F est devéloppée pour cette analyse comparative. Pour un futur travail, cette bibiothèque pourrait servir de point de départ pour d'autres analyse comparative. Cet ensemble de tests est

couramment utilisé en optimisation topologique (Susana Rojas-Labanda, Mathias Stolpe, 17 Mai 2015 [41], [4], [13], [2], [14], et [6]).

Une analyse comparative est d'abord effectuée sur 120 problèmes de compliance minimale dont 90 des 120 problèmes (45 de type poutre encastrée et 45 de type michell) sont définis dans CUTE[11] et COPS[10] et les 30 restants sont des nouveaux problèmes rajoutés (type 3D wheel ou chaise). Trois types de domaines de conception différents et chargement sont considérés (voir figure 35). La bibliothèque utilisée est definie en détail ci-aprés. Les domaines de conception, les conditions aux limites et les chargement externes sont tirés de la littérature telle que (Susana Rojas-Labanda, Mathias Stolpe, 17 Mai 2015 [55][6],[12],[4], [13], [8], [61], [54], [17], et [52]). Les conditions de tests sont décrites dans la figure 33, 34, 35, 36.

Nous avons décidé de nous concentrer sur les performances des solveurs en utilisant une approche classique (la formulation Nested) plutôt que d'étudier les performances pour différentes formulations d'optimisation topologique.

Solveur	Paramètre	Description	Valeur
MMA, GCMMA	tol kkt	norme euclidienne de l'erreur kkt	1e-5
OC	ch	différence dans la variable de conception	1e-4
IPOPT	tol	tolérance dans problème NLP	1e-6
NLOPT	tol	tolérance dans la variable primal	1e-6

Solveur	Paramètre	Description	Valeur
MMA, GCMMA	max iter Ext	nombre d'itération maximum	200
OC	boucle	nombre d'itération maximum	200
IPOPT	max iter	nombre d'itération maximum	200
NLOPT	lim iter	nombre d'itération maximum	200

FIGURE 33 – Ensemble test des critères de convergence

Solveur	Problème	E_1	E_0
MMA, GCMMA	Compliance/ volume	1	1e-9
OC	Compliance minimale	1	1e-9
IPOPT/NLOPT	Compliance minimale	1	1e-9
IPOPT/NLOPT	Minimum volume	1e1	1e1

FIGURE 34 – Ensemble test des valeurs du module de Young

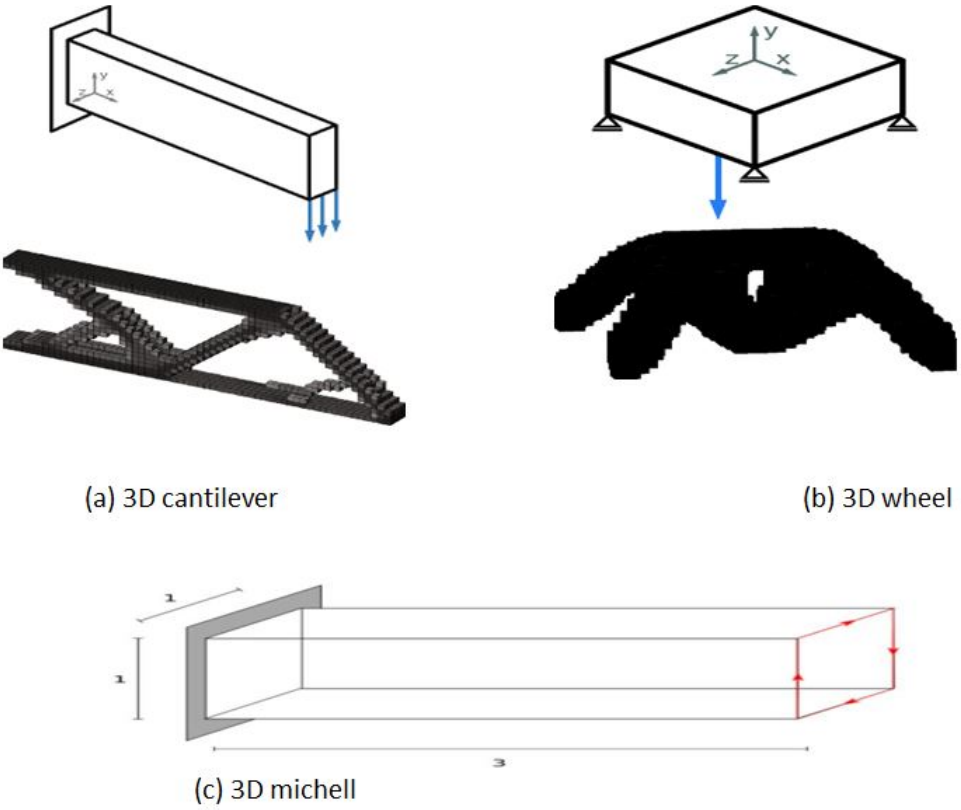


FIGURE 35 – Ensemble test de compliance minimale

Domaine	L_x	L_y	L_z	N_x	N_y	N_z	n	d
	2	1	1	88	88	176	1362944	4206051
	2	1	1	176	88	88	1362944	4206051
	2	1	1	128	64	64	524288	1635075
	4	1	1	88	88	176	1362944	4206051
Michell,	4	1	1	176	88	88	1362944	4206051
Wheel,	4	1	1	128	64	64	524288	1635075
Cantilever	4	3	1	88	88	176	1362944	4206051
	4	3	1	176	88	88	1362944	4206051
	4	3	1	128	64	64	524288	1635075

FIGURE 36 – Ensemble tests pour la structure Michell, Cantilever, Wheel. L_x, L_y, L_z sont les mesures des longueurs dans la direction x,y,z respectivement. N_x, N_y, N_z sont les nombres d'éléments dans la direction x,y,z respectivement. n est le nombre total d'éléments. d est le nombre de degrés de liberté.

Point de départ des problèmes

Un point important qui pourrait affecter les performances des solveurs est le point de départ. Le point de départ x_0 est initialisé à $x_0 = 0.5e$, avec $e = (1, ., 1)^T$. Le vecteur de déplacement U_0 est initialisé à $U_0 = 0$ pour la classe de compliance minimale et volume minimal.

Compliance minimale et volume minimal

Trois types différents de domaines de conception et de chargement sont pris en compte (voir figure 35) pour la classe de compliance minimale et volume minimal. Pour chaque domaine, nous considérons différents rapports de longueur, étant donné que nous avons constaté que les solveurs rencontrent des difficultés pour résoudre des problèmes avec une grande différence de longueur. De plus, pour chaque longueur, des tailles différentes de discrétisation, N_{lx}, N_{ly}, N_{lz}, sont définies. N_{lx}, N_{ly}, et N_{lz} font référence au nombre d'éléments dans la direction x,y,z respectivement dans la base cartésienne. Nous avons décidé de construire un ensemble de test avec un nombre assez grand d'éléments (ie, $\sim$ 1362944 nombre d'éléments et 4206051 DDL) pour tester tous les solveurs. Enfin, pour chaque domaine de conception dans la classe de compliance minimale, 5 limites supérieures différentes de fraction volumique sont considérées, ie volfrac = 0.1, 0.2, 0.3, 0.4, 0.5. En pratique, la résolution de problèmes avec de petites limites de fraction volumique est généralement plus difficile. En général, lorsque la quantité de matériau disponible est réduite, les solveurs ont plus de difficultés à distribuer la matière de sorte que la structure satisfasse les contraintes et minimise la fonction objectif.

Pour la classe de volume minimal, la contrainte supérieure sur la compliance est donnée par :

$$C = k(F^T(K^{-1}(x_0)F),$$

pour une constante $k \geq 1$ définie par l'utilisateur.

Profils de performance

On presente dans la figure 37, 38, 39 ci-dessous les résultats du benchmarking des solveurs classique OC, MMA, GCMMA

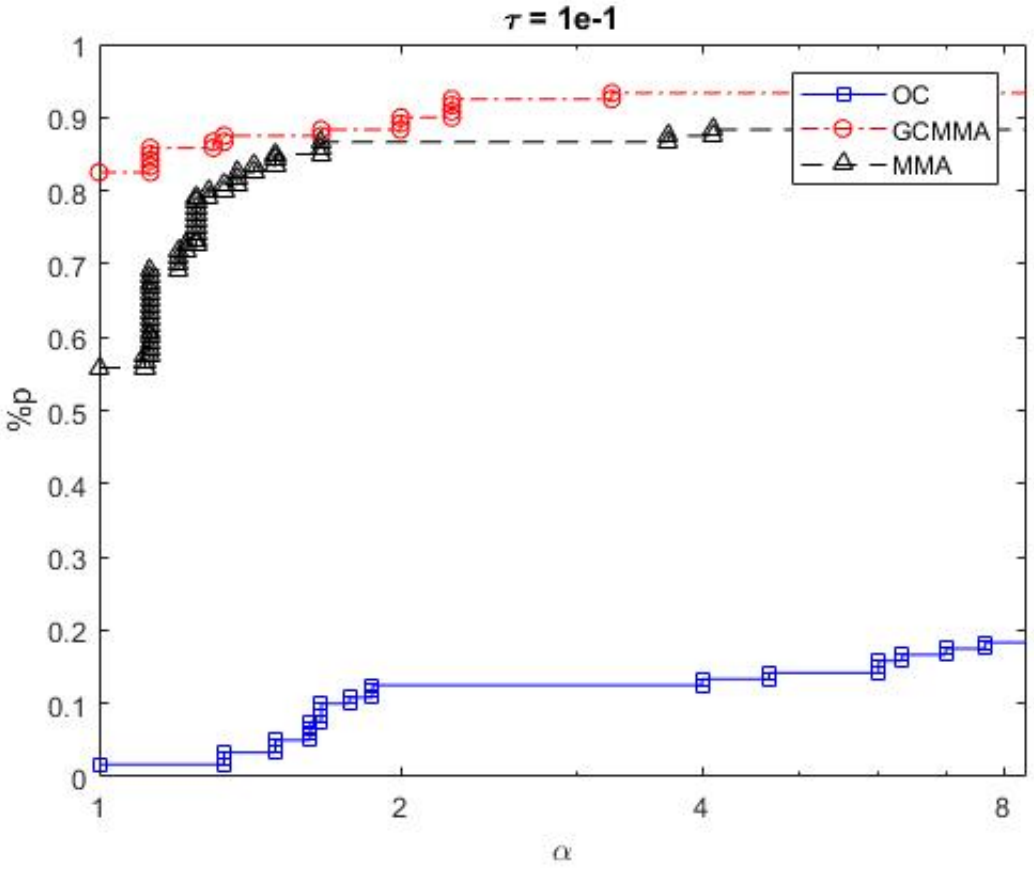


FIGURE 37 – Profils de Performance pour l'ensemble tests de 120 problèmes de compliance minimale (P_N^c). La performance mesure la valeur de la fonction objectif. Un problème est pénalisé dans les profils de performance si le test de convergence pour une tolérance donnée τ n'est pas satisfait.

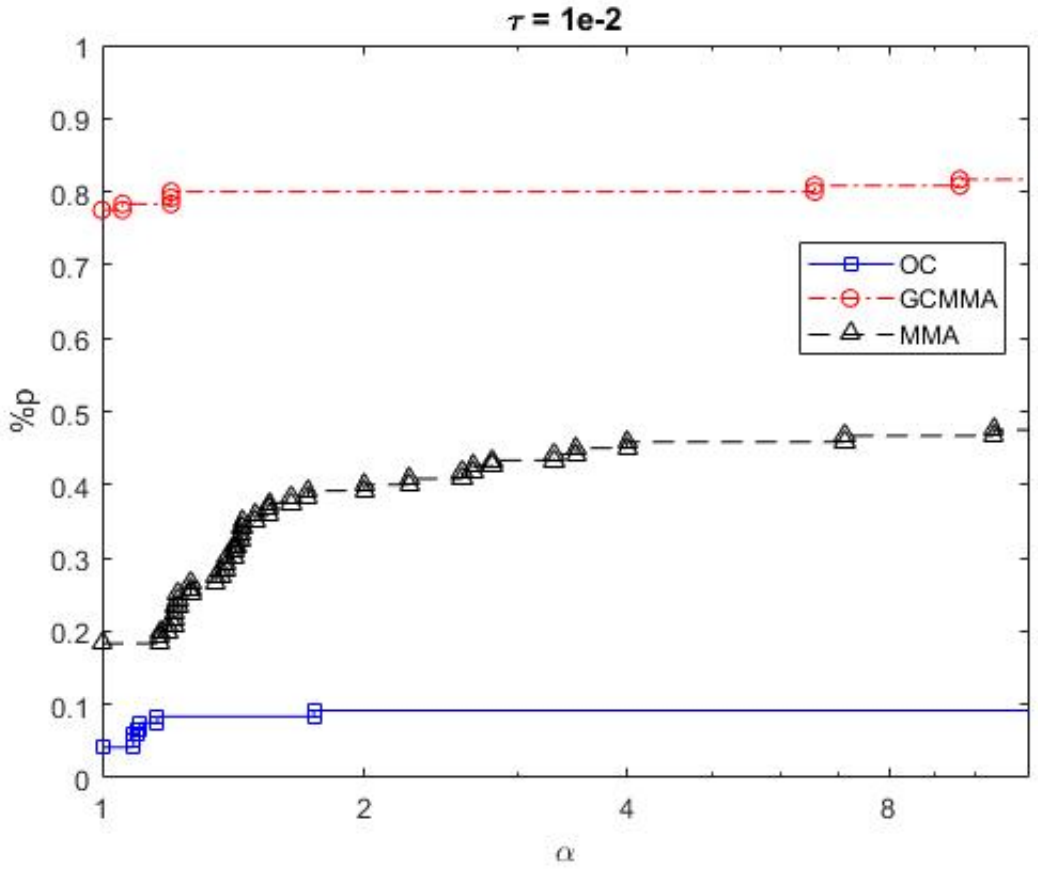


FIGURE 38 – Profils de performance pour l'ensemble tests de 120 problèmes de compliance minimale (P_N^c). La performance mesure la valeur de la fonction objectif. Un problème est pénalisé dans les profils de performance si le test de convergence pour une tolérance donnée τ n'est pas satisfait.

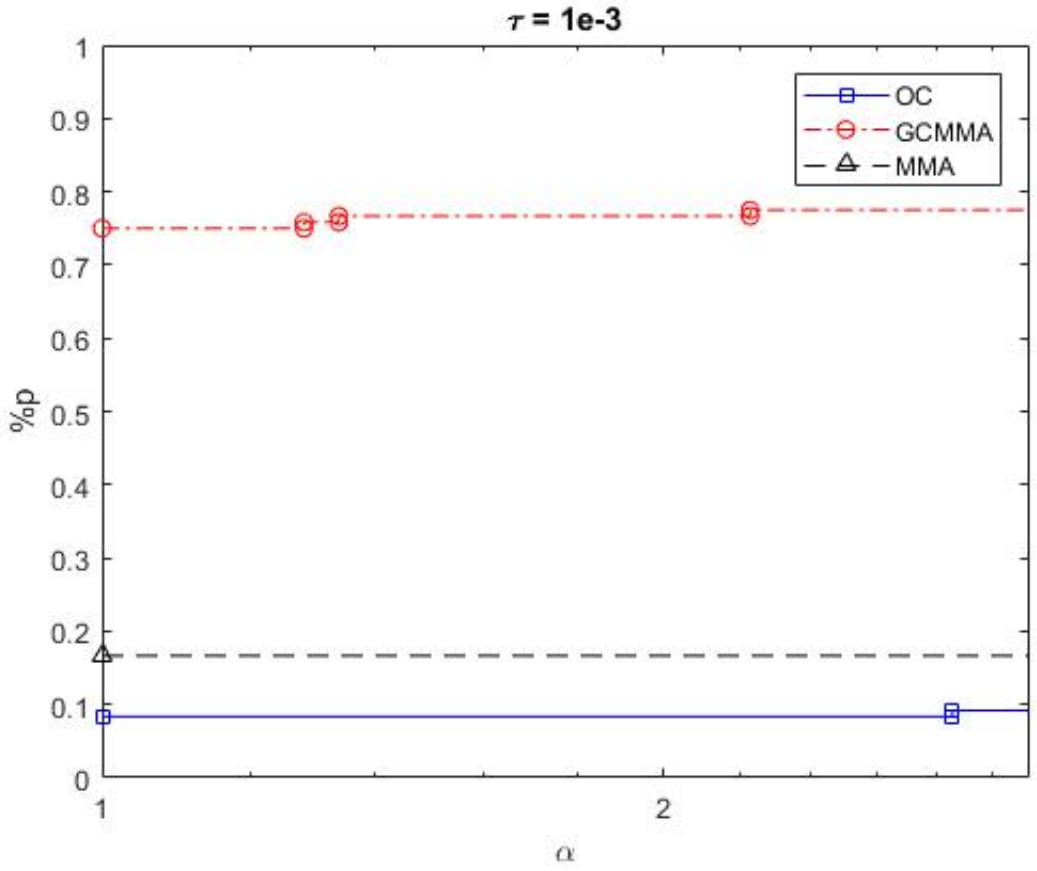


FIGURE 39 – Profils de performance pour l'ensemble tests de 120 problèmes de compliance minimale (P_N^c). La performance mesure la valeur de la fonction objectif. Un problème est pénalisé dans les profils de performance si le test de convergence pour une tolérance donnée τ n'est pas satisfait.

Data profiles

On presente dans la figure 40, 41, 42 ci-dessous les résultats du benchmarking des solveurs classique OC, MMA, GCMMA

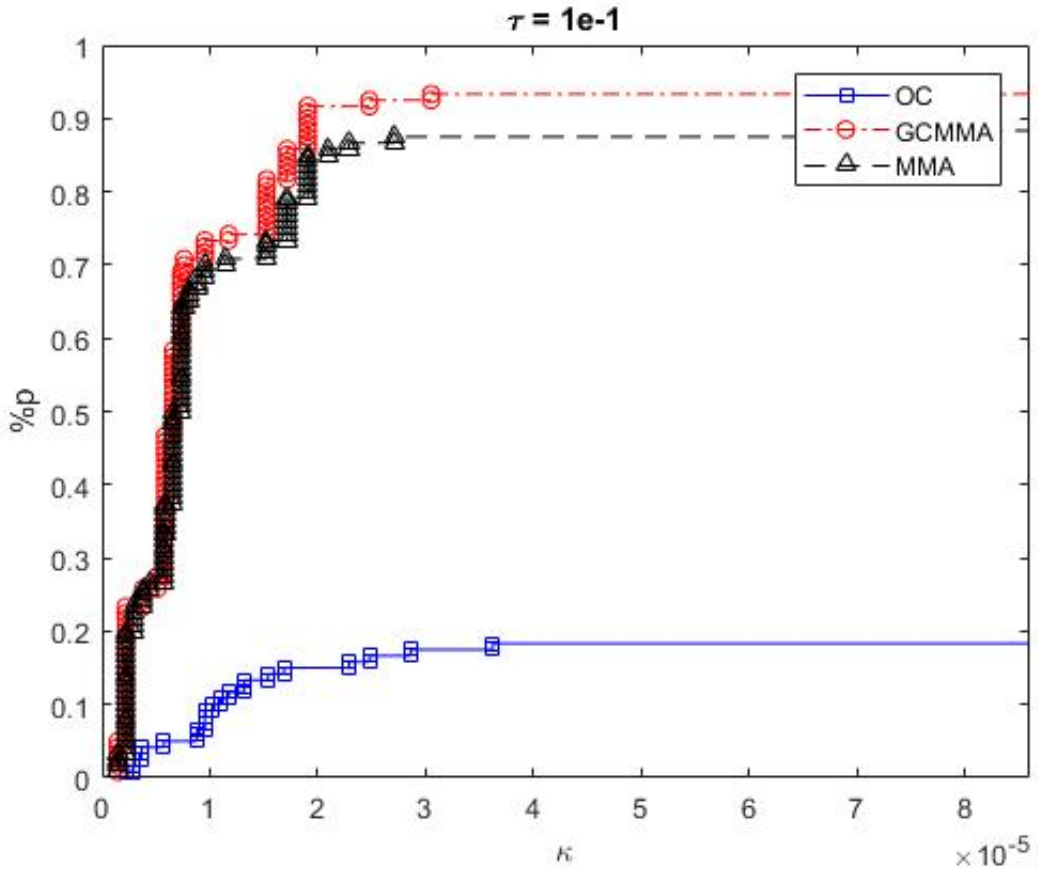


FIGURE 40 – Data profiles pour l'ensemble tests de 120 problèmes de compliance minimale (P_N^c). La performance mesure la valeur de la fonction objectif. Un problème est pénalisé dans le Data profiles si le test de convergence pour une tolérance donnée τ n'est pas satisfait.

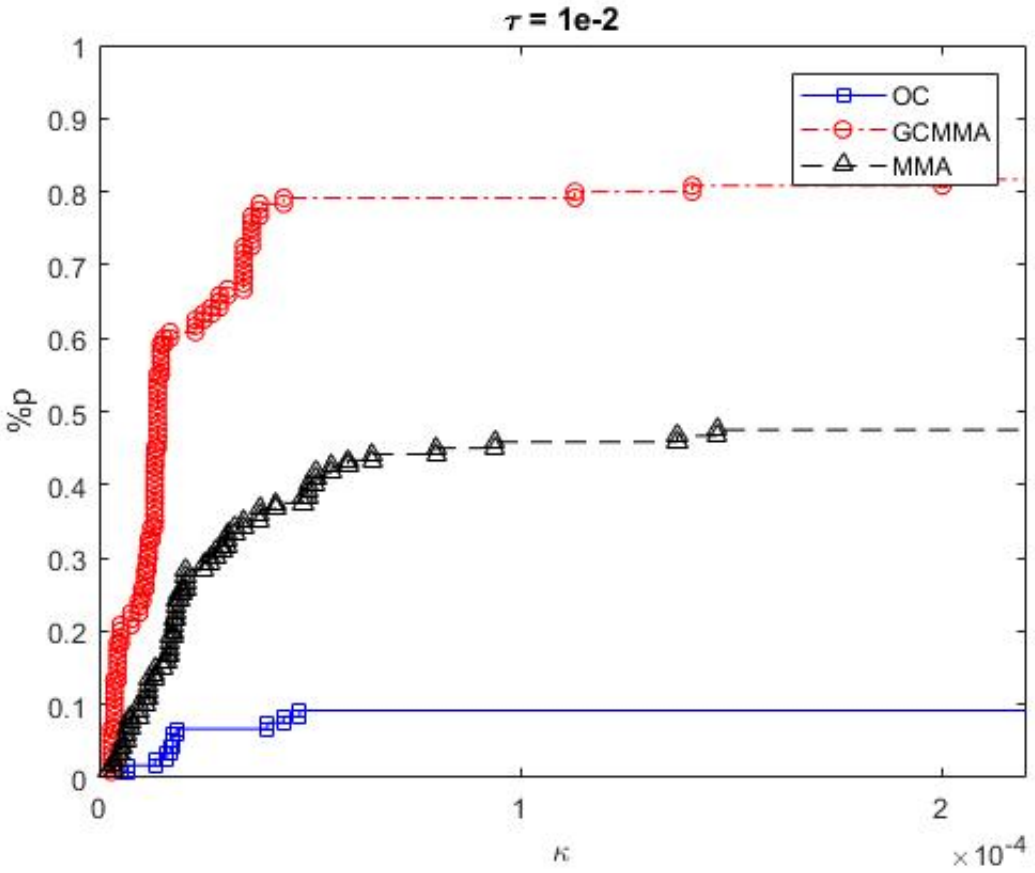


FIGURE 41 – Data profils pour l'ensemble tests de 120 problèmes de compliance minimale (P_N^c). La performance mesure la valeur de la fonction objectif. Un problème est pénalisé dans le Data profiles si le test de convergence pour une tolérance donnée τ n'est pas satisfait.

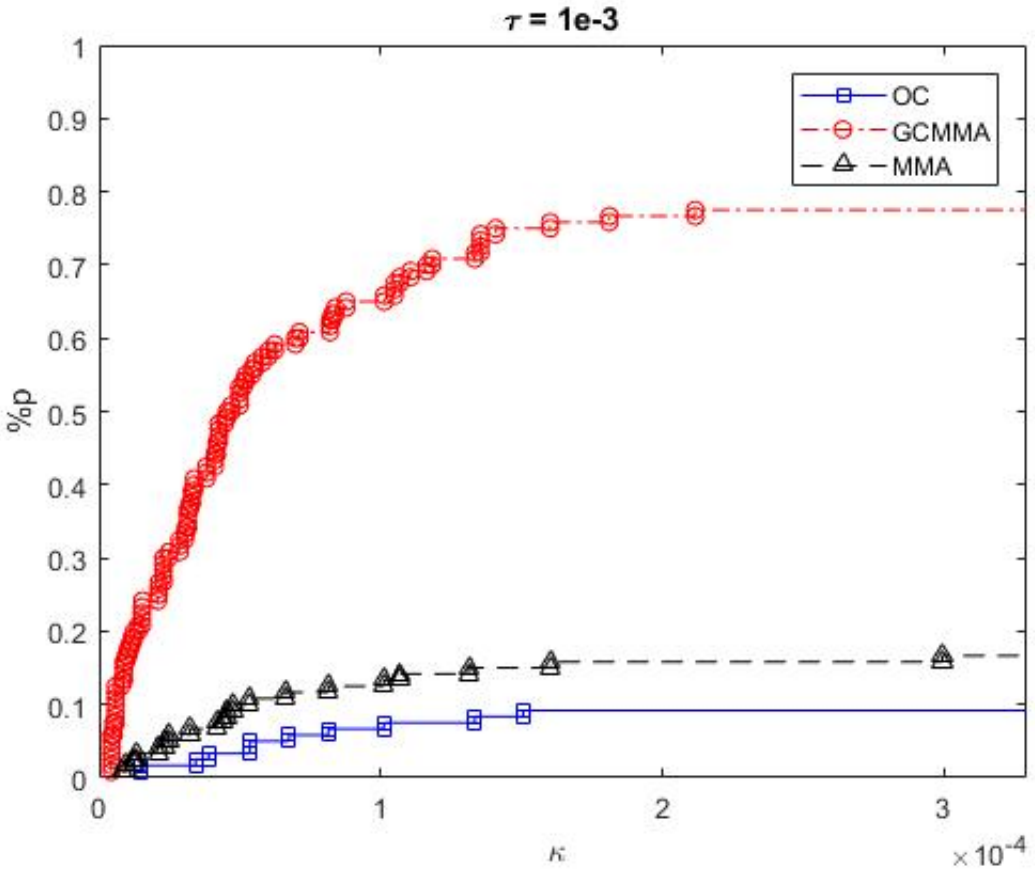


FIGURE 42 – Data profils pour l'ensemble tests de 120 problèmes de compliance minimale (P_N^c). La performance mesure la valeur de la fonction objectif. Un problème est pénalisé dans le Data de profiles si le test de convergence pour une tolérance donnée τ n'est pas satisfait.

3 Conclusion et futur travail

On considère qu'un solveur $s \in \mathcal{S}$ produit une conception x incorrecte (ie, le solveur s ne réussit pas à résoudre un problèmes donné, $p \in \mathcal{P}$) si le test de convergence (120) pour une tolérance donné τ est violé (ie, le nombre d'évaluations de la fonction objectif par le solveur s dépasse le nombre d'évaluations maximum μ_f, de la fonction objectif). La performance est mesurée avec la valeur de la fonction objectif. Les solveurs MMA, GCMMA, sont capables d'obtenir une conception avec une tolérance égale à $\tau = 1e-1$. Cependant, pour $\tau = 1e-3$, il est plus probable que les solveurs OC, MMA, échouent. Il est clair que des solveurs tels que OC, MMA sont fortement affectés par ce seuil. La performance du solveur MMA diminue considérablement lorsque nous appliquons la tolérance $\tau = 1e-2$ et $\tau = 1e-3$. Mais la performance du solveur OC est globalement trés médiocre et diminue légèrement lorsque nous appliquons la tolérance $\tau = 1e-2$ et $\tau = 1e-3$. GCMMA se comporte assez bien, bien qu'il soit également affecté lorsque nous appliquons la tolérance $\tau = 1e-2$ et $\tau = 1e-3$.

Les figures 37, 38, 39 présentent les profils de performance pour la valeur de la fonction objectif pour différentes valeurs α, ce qui donne la performance des solveurs quand ils sont proches de la meilleure conception. GCMMA obtient la meilleure valeur de fonction objectif pour presque 80% des problèmes. Les chances pour le reste des solveur (MMA, OC) de gagner sont petites, inférieures à 20%. Cependant, à mesure que α augmente (pour une tolérance $\tau = 1e-1$), MMA devient plus compétitif.

Les figures 40, 41, 42 montrent la fiabilité de chaque méthode. Quand $\kappa = \kappa_{max} = 10^5$, le pourcentage de problèmes est équivalent à la probabilité d'obtenir une conception. pour $\tau = 1e-1$, MMA ne peut obtenir un design que dans 87% des cas, GCMMA dans 95% et OC dans 19%. Pour $\tau = 1e-3$, MMA ne peut obtenir un design que dans 19% des cas, GCMMA dans 78% et OC dans 10%. MMA diminue considérablement, ce qui nous fait penser qu'il faut augmenter le maximum d'évaluations μ_f afin d'améliorer la fiabilité du solveur MMA. Il est clair que GCMMA surpasse le reste des solveurs (MMA, OC) par rapport à la valeur de la fonction objectif.

Les profils de performance et le "Data profiles" concluent que la classe de méthodes de programmation séquentielle convexe (MMA, GCMMA) sont plus efficaces, plus fiables que la méthode empirique (OC). Les solveurs MMA, GCMMA produisent généralement des variables de conception avec de meilleures valeurs de fonction objectif.

Pour le travail futur, un benchmarking des solveurs généraux d'optimisation y compris méthodes de points intérieurs dans IPOPT, des méthodes SQP dans NLOPT seront analysées quand appliquées aux problèmes de compliance minimale et volume minimal. Chaque fois que c'est possible, les méthodes seront appliquées à la formulation Nested.

Bibliographie

[1] Davis T. A. *Direct methods for sparse linear systems*, volume 2. Siam, Gainesville, Florida,USA, 2006.

[2] T. Tsuchiya A. S. El-Bakry, R. A. Tapia and Y. Zhang. On the formulation and theory of the Newton interior-point method for nonlinear programming. *Journal of Optimization Theory and Application*, 89(3) :507–541, 1996.

[3] S. Bakhtiari A.L. Tits, T.J. Urban and C.T. Lawrence. A primal-dual interior point method for nonlinear programming with strong global and local convergence properties., july 2001.

[4] Clausen A. Schevenels M. Lazarov B.S. Sigmund O Andreassen, E. Efficient topology optimization in MATLAB using 88 lines of code. *Structural and Multidisciplinary Optimization*, 43(1) :1–16, 2011.

[5] M. P. Bendsøe. Optimal shape design as a material distribution problem. *Springer*, 1(4), 1989.

[6] M.P Bendsøe. Optimal shape design as a material distribution problem. *Structural Optimization*, 1 :192–202, 1995.

[7] Sigmund O. Bendsøe, M.P. Theory, methods and applications. [On topology optimization]. *Springer*, 2003.

[8] Sigmund O Bendsøe, M.P. Topology optimization : Theory, methods and applications. *Springer*, 2003.

[9] Shanno D.F. Vanderbei R.J Benson, H.Y. A comparative study of large-scale nonlinear optimization algorithms. Tech.rep, 2002.

[10] Bortz D.M. Moré J.J Bondarenko, A.S. COPS : Large-scale nonlinearly constrained optimization problems, 1999.

[11] Conn A.R. Gould N. Toint P.L Bongartz, I. CUTE : Constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, 21(1) :123–160, 1995.

[12] Duysinx P Bruggi, M. Topology optimization for minimum weight with compliance and stress constraints. *Structural and Multidisciplinary Optimization*, 46(3) :369–384, 2012.

[13] T.E Bruns. A reevaluation of the SIMP method with filtering and an alternative formulation for solid void topology optimization. *Structural and Multidisciplinary Optimization*, 30(6) :428–436, 2005.

[14] Simone CONIGLIO. *Propulsion Airframe Topology Optimization with performance and stress criteria.* PhD thesis, Institut Supérieur de l'Aéronautique et de l'Espace, Toulouse, FRANCE, 2019.

[15] Gould N.I.M. Toint P.L. Conn, A.R. Lancelot : A FORTRAN Package for Large-Scale Nonlinear Optimization (Release A). *Springer-Verlag New York, Inc., Secaucus, NJ, USA*, 1992.

[16] M. L. Overton D. M. Gay and M. H. Wright. *A primal-dual interior method for nonconvex nonlinear programming.* Kluwer Academic, Dordretch, 1998.

[17] Dinesh M. Sahu D.K. Ananthasuresh G.K Deepak, S.R. A comparative study of the formulations and benchmark problems for the topology optimization of compliant mechanisms. *Journal of Mechanisms and Robotics*, 1(1), 2009.

[18] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles, 2002.

[19] Moré J.J. Dolan, E.D. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91 :201–213, 2002.

[20] Clemens Elster and Arnold Neumaier. A grid algorithm for bound constrained optimization of noisy functions, 1995.

[21] A. Forsgren and P. E. Gill. Primal-dual interior methods for nonconvex nonlinear programming. *SIAM Journal on Optimization*, 8(4) :1132–1152, 1998.

[22] Orban D. Toint P.L. Gould, N.I.M. CUTEr and SifDec : A constrained and unconstrained testing environment, revisited. *ACM Transactions on Mathematical Software*, 29(4) :373–394, 2003.

[23] Michaël Bruyneel . Jean-Charles Craveur . Pierre Gourmelen. *OPTIMISATION DES STRUCTURES MÉCANIQUES*. DUNOD, Grands-Augustins, Paris, 2014.

[24] Joshua D. Griffin and Tamara G. Kolda. Nonlinearly-constrained optimization using asynchronous parallel generating set search, May 2007.

[25] D. F. Shanno H. Y. Benson and R. J. Vanderbei. Interior-point methods for nonconvex nonlinear programming : Jamming and comparative numerical testing, August 2000.

[26] D. F. Shanno H. Y. Benson and R. J. Vanderbei. Interior-point methods for nonconvex nonlinear programming : Filter methods and merit functions., December 2000.

[27] H. Yabe H. Yamashita and T. Tanabe. A globally and superlinearly convergent primal-dual interior point trust region method for large scale constrained optimization, July 1997. Revised July 1998.

[28] D.S Kershaw. The incomplete cholesky–conjugate gradient method for the iterative solution of systems of linear equations. [On struct multidisc optim]. *Journal of computational physics*, 26,(Issue 1) :43–65, 1978.

[29] Achterberg T. Andersen E. Bastert O. Berthold T. Bixby R.E. Danna E. Gamrath G. Gleixner A.M. Heinz S. Lodi A. Mittelmann H. Ralphs T. Salvagnin D. Steffy D.E. Wolter K. Koch, T. MIPLIB 2010. *Mathematical Programming Computation*, 3(2) :103–163, 2011.

[30] J. E. Rooda L. F. P. Etman, Albert A. Groenwold. First-order sequential convex programming using approximate diagonal QP subproblems. *Springer*, 45, 2012.

[31] K. Liu and A. Tovar. An efficient 3D topology optimization code written in Matlab. [On struct multidisc optim]. *Springer-Verlag*, 50,(Issue 6) :1175–1196, 2014.

[32] O. Sigmund M. P. Bendsøe. Material interpolation schemes in topology optimization. *Springer-Verlag*, 69 :635–654, 1999.

[33] S. Ulbrich M. Ulbrich and L. N. Vicente. A globally convergent primal-dual interior-point filter method for nonconvex nonlinear programming, 2000.

[34] Marcelo Marazzi and Jorge Nocedal. Wedge trust region methods for derivative free optimization, March 2002.

[35] H. P. Mlejnek. Some aspects of the genesis of structures. *ScienceDirect*, 5(1-2), March 1992.

[36] Jorge J. Moré and Stefan M. Wild. Benchmarking Derivative-Free Optimization Algorithms, April 2008.

[37] Saunders M.A. Murtagh, B.A. MINOS 5.5 User's Guide. Tech. rep., Stanford University Systems Optimization Laboratory, Department of Operations Research, 1998.

[38] I.N.Rozvany M.ZhouG. The COC algorithm, Part II : Topological, geometrical and generalized shape optimization. *ScienceDirect*, 89(1-3), August 1991.

[39] S. G. Nash and A. Sofer. A barrier method for large-scale constrained optimization. *ORSA Journal on Computing*, 1(4) :40–53, 1993.

[40] Boyan Stefanov Lazarov Niels Aage, Erik Andreassen. Topology optimization using PETSc : An easy-to-use, fully parallel, open source topology optimization framework. *Springer*, 51, Verlag Berlin Heidelberg 2014.

[41] J. Nocedal and S. Wright. Numerical Optimization. *Springer*, 1999.

[42] Wächter R. Waltz R.A. Nocedal, J. Adaptive barrier update strategies for nonlinear interior methods. *SIAM Journal on Optimization*, 19(4) :1674–1693, 2009.

[43] Boyan S. Lazarov Oded Amir, Niels Aage. On multigrid-CG for efficient topology optimization. *Springer*, 49, Verlag Berlin Heidelberg 2013.

[44] E. Omojokun. *Trust Region Algorithms for Optimization with Nonlinear Equality and Inequality Constraints*. PhD thesis, University of Colorado, Boulder,CO, USA, 1989.

[45] Yoshio Oyanagi Osamu Tatebe. Efficient Implementation of the Multigrid Preconditioned Conjugate Gradient Method on Distributed Memory Machines. *IEEE Computer Society Press Los Alamitos*, pages 194–203, 1994.

[46] G. Liu R. H. Byrd and J. Nocedal. On the local behavior of an interior point method for nonlinear programming., 1997.

[47] J. Ch. Gilbert R. H. Byrd and J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89 :149–185, 2000.

[48] J. Nocedal R. H. Byrd and R. A. Waltz. Feasible interior methods using slacks for nonlinear optimization, December 2000.

[49] M. E. Hribar R. H. Byrd and J. Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4) :877–900, 1999.

[50] Luca RATTI. Topology optimization for 3d printing, technology. Supervisors : Prof. Nicola PAROLINI and Prof. Marco VERANI, Seb 2015.

[51] Y. Saad. *Iterative methods for sparse linear systems*, volume 82. Siam, PA, USA, 2003.

[52] Ananthasuresh G.K. Saxena, A. Topology synthesis of compliant mechanisms for nonlinear force-deflection and curved path specifications. *Journal of Mechanical Design*, 123(1) :33–42, 2001.

[53] Erik Andreassen · Anders Clausen · Mattias Schevenels · Boyan S. Lazarov · Ole Sigmund. Efficient topology optimization in MATLAB using 88 lines of code. [On topology optimization]. *Springer-Verlag*, 2010.

[54] O Sigmund. On the design of compliant mechanisms using topology optimization. *Journal of Structural Mechanics*, 25(4), 1997.

[55] O Sigmund. Manufacturing tolerant topology optimization. *Acta Mechanica Sinica*, 25 :227–239, 2009.

[56] Remi Amargier Joseph Morlier Simone Coniglio, Christian Gogu. Engine Pylon Topology Optimization Framework Based on Performance and Stress Criteria. *American Institute of Aeronautics and Astronautics, Inc*, 28 Aug 2019.

[57] Robert D. Falgout Steven F. Ashby. A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations. *Nuc. Sci. and Eng.*, 124 :145–159, 1996.

[58] Susana Rojas-Labanda · Mathias Stolpe. Benchmarking optimization solvers for structural topology optimization. [On structural and multidisciplinary optimization]. *Springer*, 52,(Issue 3) :527–547, 2015.

[59] Krister Svanberg. MMA and GCMMA – two methods for nonlinear optimization, 2007.

[60] R. J. Vanderbei and D. F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13 :231–252, 1999.

[61] Lazarov. B.S. Sigmund. O. Wang, F. On projection methods, convergence and robust formulations in topology optimization. *Structural and Multidisciplinary Optimization*, 43(6) :767–784, 2011.

[62] Andreas Wächter. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering.* PhD thesis, Carnegie Mellon University,Pittsburgh, Pennsylvania,USA, January 2002.

[63] Biegler L.T Wächter, A. On the implementation of an interior point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1) :25–57, 2006.

[64] H. Yamashita. A globally convergent primal-dual interior-point method for constrained optimization. *Optimization Methods and Software*, 10 :443–469, 1998.