Optimisation bayésienne sous contraintes et en grande dimension appliquée à la conception avion avant projet.

Rémy Priem

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Institut Supérieur de l'Aéronautique et de l'Espace (ISAE)*

Présentée et soutenue le *24 novembre 2020* **par :**

Rémy PRIEM

Optimisation bayésienne sous contraintes et en grande dimension appliquée à la conception avion avant projet.

JURY

OLIVIER ROUSTANT	Prof., INSA Toulouse	Président
SÉBASTIEN LE DIGABEL	Prof., Polytechnique Montréal	Rapporteur
RODOLPHE LE RICHE	Dir. de recherche, LIMOS	Rapporteur
VICTOR PICHENY	Ing. de recherche, Secondmind	Examinateur
NATHALIE BARTOLI	Maître de recherche, ONERA	Directrice
YOUSSEF DIOUANE	Prof. Associé, ISAE-SUPAERO	Co-Directeur
HUGO GAGNON	Ing. de recherche, Bombardier	Invité
SYLVAIN DUBREUIL	Ing. de recherche, ONERA	Invité

École doctorale et spécialité :
AA : Aéronautique et Astronautique
Unité de Recherche :
ISAE-ONERA MOIS - MOdélisation et Ingénierie des Systèmes
Directeur(s) de Thèse :
Nathalie BARTOLI et *Youssef DIOUANE*
Rapporteurs :
Sébastien LE DIGABEL et *Rodolphe LE RICHE*

Table des matières

Remerciements

« *Elle regarda les visages qui l'entouraient, les nomma et les compta comme des bijoux dans une boite au trésor. Ses amis.* »

Robin Hobb

Ce document est le fruit de trois années de travail qui auraient été impossible de réaliser sans le soutien de nombreuses personnes et amis. Tout d'abord, je voudrais remercier Nathalie BARTOLI qui a dirigé ces travaux avec passion pendant trois ans. Sa joie de vivre et son implication sont pour beaucoup dans l'avancement des recherches présentées dans ce manuscrit. De même, Youssef DIOUANE m'a obligé à être plus exigeant avec moi-même et je l'en remercie.

Je remercie également Sylvain DUBREUIL et Joseph MORLIER pour leur soutiens et leur confiance et leur enthousiasme tout au long de ces trois ans. J'adresse ma reconnaissance à Hugo GAGNON pour son accueil chaleureux lors de mon stage au Canada à Bombardier Aviation. Sans lui, rien dans cette aventure n'aurait été possible.

Mes sincères remerciements à Olivier ROUSTANT d'avoir présidé mon jury de thèse, à Rodolphe LE RICHE et Sébastien LE DIGABEL pour la relecture très précise du manuscrit de thèse ainsi que pour leurs conseils avisés et à Victor PICHENY d'avoir accepté d'examiner ma thèse et pour ces questions pertinentes.

Ma gratitude va aussi à Ian CHITTICK et Stéphane DUFRESNE pour leur collaboration aux travaux réalisés à Bombardier ainsi qu'à Alessandro SGUEGLIA pour nos différentes coopérations.

Pendant ces trois années, j'ai pu côtoyer de nombreuses personnes du département DTIS de l'ONERA et du DISC de l'ISAE-SUPAERO. Je les remercie pour leur accueil et leur bonne humeur. J'adresse entre autres ma gratitude à Pierre, Maxime, Gaspard, Marco, Vincenzo, Paul, Rémy, Gabriel, Tiphaine, Vincent, Rémi, Nina membres du club CDD du DTIS, pour cette ambiance sans pareil et les parties de tarots qui me manqueront surement. Je remercie plus particulièrement ma work wife et amie Morgane MENZ pour son soutiens indéfectible durant ces trois ans.

Je suis aussi très reconnaissant à la DGA d'avoir attendu plus de 10 mois avant de me compter parmi eux. Cette possibilité m'a permis de passer sereinement du statut de doctorant à ingénieur.

Le soutien de mes amis à également une part importante dans la réalisation de ces travaux de thèse. Je remercie donc Aurore MOCQUANT, Benoit COUTURE, Arnaud DEPONT, Yale LEE, Ludovic BECQ, Marie-Morgane IMBAULT, Nadia VOEDTS, Aline LEAL et Xavier LEAL pour tous ces moments de joies partagés.

Je n'oublie pas ma famille qui a toujours cru en moi notamment mes parents Cathy et Marc PRIEM, ma sœur Lydie DAMBRUNE, mon frère Jonathan PRIEM, mon beau-frère Raphael DAM-

BRUNE, ma belle-sœur Justine LIEBERT, mes filleuls Léa DAMBRUNE et Batiste PRIEM ainsi que mes neveux et nièces Quentin DAMBRUNE et Inès DAMBRUNE.

Je dédie ces trois ans de travail à ma marraine Cathy LOZINGOT, présente lors de la soutenance et disparue depuis. Ta joie de vivre et ton amour resterons à jamais gravée dans nos mémoires.

Pour finir, je remercie mon futur mari Morgan BIZOT, d'avoir supporté mes jérémiades pendant trois ans.

Résumé

De nos jours, la conception avant-projet en aéronautique repose majoritairement sur des modèles numériques faisant interagir de nombreuses disciplines visant à évaluer les performances de l'avion. Ces disciplines, comme l'aérodynamique, la structure et la propulsion, sont connectées entre elles afin de prendre en compte leurs interactions. Cela produit un processus d'évaluation des performances de l'avion coûteux en temps de calcul. En effet, une évaluation peut prendre de trente secondes pour les modèles de basse fidélité jusqu'à plusieurs semaines pour les modèles de plus haute fidélité. De plus, à cause de la multi-disciplinarité du processus et de la diversité des outils de calcul, nous n'avons généralement pas accès aux propriétés ou au gradient de cette fonction de performance. En outre, chaque discipline utilise ses propres variables de conception et doit respecter des contraintes d'égalité ou d'inégalité qui sont souvent nombreuses et multi-modales. On cherche finalement à trouver la meilleur configuration possible dans un espace de conception donné.

Cette recherche peut se traduire mathématiquement par un problème d'optimisation boite-noire sous contraintes d'inégalité et d'égalité, aussi connues comme contraintes mixtes, dépendant d'un grand nombre de variables de conception. De plus, les contraintes et la fonction objective sont coûteuses à évaluer et leur régularité n'est pas connue. C'est pourquoi, on s'intéresse aux méthodes d'optimisations sans dérivées et particulièrement celles reposant sur les modèles de substitution. Les méthodes d'optimisation Bayésienne, utilisant des processus gaussiens, sont notamment étudiées car elles ont montré des convergences rapides sur des problèmes multi-modaux. En effet, l'utilisation d'algorithmes d'optimisation évolutionnaire ou reposant sur le gradient n'est pas envisageable du fait du coût de calcul que cela implique : trop d'appels pour générer des populations de points, ou pour approcher le gradient par différences finies.

Cependant la méthode d'optimisation Bayésienne est classiquement utilisée pour des problèmes d'optimisation sans contrainte et de faible dimension. Des extensions ont été proposées pour prendre en compte ce verrou de manière partielle. D'une part, des méthodes d'optimisation ont été introduites pour résoudre des problèmes d'optimisation à contraintes mixtes. Toutefois, aucune d'entre elles n'est adaptable à la grande dimension, aux problèmes multi-modaux et aux contraintes mixtes. D'autre part, des méthodes d'optimisation ont été développées pour la grande dimension pouvant aller jusqu'au million de variables de conception. De même, ces méthodes ne s'étendent que difficilement aux problèmes contraints à cause du temps de calcul qu'ils nécessitent ou de leur caractère aléatoire.

Une première partie de ce travail repose sur le développement d'un algorithme d'optimisation Bayésienne résolvant les problèmes d'optimisation sans contrainte en grande dimension. Il repose sur une stratégie d'apprentissage adaptatif d'un sous-espace linéaire réalisée conjointement à l'optimisation. Ce sous-espace linéaire est ensuite utilisé pour réaliser l'optimisation. Cette mé-

thode a été testée sur des cas tests académiques.

Une deuxième partie de ce travail traite du développement d'un algorithme d'optimisation Bayésienne pour résoudre les problèmes d'optimisation multi-modaux sous contraintes mixtes. Il a été comparé aux algorithmes de la littérature de manière intensive sur une grande batterie de tests académiques.

Finalement, on a confronté le second algorithme à deux cas tests aéronautiques. Le premier cas test est une configuration classique d'avion moyen-courrier à propulsion hybride électrique développé par l'ONERA et l'ISAE-Supaero. Le second cas test est une configuration classique d'avion d'affaire développée par Bombardier Aviation. Ce cas test repose sur une optimisation à deux niveaux de fidélité. Un niveau de fidélité conceptuel et un niveau de fidélité préliminaire pour lesquels le problème est respectivement évalué en trente secondes et 25 minutes. Cette dernière étude a été réalisée lors d'une mobilité internationale chez Bombardier Aviation à Montréal (CA). Les résultats ont montré l'intérêt de la méthode mise en place.

Mots Clefs : Optimisation Bayésienne, Modèle de substitution, Optimisation en grande dimension, Conception avion.

Abstract

Nowadays, the preliminary design in aeronautics is based mainly on numerical models bringing together many disciplines aimed at evaluating the performance of the aircraft. These disciplines, such as aerodynamics, structure and propulsion, are interconnected in order to take into account their interactions. This produces a computationally expensive aircraft performance evaluation process. Indeed, an evaluation can take from thirty seconds for low fidelity models to several weeks for higher fidelity models. In addition, because of the multi-disciplinarity of the process and the diversity of the calculation tools, we do not always have access to the properties or the gradient of this performance function. In addition, each discipline uses its own design variables and must respect equality or inequality constraints which are often numerous and multi-modal. We ultimately seek to find the best possible configuration in a given design space.

This research can be mathematically translated to a black-box optimization problem under inequality and equality constraints, also known as mixted constraints, depending on a large number of design variables. Moreover, the constraints and the objective function are expensive to evaluate and their regularity is not known. This is why we are interested in derivative-free optimization methods and more specifically the ones based on surrogate models. Bayesian optimization methods, using Gaussian processes, are more particularly studied because they have shown rapid convergence on multimodal problems. Indeed, the use of evolutionary optimization algorithms or other gradient-based methods is not possible because of the computational cost that this implies: too many calls to generate populations of points, or to approach the gradient by finite difference.

However, the Bayesian optimization method is conventionally used for optimization problems without constraints and of small dimension. Extensions have been proposed to partially take this lock into account. On the one hand, optimization methods have been introduced to solve optimization problems with mixed constraints. However, none of them is adaptable to the large dimension, to the multi-modal problems and to mixed constraints. On the other hand, non-linear optimization methods have been developed for the large dimension up to a million design variables. In the same way, these methods extend only with difficulty to the constrained problems because of the computing time which they require or their random character.

A first part of this work is based on the development of a Bayesian optimization algorithm solving unconstrained optimization problems in large dimensions. It is based on an adaptive learning strategy of a linear subspace carried out in conjunction with the optimization. This linear subspace is then used to perform the optimization. This method has been tested on academic test cases.

A second part of this work deals with the development of a Bayesian optimization algorithm to solve multi-modal optimization problems under mixed constraints. It has been extensively compared to algorithms in the literature on a large battery of academic tests.

Finally, the second algorithm was compared with two aeronautical test cases. The first test case is a classic medium range aircraft configuration with hybrid electric propulsion developed by ONERA and ISAE-Supaero. The second test case is a classic business aircraft configuration developed at Bombardier Aviation. This test case is based on an optimization at two levels of fidelity. A conceptual fidelity level and a preliminary fidelity level for which the problem is evaluated in thirty seconds and 25 minutes, respectively. This last study was carried out during an international mobility at Bombardier Aviation in Montreal (CA). The results showed the interest of the implemented method.

Key Words : Bayesian optimization, Surrogate models, Large scale optimization, Aircraft design.

Liste des figures

Liste des tableaux

Liste des Algorithmes

Chapitre 1

Introduction

« *Réfléchir n'est pas toujours... rassurant. C'est toujours bien, mais pas toujours rassurant.* »

Robin Hobb

Sommaire

1.1 Cadre

Ce travail de thèse s'est déroulé au sein de l'école doctorale aéronautique et astronautique (ED-AA) en partenariat entre l'Office National d'Etudes et de Recherches Aérospatiales (ONERA) et l'Institut Supérieur de l'Aéronautique et de l'Espace - SUPAERO (ISAE-SUPAERO). Il est dirigé par Nathalie BARTOLI, maître de recherche dans l'unité méthodes multidisciplinaires, concepts intégrés (M2CI) du département traitement de l'information et systèmes (DTIS) de l'ONERA, et co-dirigé par Youssef DIOUANE, professeur associé de l'unité mathématiques appliquées (MA) du département d'ingénierie des systèmes complexes (DISC) de l'ISAE-SUPAERO. Le travail de recherche s'est effectué dans ces deux unités de recherche de l'ONERA et l'ISAE-SUPAERO. Pour finir, quatre mois ont été réalisés dans le département Advanced Design de Bombardier Aviation à Montréal (CA) sous la responsabilité de Hugo GAGNON, ingénieur dans le département Advanced Design, dans le cadre d'une collaboration internationale.

1.2 Contexte et objectif

La conception avion a pour but de proposer de nouvelles configurations d'avion avec les meilleures performances possibles ; par exemple, en minimisant la consommation de carburant ou d'énergie. Cette recherche est cependant soumise à de nombreuses contraintes définies par l'utilisateur,

la compagnie aérienne et/ou le fabricant et à des exigences de sécurité et de sûreté. En général, la performance et les contraintes sont impactées par l'ensemble des disciplines prises en compte dans le processus de conception; par exemple, l'aérodynamique, la structure ou la propulsion. L'ingénieur de conception est ensuite capable, grâce à des méthodes de couplage entre ces disciplines, de produire une configuration avion qui respecte les contraintes tout en fournissant les meilleures performances possibles. De nombreuses possibilités de couplage entre ces disciplines sont envisageables et sont regroupées en grande famille de formulations d'optimisation de conception multi-disciplinaire, ou multi-disciplinary design optimization (MDO). Une démarche similaire peut aussi être appliquée à d'autres processus de conception; par exemple, des hélicoptères, des drones ou des planeurs [99].

Au cours de la dernière décennie, on a tenté d'améliorer l'efficacité des processus de conception avion en développant des outils et des techniques dans le domaines de la MDO [26]. En effet, chaque connexion entre les disciplines représente l'influence d'une discipline sur une autre. On crée donc un problème fortement interdépendant. Suivant la formulation MDO choisie pour représenter ces interdépendances, on peut être amené à ajouter des contraintes, dites de consistance, qui assurent la faisabilité de l'avion et la cohérence entre les disciplines considérées. En résumé, la MDO est un outil performant capable de gérer automatiquement le compromis entre les différentes disciplines tout en prenant en compte les différentes contraintes de conception et de consistance.

La structure, la propulsion, l'aérodynamique, l'automatique et la sécurité sont des exemples de disciplines à prendre en compte. Chacune d'entre elles est accompagnée de ses propres contraintes et variables de conception qui peuvent être soit liées à d'autres disciplines soit directement gérées par l'utilisateur. C'est pourquoi lorsque le nombre de disciplines augmente, le nombre de contraintes et de variables de conception peut, quant à lui, fortement augmenter. De plus, à cause de l'interdépendance des disciplines, les contraintes de conception peuvent définir des zones faisables non-convexes. On les nome contraintes multi-modales. De la même manière, la fonction objectif représentant les performances à optimiser dépend de l'ensemble des disciplines ce qui peut la rendre fortement multi-modale. On peut ainsi obtenir des problèmes MDO avec plus d'une centaine de variables de conception et plus d'une quarantaine de contraintes d'égalité et/ou d'inégalité multi-modales.

Les modèles utilisés pour représenter les disciplines considérées dans le processus sont souvent très gourmands en temps de calcul et ne fournissent pas toujours les dérivées de la fonction objectif et des contraintes [9]. Les algorithmes d'optimisation classiques ne peuvent donc pas être utilisés efficacement puisqu'ils ont besoin soit des dérivées, que l'on ne peut estimer par différences finies au vue du nombre d'appels nécessaires à ce calcul, soit de beaucoup d'évaluations du problème [76].

Les problèmes MDO font finalement partie d'une grande famille de problèmes d'optimisation boite noire, ou black box optimization (BBO) [4]. Cela signifie que l'on n'a aucune connaissance sur les fonctions objectif et les contraintes, leurs propriétés mathématiques (régularité, continuité, linéarité, etc.), leurs dérivées ou leur Hessienne. De plus, on considère que ces problèmes sont coûteux à évaluer en temps de calcul ou en ressource.

Pour résoudre ce type de problèmes, les méthodes d'optimisation sans dérivées ou derivative-free optimization (DFO) [4, 24] peuvent être utilisées dans la majorité des cas. Cependant, ces mé-

thodes résolvent difficilement les problèmes BBO très couteux en temps de calcul et comportant des contraintes multi-modales mixtes. En effet, la plupart de ces méthodes cherchent à résoudre des problèmes comportant uniquement des contraintes d'inégalité et ne prennent pas en considération le coût d'évaluation des problèmes BBO considérés dans ce manuscrit.

Au contraire, les méthodes d'optimisation Bayésienne, ou Bayesian optimization (BO) [71] ont été spécialement développées pour résoudre des problèmes BBO [37, 117] couteux à évaluer en temps ou en ressources. Elles cherchent à utiliser le moins possible d'évaluations grâce à l'emploi de modèles de substitution et plus particulièrement de processus gaussiens, ou Gaussian processes (GP) [98]. Ces méthodes permettent notamment de limiter fortement le nombre d'appels aux problèmes et ainsi de réduire le temps nécessaire à l'optimisation des problèmes considérés.

Au final, l'objectif de cette thèse est de développer et de valider une ou des méthodes BO capables de prendre en compte jusqu'à un millier de variables de conception et plus d'une quarantaine de contraintes multi-modales, c.-à-d. des contraintes définissant une zone faisable non convexe et pouvant être à la fois d'égalité et d'inégalité. On souhaite également que les méthodes proposées soient validées sur de nombreux exemples analytiques et sur des exemples de conception avion avant-projet traités dans l'équipe ou dans le cadre de collaborations.

1.3 Contribution

Durant ces trois années de thèse, on s'est intéressé à la résolution de problèmes BBO ayant $d \geq 100$ variables de conception et comportant p contraintes d'égalité et/ou m contraintes d'inégalité multi-modales et qui peuvent être nombreuses ($m + p \geq 40$). La fonction objectif considérée peut également être fortement multi-modale. Ces problèmes s'écrivent sous la forme mathématique :

$$\min_{\boldsymbol{x}\in\Omega} \left\{ f(\boldsymbol{x}) \;\text{ s.c. }\; \boldsymbol{g}(\boldsymbol{x}) \geq 0 \text{ et } \boldsymbol{h}(\boldsymbol{x}) = 0 \right\}, \tag{1.1}$$

où $f : \mathbb{R}^d \mapsto \mathbb{R}$ est la fonction objectif, $\boldsymbol{g} : \mathbb{R}^d \mapsto \mathbb{R}^m$ sont les contraintes d'inégalité et $\boldsymbol{h} : \mathbb{R}^d \mapsto \mathbb{R}^p$ correspondent aux contraintes d'égalité. L'ensemble des contraintes est QRSK (connu, issu de simulations, relâchable et quantifiable) suivant la taxonomie des contraintes introduite par **?**]. De plus, ces problèmes sont coûteux à évaluer en temps ou en ressource. C'est pourquoi on s'est particulièrement intéressé aux méthodes BO capables de résoudre ce type de problème.

Cependant ces méthodes souffrent de quatre verrous principaux. Tout d'abord, on dénombre deux verrous s'appliquant aux méthodes aptes à prendre en compte un grand nombre de variables de conception.

1. L'optimisation dans l'espace de grande dimension est limitée par une augmentation du temps de calcul avec le nombre de variables de conception du problème. Ceci est notamment dû à la construction de modèles de substitutions ou à la résolution de sous-problèmes d'optimisation nécessaires au processus global de recherche d'optimum.

2. Le verrou 1 est levé par certaines méthodes qui réalisent le processus BO dans un sous-espace de très faible dimension, inférieur à 5 en général. Toutefois, ce sous-espace est généré aléatoirement ce qui ne garantit pas de trouver le minimum de la fonction objectif.

En ce qui concerne les méthodes d'optimisation sous-contraintes, deux autres verrous sont également identifiés.

3. Certaines méthodes ne permettent pas de considérer les contraintes d'égalité.
4. D'autres méthodes restent bloquées dans la première zone faisable découverte. En effet, elles ne cherchent pas à explorer le domaine de conception à la recherche des meilleures zones faisables du domaine, c.-à-d. qui contiennent des valeurs de fonction objectif faibles.

On souligne que les verrous 2 et 4 soulèvent la même problématique d'impossibilité de découvrir le minimum global de la fonction. Toutefois, on considère deux verrous distincts car ils ne sont pas provoqués par les mêmes mécanismes.

Pour lever les verrous 1 et 2, on a proposé une méthode d'optimisation Bayésienne en grande dimension, ou high dimensionnal Bayesian optimization (HDBO) que l'on a validé sur 5 cas tests analytiques comportant 10 à 1000 variables de conception.

Pour lever les verrous 3 et 4, on a développé un algorithme d'optimisation Bayésienne sous contraintes, ou constrained Bayesian optimization (CBO), ne souffrant pas du verrou 1, qui permet de résoudre des problèmes d'optimisation sous-contraintes d'égalité et d'inégalité. Cette dernière méthode a été testée sur un ensemble de 29 problèmes analytiques de la littérature. Grâce à cette base de problèmes, on a réalisé un grand nombre de tests afin d'évaluer la méthode proposée. De plus, on a également testé la méthode sur deux cas tests aéronautiques. Le premier consiste à concevoir un avion hybride à propulsion distribuée électrique développé à l'ONERA et à l'ISAE-SUPAERO. Ce travail a été possible grâce à une collaboration avec Alessandro SGUEGLIA, doctorant dans l'unité M2CI du DTIS de l'ONERA. Le deuxième consiste à concevoir un avion d'affaires de chez Bombardier Aviation. Cette dernière étude a été réalisée lors d'une mobilité internationale de 4 mois à Bombardier Aviation à Montréal (CA) avec la collaboration de Hugo GAGNON et Ian CHITTICK, ingénieurs à Bombardier Aviation. Deux autres collaborations ont été possibles durant ces trois années de thèse dans le cadre du projet européen Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts (AGILE) (2015 - 2018), notamment avec le département de génie mécanique, industriel et aérospatial de l'université de Concordia à Montréal (CA) pour la conception d'un avion hybride électrique à énergie solaire ; et avec le TsAGI (RU) pour la conception d'une nacelle de moteur. Ces deux dernières collaborations ne sont pas présentées dans cette thèse pour éviter les redondances mais ont donné lieu à des communications lors de conférences [54, 91].

1.4 Organisation

Le document se divise en 6 chapitres. Dans le Chapitre 2, une étude bibliographique des méthodes HDBO et CBO est réalisée. Cette étude met en évidence les apports et les manques des méthodes HDBO et CBO actuelles.

Dans le Chapitre 3, on introduit une méthode HDBO reposant sur l'utilisation de sous-espaces de recherche dans lesquels la fonction objectif varie fortement. De plus, on cherche à limiter le temps de calcul nécessaire à ce processus d'optimisation. On évalue ensuite les performances de cette méthode en sélectionnant les paramètres les plus prometteurs et en la comparant aux méthodes de la littérature sur des cas tests académiques.

Dans le Chapitre 4, on propose une méthode CBO qui limite le temps de calcul et prend en compte des contraintes multi-modales qui peuvent être d'égalité et/ou d'inégalité. De la même manière que pour les méthodes HDBO, on évalue l'impact des paramètres de ce nouvel algorithme et on le compare avec les algorithmes de la littérature sur des cas tests analytiques.

Dans le Chapitre 5, on évalue les performances de la méthode CBO proposée sur les problèmes d'optimisation aéronautiques.

Dans le Chapitre 6, on dresse les conclusions et perspectives de ce travail.

1.5 Travaux publiés en lien avec la thèse

Certains des travaux présentés dans cette thèse ont fait l'objet de publications dans des journaux et de communications lors de conférences internationales.

Publications dans des journaux

— PRIEM, R., N. BARTOLI, Y. DIOUANE et A. SGUEGLIA. 2020, «Upper trust bound feasibility criterion for mixed constrained Bayesian optimization with application to aircraft design», Aerospace Science and Technology, vol. 105, p. 105 980 [96].

— BARTOLI, N., T. LEFEBVRE, S. DUBREUIL, R. OLIVANTI, R. PRIEM, N. BONS, J. R. R. A. MARTINS et J. MORLIER. 2019, «Adaptive modeling strategy for constrained global optimization with application to aerodynamic wing design», Aerospace Science and Technology, vol. 90, p. 85–102 [9].

Publications dans des actes

— PRIEM, R., H. GAGNON, I. CHITTICK, S. DUFRESNE, Y. DIOUANE et N. BARTOLI. 2020, «An efficient application of Bayesian optimization to an industrial MDO framework for aircraft design.», dans Proceedings of the AIAA AVIATION 2020 FORUM, VIRTUAL EVENT [97].

— PRIEM, R., N. BARTOLI et Y. DIOUANE. 2019, «On the Use of Upper Trust Bounds in Constrained Bayesian Optimization Infill Criteria», dans Proceedings of the AIAA AVIATION 2019 FORUM, p. 2986 [91].

— JEYARAJ, A., F. SANCHEZ, P. EARNEST, E. MURUGESAN, R. PRIEM et S. LISCOUET-HANKE. 2019, «Exploring collaborative and multidisciplinary aircraft optimization through the AGILE academy challenge – a case study for an aircraft auxiliary solar power system», dans Proceedings of the 19th Astronautics Conference of the Canadian Aeronautics and Space Institute [54].

Communications à des conférences sur résumé

— PRIEM, R., N. BARTOLI, Y. DIOUANE et H. GAGNON. 2019, «On a feasibility criterion for aircraft efficient global multidisciplinary optimization», dans Proceedings of the 1st MDOPhD Day [93].

— PRIEM, R., N. BARTOLI, Y. DIOUANE et A. SGUEGLIA. 2019, «An upper trust bound feasibility criterion for constrained bayesian optimization», dans Proceedings of the Optimization Days 2019 [95].

— PRIEM, R., N. BARTOLI, Y. DIOUANE, S. DUBREUIL et T. LEFEBVRE. 2018, «An adaptive feasibility approach for constrained bayesian optimization with application in aircraft design», dans Proceedings of the 6th International Conference on Engineering Optimization [92].

Communication à des conférences sur poster

— PRIEM, R., N. BARTOLI, Y. DIOUANE, T. LEFEBVRE, S. DUBREUIL, M. SALAÜN et J. MORLIER. 2018, «SEGOMOE : Super efficient global optimization with mixture of experts», dans Proceedings of the CIMI Optimization and Learning Workshop [94].

Prix

— 1er prix du challenge AGILE Academy, Naples, Octobre 2018 [54].

Chapitre 2

Optimisation Bayésienne sous-contraintes et en grande dimension : un état de l'art

« *Il y a toujours des solutions mais, parfois, aucune de satisfaisante.* »

Robin Hobb

Sommaire

Objectifs du chapitre

- Introduire les processus gaussiens et leur extension pour la grande dimension.
- Introduire l'optimisation Bayésienne et ses extensions pour la grande dimension.
- Introduire les méthodes d'optimisation Bayésienne avec contraintes et leur extension pour la grande dimension.

Ce chapitre s'efforce de réaliser l'état de l'art et d'introduire les outils nécessaires à la compréhension et à l'utilisation des méthodes d'optimisation Bayésienne sous-contraintes en grande dimension, ou high dimensionnal constrained Bayesian optimization (HDCBO). Il est divisé en trois grandes parties. On commence par introduire la théorie de l'interpolation par GP et les instruments requis pour cette construction. Dans cette partie, les méthodes d'interpolation par GP pour la grande dimension sont présentées. Ensuite, la structure de la méthode de BO sans contrainte est développée. En outre, on discute des principales méthodes de la littérature ainsi que de leurs extensions à la grande dimension. Enfin, on s'intéresse aux différentes méthodes BO prenant en compte des contraintes et on étend cet intérêt aux méthodes de HDCBO.

2.1 Généralité sur l'interpolation par processus gaussiens

L'interpolation par GP à sortie scalaire [36, 61, 98] a d'abord été introduite par KRIGE [61] dans le cadre de l'exploitation minière. Les GP sont maintenant largement utilisés grâce aux nouveaux moyens informatiques notamment dans la communauté de l'apprentissage automatique, ou machine learning (ML) et de l'intelligence artificielle, ou artificial intelligence (AI) [98] ainsi qu'en ingénierie pour la MDO [36].

Un GP est entièrement défini par une fonction de moyenne a priori et un noyau de covariance. La fonction de moyenne a priori décrit le comportement global du GP alors que le noyau de covariance représente la similitude entre deux points d'échantillonnage du domaine de conception.

2.1.1 Formulation des processus Gaussiens

Une description détaillée de la méthode d'interpolation par GP à sortie scalaire est donnée ci-après. Par abus de langage, on dira GP à la place d'interpolation par GP.

On cherche ici à approcher une fonction scalaire $s : \mathbb{R}^d \mapsto \mathbb{R}$ en utilisant les l points du plan d'expériences, ou design of experiments (DoE) $\mathcal{D}_s^{(l)} = \left\{\boldsymbol{x}^{(k)}, y_s^{(k)}\right\}_{k=1,\dots,l}$ où $\boldsymbol{x}^{(k)} \in \Omega$, $\Omega \subset \mathbb{R}^d$ et $y_s^{(k)} = s\left(\boldsymbol{x}^{(k)}\right) \in \mathbb{R}$. De plus, on suppose que la fonction scalaire s est la réalisation d'un GP, noté $\mathcal{GP}\left(\mu, k\right)$, de moyenne a priori $\mu : \mathbb{R}^d \mapsto \mathbb{R}$ et de noyau de covariance $k : \mathbb{R}^{d\times 2} \mapsto \mathbb{R}$. On conditionne ensuite le GP pour qu'il prenne en compte les données connues de s en utilisant $\mathcal{D}_s^{(l)}$. Ce GP conditionné par les l points d'échantillonnage $\mathcal{D}_s^{(l)}$ définit une variable aléatoire $y_{s,\boldsymbol{x}}^{(l)}$ pour chaque point $\boldsymbol{x}$ de Ω qui suit une loi gaussienne

$$y_{s,\boldsymbol{x}}^{(l)} \sim \mathbb{P}\left(y_s | \mathcal{D}_s^{(l)}, \boldsymbol{x}\right) = \mathcal{GP}\left(\mu, k | \mathcal{D}_s^{(l)}, \boldsymbol{x}\right) = \mathcal{N}\left(\hat{\mu}_s^{(l)}(\boldsymbol{x}), \left[\hat{\sigma}_s^{(l)}(\boldsymbol{x})\right]^2\right) \tag{2.1}$$

et dont $p\left(y_s | \mathcal{D}_s^{(l)}, \boldsymbol{x}\right)$ est la fonction de densité. La moyenne $\hat{\mu}_s^{(l)}$ et l'écart type $\hat{\sigma}_s^{(l)}$ s'expriment

comme suit :

$$\hat{\mu}_s^{(l)}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \boldsymbol{k}^{(l)}(\boldsymbol{x})^\top \left[\mathbf{K}^{(l)}\right]^{-1} \left(\mathbf{Y}_s^{(l)} - \boldsymbol{\mu}^{(l)}\right), \tag{2.2}$$

$$\hat{\sigma}_s^{(l)}(\boldsymbol{x}) = \left(k(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{k}^{(l)}(\boldsymbol{x})^\top \left[\mathbf{K}^{(l)}\right]^{-1} \boldsymbol{k}^{(l)}(\boldsymbol{x})\right)^{\frac{1}{2}}, \tag{2.3}$$

où $\boldsymbol{\mu}^{(l)} = \left[\mu(\boldsymbol{x}^{(1)}),\ldots,\mu(\boldsymbol{x}^{(l)})\right]^\top \in \mathbb{R}^l$ est le vecteur de moyenne pour les points d'échantillonnage de $\mathcal{D}_s^{(l)}$, $\boldsymbol{k}^{(l)}(\boldsymbol{x}) = \left[k(\boldsymbol{x}^{(1)},\boldsymbol{x}),\ldots,k(\boldsymbol{x}^{(l)},\boldsymbol{x})\right]^\top \in \mathbb{R}^l$ est le vecteur de covariance entre $\boldsymbol{x}$ et les points d'échantillonnage de $\mathcal{D}_s^{(l)}$, $\mathbf{K}^{(l)} = \left[k(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)})\right]_{i,j=1,\ldots,l} \in \mathbb{R}^{l\times l}$ est la matrice de covariance évaluée sur tous les points d'échantillonnage de $\mathcal{D}_s^{(l)}$. Enfin, $\mathbf{Y}_s^{(l)} = \left[y_s^{(1)},\ldots,y_s^{(l)}\right]^\top \in \mathbb{R}^l$ est le vecteur des sorties de s.

Pour finir, la Figure 2.1 représente le GP de la fonction $s(\boldsymbol{x}) = \boldsymbol{x}\sin(3\pi(\boldsymbol{x}+0.1))$, construit à partir d'un DoE de trois points, pour lequel on a tracé la moyenne et l'écart type sur tout le domaine de définition $\Omega = [0,1]$. On a aussi tracé la distribution normale définie par le GP au point $\boldsymbol{x} = 0$. La Figure 2.1 et l'équation (2.3) montrent que la moyenne du GP interpole les points du DoE. De

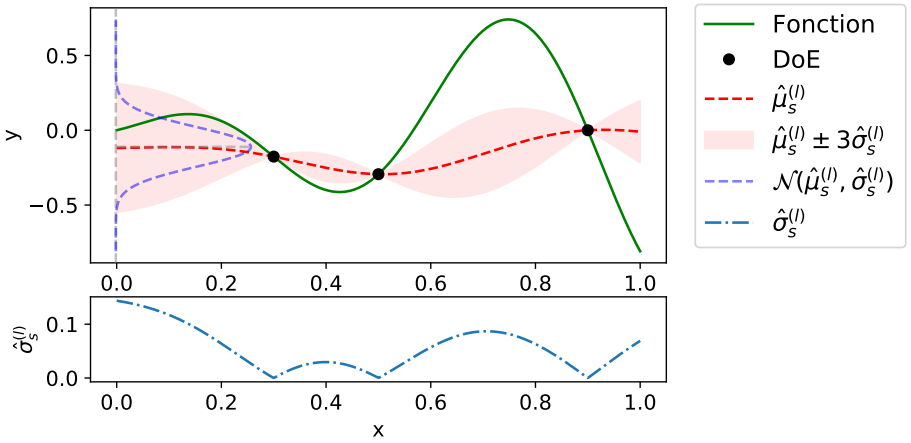


FIGURE 2.1 – Illustration d'un modèle de GP de la fonction $s(\boldsymbol{x}) = \boldsymbol{x}\sin(3\pi(\boldsymbol{x}+0.1))$.

plus, la valeur de l'écart type est nulle en ces points. On considère donc que la moyenne $\hat{\mu}_s^{(l)}(\boldsymbol{x})$ est la prédiction du modèle de régression associé au GP de la fonction s et que l'écart type $\hat{\sigma}_s^{(l)}(\boldsymbol{x})$ représente une estimation de l'erreur locale de la prédiction au point $\boldsymbol{x}$ du modèle de régression.

Remarque 2.1.1 *On souligne que dans l'hypothèse où la fonction à modéliser s est la réalisation d'un GP dont on connaît les hyper-paramètres, le noyau de covariance et la moyenne a priori, alors il y a 99% de chance que la valeur au point $\boldsymbol{x}$ de la fonction à modéliser soit comprise dans l'intervalle $\left[\hat{\mu}_s^{(l)}(\boldsymbol{x}) - 3\hat{\sigma}_s^{(l)}(\boldsymbol{x}),\ \hat{\mu}_s^{(l)}(\boldsymbol{x}) + 3\hat{\sigma}_s^{(l)}(\boldsymbol{x})\right]$. Dans le cas où on ne peut pas vérifier ces hypothèses, cet intervalle perd la valeur connue de probabilité d'appartenance. En pratique, on considère que cet intervalle contient avec une forte probabilité la valeur de la fonction à modéliser même si l'hypothèse initiale n'est pas vérifiable.*

2.1.2 Noyaux de Covariance

Les noyaux de covariance permettent de mesurer la similitude entre deux points $\boldsymbol{x}$ et $\boldsymbol{x}'$ du domaine Ω [98]. Ils peuvent être de deux types : soit stationnaires, c.-à-d. qu'ils sont invariants

par translation, soit non-stationnaires. Pour qu'une application $k : \mathbb{R}^{d\times 2} \mapsto \mathbb{R}$ soit un noyau de covariance, il est nécessaire et suffisant qu'elle soit symétrique semi-définie positive [98]. Ceci se traduit mathématiquement par :

— **Symétrie :** $\forall \boldsymbol{x}, \boldsymbol{x}' \in \Omega, k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$

— **Positivité :** $\forall \eta_0 \in \mathbb{N}^*, \forall \lambda_1, \ldots, \lambda_{\eta_0}, \forall \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\eta_0)} \in \Omega, \sum_{i=1}^{\eta_0} \sum_{j=1}^{\eta_0} \lambda_i \lambda_j k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) \geq 0$

Le choix du noyau de covariance est souvent fait a priori ou par des méthodes de validation croisée [32, 98] parmi un ensemble varié de possibilités [32].

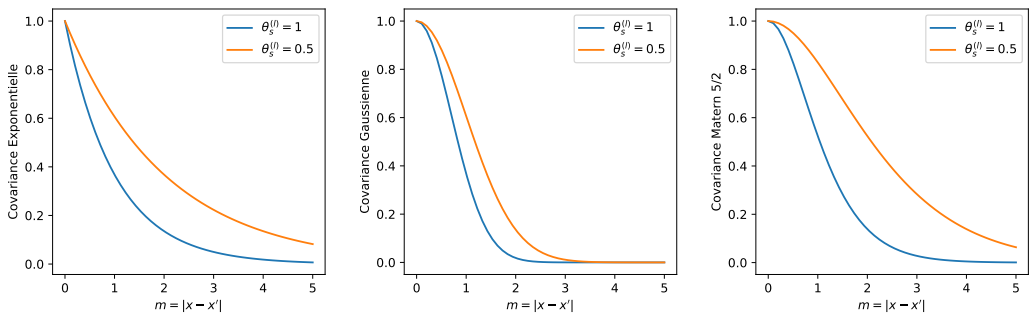
Dans les sections suivantes, on introduit les principaux noyaux de covariance utilisés ainsi que des méthodes pour en créer de nouveaux à partir de noyaux connus.

2.1.2.1 Exemple de noyaux de covariance

Il existe de nombreux noyaux de covariance [32] qui permettent de représenter une grande variété de fonctions. Les trois noyaux principalement utilisés sont les noyaux exponentiels, gaussiens et Matérn $\frac{5}{2}$. Ils sont caractérisés par $N_p \in \mathbb{N}^+$ hyper-paramètres $\boldsymbol{\theta}_s^{(l)} \in \left[\mathbb{R}^+\right]^{N_p}$ qui codent la similitude entre deux points pour chaque dimension du domaine Ω. Leurs expressions ainsi que leurs nombres d'hyper-paramètres sont donnés dans le Tableau 2.1. La Figure 2.2 représente la covariance unidimensionnelle pour ces trois noyaux. On constate que la longueur de corrélation est plus faible lorsque $\boldsymbol{\theta}_s^{(l)}$ est grand. En effet, la longueur de corrélation $l_{s,i}^{(l)}$ pour chaque dimension

TABLEAU 2.1 – Exemple de noyaux de covariance où $m_i = \left|x_i - x_i'\right|$ représente la i^e composante de $\boldsymbol{m} = \left|\boldsymbol{x} - \boldsymbol{x}'\right|$ et $\theta_{s,i}^{(l)}$ représente la i^e composante du vecteur d'hyper-paramètres $\boldsymbol{\theta}_s^{(l)}$

Noyau de covariance	Expression	Nombre d'hyper-paramètres
Exponentiel	$\left[\theta_{s,0}^{(l)}\right]^2 \prod_{i=1}^{d} \exp\left(-\theta_{s,i}^{(l)} m_i\right)$	$d+1$
Gaussien	$\left[\theta_{s,0}^{(l)}\right]^2 \prod_{i=1}^{d} \exp\left(-\theta_{s,i}^{(l)} m_i^2\right)$	$d+1$
Matérn $\frac{5}{2}$	$\left[\theta_{s,0}^{(l)}\right]^2 \prod_{i=1}^{d} \left(1+\sqrt{5}\theta_{s,i}^{(l)} m_i + \frac{5}{3}\left[\theta_{s,i}^{(l)} m_i\right]^2\right) \exp\left(-\sqrt{5}\theta_{s,i}^{(l)} m_i\right)$	$d+1$



(a) Exponentiel. (b) Gaussienne. (c) Matérn $\frac{5}{2}$.

FIGURE 2.2 – Noyaux de covariance classiques unidimensionnels en fonction de la distance entre deux points.

i est directement reliée à l'hyper-paramètre $\theta_{s,i}^{(l)}$ par la relation suivante :

$$l_{s,i}^{(l)} = \frac{1}{\sqrt{2\theta_{s,i}^{(l)}}}. \tag{2.4}$$

Ainsi, il existe une longueur de corrélation, c.-à-d. un hyper-paramètre, qui représente au mieux la similitude entre les points. Cette étape d'estimation des meilleurs hyper-paramètres est présentée dans la Section 2.1.3.

2.1.2.2 Création de nouveaux noyaux de covariance

A partir des noyaux classiques [32, 98], il est possible de créer de nouveaux noyaux pour combiner leurs propriétés. Il existe de nombreuses manières de combiner des noyaux existants. Les plus élémentaires sont les suivantes :

— la somme de deux noyaux $k_1 : \mathbb{R}^{d\times 2} \mapsto \mathbb{R}$ et $k_2 : \mathbb{R}^{d\times 2} \mapsto \mathbb{R}$.

$$k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}') + k_2(\boldsymbol{x}, \boldsymbol{x}'). \tag{2.5}$$

— le produit de deux noyaux $k_1 : \mathbb{R}^{d\times 2} \mapsto \mathbb{R}$ et $k_2 : \mathbb{R}^{d\times 2} \mapsto \mathbb{R}$

$$k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}') \times k_2(\boldsymbol{x}, \boldsymbol{x}'), \tag{2.6}$$

ou le produit de deux noyaux $k_1 : \mathbb{R}^{d\times 2} \mapsto \mathbb{R}$ et $k_2 : \mathbb{R}^{d'\times 2} \mapsto \mathbb{R}$

$$k(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{u}, \boldsymbol{u}') = k_1(\boldsymbol{x}, \boldsymbol{x}') \times k_2(\boldsymbol{u}, \boldsymbol{u}'). \tag{2.7}$$

— le produit d'un noyau $k_1 : \mathbb{R}^{d\times 2} \mapsto \mathbb{R}$ par une fonction $h : \mathbb{R}^d \mapsto \mathbb{R}$.

$$k(\boldsymbol{x}, \boldsymbol{x}') = h(\boldsymbol{x}) k_1(\boldsymbol{x}, \boldsymbol{x}') h(\boldsymbol{x}'). \tag{2.8}$$

Ces différentes méthodes de combinaison de noyaux de covariance permettent de coupler leurs propriétés afin de mieux représenter la fonction à modéliser.

2.1.3 Estimation des hyper-paramètres

Les GP sont paramétrés par les inconnues $\boldsymbol{\theta}_s^{(l)}$ qui ont été supposées connues dans les Sections 2.1.1 et 2.1.2. Les méthodes de l'estimateur du maximum de vraisemblance, de validation croisée et de maximum a posteriori sont les trois principalement utilisées pour estimer ces hyper-paramètres [5, 44]. Dans ce travail, on considère uniquement la méthode de l'estimateur du maximum de vraisemblance.

2.1.3.1 La fonction de log-vraisemblance

Dans cette section, la fonction de log-vraisemblance est introduite dans le cadre des GP. Cette fonction à maximiser suivant les $\boldsymbol{\theta}_s^{(l)}$ s'exprime analytiquement par

$$\mathcal{L}\left(\mathbf{Y}_s^{(l)}, \boldsymbol{\theta}_s^{(l)}\right) = \frac{1}{(2\pi)^{\frac{l}{2}} \left|\mathbf{K}^{(l)}\right|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\left(\mathbf{Y}_s^{(l)} - \boldsymbol{\mu}^{(l)}\right)^\top \left[\mathbf{K}^{(l)}\right]^{-1} \left(\mathbf{Y}_s^{(l)} - \boldsymbol{\mu}^{(l)}\right)\right). \tag{2.9}$$

De manière équivalente, on peut minimiser l'opposé de la log-vraisemblance fournie ci-dessous :

$$-2\log\left(\mathcal{L}\left(\mathbf{Y}_s^{(l)},\boldsymbol{\theta}_s^{(l)}\right)\right)=\left(\mathbf{Y}_s^{(l)}-\boldsymbol{\mu}^{(l)}\right)^\top\left[\mathbf{K}^{(l)}\right]^{-1}\left(\mathbf{Y}_s^{(l)}-\boldsymbol{\mu}^{(l)}\right)+\log\left|\mathbf{K}^{(l)}\right|+l\log(2\pi). \tag{2.10}$$

En effet, on préfère utiliser l'opposé de la log-vraisemblance car elle réduit les fortes variations qui apparaissent souvent dans la fonction de vraisemblance. Ainsi, son optimisation est grandement facilitée.

2.1.3.2 Optimisation de la fonction de log-vraisemblance

Dans cette section, on introduit l'estimation des hyper-paramètres réalisée grâce à l'optimisation de la fonction de log-vraisemblance (2.10). Les hyper-paramètres $\boldsymbol{\theta}_s^{(l)}$ qui maximisent la fonction de log-vraisemblance ne peuvent pas être calculés analytiquement à cause de l'inverse de la matrice de covariance $\mathbf{K}^{(l)}$. Pour cela, on peut utiliser différentes méthodes d'optimisation classiques pour estimer ces hyper-paramètres. BOUHLEL et al. [19] utilisent l'algorithme d'optimisation sans dérivées constrained optimization by linear approximation (COBYLA) [90] dans surrogate modeling toolbox (SMT) ; ROUSTANT et al. [105] utilisent un algorithme de type limited memory Broyden-Fletcher-Goldfarb-Shanno method subject to bounds (L-BFGS-B) [141] ou un algorithme génétique utilisant l'information des dérivées; TOAL et al. [127] ont développé une méthode couplant l'optimisation par essaims particulaires avec une recherche locale à base de gradient. Cependant, le coût de l'optimisation de la fonction de log-vraisemblance augmente avec la dimension [127, 128]. L'utilisation des dérivées devient donc nécessaire pour accélérer l'optimisation. Celui-ci s'exprime comme suit :

$$-2\frac{\partial\log\left(\mathcal{L}\left(\mathbf{Y}_s^{(l)},\boldsymbol{\theta}_s^{(l)}\right)\right)}{\partial\theta_{s,i}^{(l)}}=-\frac{\partial\left[\boldsymbol{\mu}^{(l)}\right]^\top}{\partial\theta_{s,i}^{(l)}}\Gamma_s^{(l)}-\left[\Gamma_s^{(l)}\right]^\top\frac{\partial\boldsymbol{\mu}^{(l)}}{\partial\theta_{s,i}^{(l)}}-\left[\Gamma_s^{(l)}\right]^\top\frac{\partial\mathbf{K}^{(l)}}{\partial\theta_{s,i}^{(l)}}\Gamma_s^{(l)}+\mathrm{Tr}\left(\left[\mathbf{K}^{(l)}\right]^{-1}\frac{\partial\mathbf{K}^{(l)}}{\partial\theta_{s,i}^{(l)}}\right), \tag{2.11}$$

avec $\frac{\partial\cdot}{\partial\theta_{s,i}^{(l)}}$ la dérivé partielle par rapport à la i$^{\text{e}}$ composante de $\boldsymbol{\theta}_s^{(l)}$, $\Gamma_s^{(l)}=\left[\mathbf{K}^{(l)}\right]^{-1}\left(\mathbf{Y}_s^{(l)}-\boldsymbol{\mu}^{(l)}\right)$ et $\mathrm{Tr}:\mathbb{R}^{l\times l}\longmapsto\mathbb{R}$ l'application trace.

Dans cette section, les principes généraux pour la construction des GP ont été introduits. Ont notamment été introduites les formules de moyenne (2.2) et d'écart type (2.3) des GP; la fonction de log-vraisemblance (2.10) ainsi que ses dérivés partielles (2.11). Toutefois, lorsque la dimension augmente ($d\approx 20$), les GP classiques commencent à souffrir de plusieurs maux.

— GARNETT et al. [40] soulignent que l'utilisation des GP classiques en grande dimension ne permet pas de corréler les informations des points du DoE car la longueur de corrélation entre les points devient trop importante.

— BOUHLEL et al. [17] remarquent que le nombre d'hyper-paramètres à estimer augmente fortement le temps d'optimisation à cause de l'inversion de la matrice de covariance.

Dans la section suivante, on s'intéressera donc aux méthodes mises en place pour étendre les GP à la grande dimension.

2.1.4 Processus Gaussiens pour la grande dimension

De nombreuses méthodes ont été mises en place pour gérer la grande dimension dans le contexte des GP [17, 20, 21, 27, 32, 40, 53, 67, 77, 120, 122]. Pour la plupart, elles reposent sur la

définition de nouveaux types de noyaux de covariance qui permettent de mieux corréler les points du domaine entre eux. D'autres cherchent à réduire le nombre d'hyper-paramètres pour accélérer leur estimation.

Dans la suite de cette section, on introduit les deux méthodes de construction de GP en grande dimension [17, 40] étudiées dans ce manuscrit. Elles reposent sur l'hypothèse que la plupart des fonctions de grande dimension dépendent uniquement de $d_e < d$ directions que l'on appelle directions efficaces. Ceci signifie qu'il existe une fonction $s_{\mathcal{A}} : \mathbb{R}^{d_e} \mapsto \mathbb{R}$ telle que $s_{\mathcal{A}}(\mathbf{A}\boldsymbol{x}) = s(\boldsymbol{x})$ avec $\mathcal{A} \subset \mathbb{R}^{d_e}$, $\mathcal{A} = \{\boldsymbol{u} = \mathbf{A}\boldsymbol{x}, \forall \boldsymbol{x} \in \Omega\}$ et $\mathbf{A} \in \mathbb{R}^{d_e \times d}$. Cette opération permet de transformer l'espace Ω pour ne s'intéresser qu'aux directions de variation de la fonction. Par exemple, la fonction de deux variables de conception $s(\boldsymbol{x}) = 3(x_1 + x_2 + 3) + 24(x_1 + x_2 + 3)^2$, représentée par la Figure 2.3, comporte un sous-espace linéaire unidimensionnel $\mathcal{A}$ qui peut se définir avec la matrice $\mathbf{A} = [1, 1]$. Ce sous-

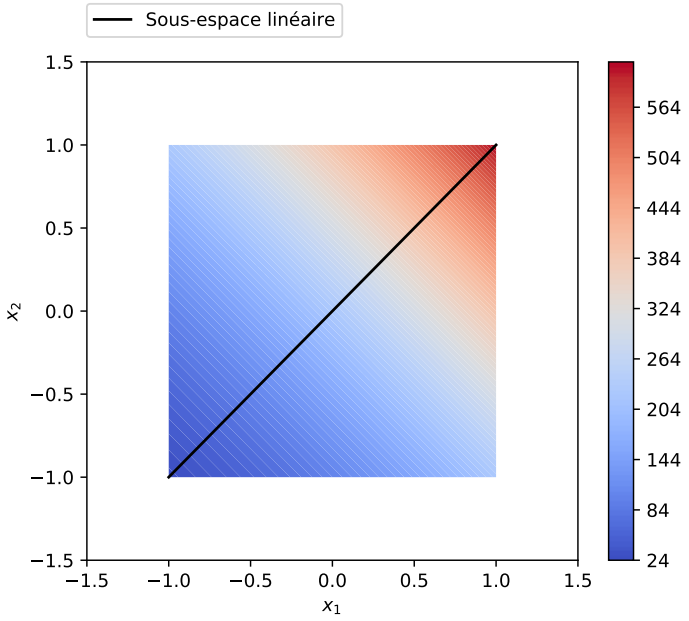


FIGURE 2.3 – Sous-espace linéaire de la fonction $s(\boldsymbol{x}) = 3(x_1 + x_2 + 3) + 24(x_1 + x_2 + 3)^2$.

espace linéaire $\mathcal{A}$ est représenté par la ligne noire sur la Figure 2.3.

On peut ensuite définir un GP pour $s_{\mathcal{A}}$ avec une moyenne a priori $\mu_{\mathcal{A}} : \mathbb{R}^{d_e} \mapsto \mathbb{R}$ et un noyau de covariance $k_{\mathcal{A}} : \mathbb{R}^{d_e \times 2} \mapsto \mathbb{R}$ où $\mu_{\mathcal{A}}(\mathbf{A}\boldsymbol{x}) = \mu(\boldsymbol{x})$ et $k_{\mathcal{A}}(\mathbf{A}\boldsymbol{x}, \mathbf{A}\boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}')$ pour tout $(\boldsymbol{x}, \boldsymbol{x}') \in \Omega^2$. Un GP est donc aussi défini pour la fonction s. On prend, par exemple, un noyau de covariance gaussien dans le sous-espace linéaire $\mathcal{A}$. Il s'écrit :

$$k_{\mathcal{A}}(\boldsymbol{u}, \boldsymbol{u}') = \left[\theta_{s,0}^{(l)}\right]^2 \exp\left(-\frac{1}{2}(\boldsymbol{u} - \boldsymbol{u}')^\top (\boldsymbol{u} - \boldsymbol{u}')\right), \tag{2.12}$$

avec $(\boldsymbol{u}, \boldsymbol{u}') \in \mathcal{A}^2$. Si on considère que $\boldsymbol{u} = \mathbf{A}\boldsymbol{x}$ and $\boldsymbol{u}' = \mathbf{A}\boldsymbol{x}'$, le noyau de covariance précédemment introduit dans $\mathcal{A}$ se traduit par le noyau de covariance suivant dans Ω^2 :

$$k(\boldsymbol{x}, \boldsymbol{x}') = \left[\theta_{s,0}^{(l)}\right]^2 \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^\top \mathbf{A}^\top \mathbf{A}(\boldsymbol{x} - \boldsymbol{x}')\right). \tag{2.13}$$

Dans les deux prochaines sections, on s'intéresse plus particulièrement à deux types de GP utilisant une réduction de dimension : les processus gaussien marginal, ou marginal Gaussian

process (MGP) et les modèles Krigeage combiné aux moindres carrés partiels, ou kriging coupled with partial least squares (KPLS).

2.1.4.1 Processus Gaussiens marginaux

L'idée des MGP [40] est de considérer la matrice vectorisée $\text{vect}(\mathbf{A}) \in \mathbb{R}^{d \cdot d_e}$ comme une réalisation d'une loi gaussienne $\mathbb{P}(\mathrm{A}) = \mathcal{N}\left(\text{vect}\left(\mathbf{A}_p\right), \mathbf{\Sigma}_p\right)$ où $\text{vect}\left(\mathbf{A}_p\right)$ et $\mathbf{\Sigma}_p$ sont respectivement la moyenne et la matrice de covariance a priori. On note $p(\mathbf{A})$ la fonction de densité de $\mathbb{P}(\mathrm{A})$. On cherche ensuite à estimer $\mathbb{P}\left(\mathbf{A}|\mathcal{D}_s^{(l)}\right)$, la loi de probabilité a posteriori de $\text{vect}(\mathbf{A})$ sachant $\mathcal{D}_s^{(l)}$. Pour cela, l'approximation de Laplace peut être utilisée et donne l'approximation gaussienne suivante :

$$\mathbb{P}\left(\mathbf{A}|\mathcal{D}_s^{(l)}\right) \approx \mathcal{N}\left(\text{vect}(\hat{\mathbf{A}}), \hat{\mathbf{\Sigma}}\right), \tag{2.14}$$

$$p\left(\mathbf{A}|\mathcal{D}_s^{(l)}\right) \approx p(\mathbf{A}) \cdot \mathcal{L}\left(\mathbf{Y}_s^{(l)}, \mathbf{A}\right), \tag{2.15}$$

où $\text{vect}\left(\hat{\mathbf{A}}\right)$ est un maximum local de $p\left(\mathbf{A}|\mathcal{D}_s^{(l)}\right)$, la fonction de densité de $\mathbb{P}\left(\mathbf{A}|\mathcal{D}_s^{(l)}\right)$. La matrice de covariance $\hat{\mathbf{\Sigma}}$ est, quant à elle, estimée par l'inverse de la Hessienne de la fonction du logarithme de $p\left(\mathbf{A}|\mathcal{D}_s^{(l)}\right)$ évaluée en $\text{vect}\left(\hat{\mathbf{A}}\right)$

$$\hat{\mathbf{\Sigma}}^{-1} = -\nabla^2 \log p\left(\mathbf{A}|\mathcal{D}_s^{(l)}\right)\Big|_{\mathbf{A}=\hat{\mathbf{A}}} \approx -\nabla^2 \log \mathcal{L}\left(\mathbf{Y}_s^{(l)}, \mathbf{A}\right)\Big|_{\mathbf{A}=\hat{\mathbf{A}}} - \nabla^2 \log p\left(\mathbf{A}\right)\Big|_{\mathbf{A}=\hat{\mathbf{A}}} \tag{2.16}$$

où ∇^2 correspond à l'opérateur Hessienne par rapport à $\text{vect}(\mathbf{A})$. Pour calculer le log a posteriori, il est nécessaire de connaître le log a priori ainsi que la log vraisemblance. Bien que les dérivées et la Hessienne du log a priori soient triviaux, ceux de la log vraisemblance le sont moins. Les dérivées de la log vraisemblance a déjà été introduit précédemment par (2.11) tandis que la Hessienne est donnée par :

$$\begin{aligned}
-2\frac{\partial^2 \log\left(\mathcal{L}\left(\mathbf{Y}_s^{(l)}, \boldsymbol{\theta}_s^{(l)}\right)\right)}{\partial\theta_{s,j}^{(l)}\partial\theta_{s,i}^{(l)}} = {} & 2\frac{\partial\left[\boldsymbol{\mu}^{(l)}\right]^\top}{\partial\theta_{s,i}^{(l)}}\left[\mathbf{K}^{(l)}\right]^{-1}\frac{\partial\boldsymbol{\mu}^{(l)}}{\partial\theta_{s,j}^{(l)}} - 2\left[\boldsymbol{\Gamma}_s^{(l)}\right]^\top\frac{\partial^2\boldsymbol{\mu}^{(l)}}{\partial\theta_{s,j}^{(l)}\partial\theta_{s,i}^{(l)}} \\
& + 2\left[\boldsymbol{\Gamma}_s^{(l)}\right]^\top\left[\frac{\partial\mathbf{K}^{(l)}}{\partial\theta_{s,i}^{(l)}}\left[\mathbf{K}^{(l)}\right]^{-1}\frac{\partial\boldsymbol{\mu}^{(l)}}{\partial\theta_{s,j}^{(l)}} + \frac{\partial\mathbf{K}^{(l)}}{\partial\theta_{s,j}^{(l)}}\left[\mathbf{K}^{(l)}\right]^{-1}\frac{\partial\boldsymbol{\mu}^{(l)}}{\partial\theta_{s,i}^{(l)}}\right] \\
& + \left[\boldsymbol{\Gamma}_s^{(l)}\right]^\top\left[-\frac{\partial^2\mathbf{K}^{(l)}}{\partial\theta_{s,j}^{(l)}\partial\theta_{s,i}^{(l)}} + 2\frac{\partial\mathbf{K}^{(l)}}{\partial\theta_{s,i}^{(l)}}\left[\mathbf{K}^{(l)}\right]^{-1}\frac{\partial\mathbf{K}^{(l)}}{\partial\theta_{s,j}^{(l)}}\right]\boldsymbol{\Gamma}_s^{(l)} \\
& - \text{Tr}\left(\left[\mathbf{K}^{(l)}\right]^{-1}\frac{\partial\mathbf{K}^{(l)}}{\partial\theta_{s,j}^{(l)}}\left[\mathbf{K}^{(l)}\right]^{-1}\frac{\partial\mathbf{K}^{(l)}}{\partial\theta_{s,i}^{(l)}}\right) + \text{Tr}\left(\left[\mathbf{K}^{(l)}\right]^{-1}\frac{\partial^2\mathbf{K}^{(l)}}{\partial\theta_{s,j}^{(l)}\partial\theta_{s,i}^{(l)}}\right),
\end{aligned} \tag{2.17}$$

où on a noté $\boldsymbol{\theta}_s^{(l)} = \text{vect}(\mathbf{A})$ pour faciliter l'écriture, $\boldsymbol{\Gamma}_s^{(l)} = \left[\mathbf{K}^{(l)}\right]^{-1}\left(\mathbf{Y}_s^{(l)} - \boldsymbol{\mu}^{(l)}\right)$ et $\theta_{s,i}^{(l)}$ et la i$^{\text{e}}$ composante de $\boldsymbol{\theta}_s^{(l)}$.

Maintenant, on peut prendre en compte la distribution de $\boldsymbol{\theta}_s^{(l)} = \text{vect}(\mathbf{A})$ dans la construction du processus stochastique représentant la fonction s. La fonction de densité du processus stochastique de s sachant $\mathcal{D}_s^{(l)}$ et $\boldsymbol{x}$, $p\left(y_s|\mathcal{D}_s^{(l)}, \boldsymbol{x}\right)$, et n'admet pas d'expression analytique. GARNETT et al. [40] propose donc une marginalisation sur $\boldsymbol{\theta}_s^{(l)}$ et une série d'approximations afin d'exprimer analytiquement la moyenne $\tilde{\mu}_s^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}$ et l'écart type $\tilde{\sigma}_s^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}$ de $\mathbb{P}\left(y_s|\mathcal{D}_s^{(l)}, \boldsymbol{x}\right)$. Notamment,

on approche la loi de probabilité $\mathbb{P}\left(y_s|\mathcal{D}_s^{(l)}, \boldsymbol{\theta}_s^{(l)}, \boldsymbol{x}\right)$ par une loi gaussienne dont la moyenne dépend linéairement des hyper-paramètres $\boldsymbol{\theta}_s^{(l)}$. Les paramètres de cette loi gaussienne sont ensuite estimés grâce à une expansion locale. Grâce à ces deux étapes succinctement introduites ici, on obtient :

$$\tilde{\mu}_s^{(l)} = \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}, \tag{2.18}$$

$$\left[\tilde{\sigma}_s^{(l)}\right]^2 = \tilde{v}_s^{(l)} = \frac{4}{3}\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} + \left[\nabla\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}\right]^\top \hat{\boldsymbol{\Sigma}} \nabla\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} + \frac{1}{3\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}} \left[\nabla\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}\right]^\top \hat{\boldsymbol{\Sigma}} \nabla\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}, \tag{2.19}$$

où $\hat{\boldsymbol{\Sigma}}$ est donnée par (2.3), $\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} = [\hat{\sigma}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}]^2$, $\hat{\boldsymbol{\theta}}_s^{(l)} = \text{vect}\left(\hat{\mathbf{A}}\right)$, $\nabla\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} : \mathbb{R}^d \mapsto \mathbb{R}^{d_e \cdot d}$ et $\nabla\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} : \mathbb{R}^d \mapsto \mathbb{R}^{d_e \cdot d}$ sont respectivement les dérivées par rapport à $\boldsymbol{\theta}_s^{(l)}$ évalués en $\hat{\boldsymbol{\theta}}_s^{(l)}$ de la moyenne et de la variance du GP de s construit avec les hyper-paramètres $\boldsymbol{\theta}_s^{(l)}$. Ces dérivées ont des expressions analytiques qui dépendent du noyau de covariance et de la moyenne a priori choisis par l'utilisateur. De plus, $\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} : \mathbb{R}^d \mapsto \mathbb{R}$ et $\hat{\sigma}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} : \mathbb{R}^d \mapsto \mathbb{R}$ sont respectivement la moyenne et l'écart type du GP de s construit avec les hyper-paramètres $\hat{\boldsymbol{\theta}}_s^{(l)}$ (voir (2.2) et (2.3)). Les détails des calculs nécessaires à l'obtention des expressions de $\tilde{\mu}_s^{(l)}$ et $\tilde{\sigma}_s^{(l)}$ sont donnés en Annexe A.1.

En pratique, le processus de construction d'un MGP est donné par l'Algorithme 2.1.

Algorithme 2.1 Processus de construction d'un MGP.

entrée : le DoE initial, la moyenne a priori des hyper-paramètres vect$\left(\mathbf{A}_p\right)$, la covariance des hyper-paramètres $\boldsymbol{\Sigma}_p$

1: Maximiser la vraisemblance de vect$(\mathbf{A})$ en utilisant (2.11) pour trouver vect$\left(\hat{\mathbf{A}}\right)$.
2: Calculer $\hat{\Sigma}$ avec (2.16) et (2.17) évaluées en vect$\left(\hat{\mathbf{A}}\right)$.
3: Construire un GP avec $\hat{\boldsymbol{\theta}}_s^{(l)} = \text{vect}\left(\hat{\mathbf{A}}\right)$. On obtient $\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}$ et $\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}$.
4: Calculer $\nabla\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}$ et $\nabla\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}$.
5: Calculer $\tilde{\mu}_s^{(l)}$ (2.18) et $\tilde{\sigma}_s^{(l)}$ (2.19).

sortie : le MGP

Enfin, GARNETT et al. [40] ont adopté une méthode d'enrichissement adaptatif du DoE utilisant le MGP précédemment défini, la méthode Bayesian active learning by disagreement (BALD) [53], pour découvrir de manière itérative le meilleur sous-espace $\mathcal{A}$. La stratégie d'enrichissement BALD repose sur une fonction $\alpha^{(l)}_{\text{BALD},s} : \mathbb{R}^d \mapsto \mathbb{R}$ représentant l'amélioration du sous-espace linéaire en ajoutant le point $\boldsymbol{x} \in \Omega$ au DoE. Cette fonction s'écrit

$$\alpha^{(l)}_{\text{BALD},s}(\boldsymbol{x}) = \tilde{v}_s^{(l)}(\boldsymbol{x}) \left(\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}\right)^{-1}. \tag{2.20}$$

On cherche ensuite à sélectionner le point $\boldsymbol{x}^{(l+1)}$ qui améliore le plus le sous-espace linéaire, c.-à.-d. qui est solution du problème d'optimisation

$$\boldsymbol{x}^{(l+1)} \in \arg\max_{\boldsymbol{x}\in\Omega} \alpha^{(l)}_{\text{BALD},s}(\boldsymbol{x}). \tag{2.21}$$

On ajoute ensuite ce point au DoE et on itère ce processus jusqu'à un nombre maximal d'itérations max_nb_it $\in \mathbb{N}^+$. Ce processus d'enrichissement itératif est finalement donné par l'Algorithme 2.2.

Algorithme 2.2 Apprentissage adaptatif du sous-espace linéaire.

entrée : fonction à modéliser, DoE initial, un nombre maximal d'itérations max_nb_it, le nombre de directions efficaces d_e

1: **pour** $l = 1$ **à** max_nb_it-1 **faire**
2: Construire le MGP avec d_e directions et le DoE $\mathcal{D}_s^{(l)}$ (voir l'Algorithme 2.1)
3: Trouver $\boldsymbol{x}^{(l+1)}$ la solution du problème d'optimisation (2.21)
4: Évaluer la fonction en $\boldsymbol{x}^{(l+1)}$
5: Mise à jour du DoE
6: **fin pour**

sortie : le MGP

Pour conclure, les MGP offrent trois capacités intéressantes :

— la découverte du sous-espace linéaire qui représente le mieux la fonction à modéliser, avec un nombre de directions efficaces fixé.

— l'apprentissage adaptatif du sous-espace linéaire en ajoutant des points au DoE qui améliorent le plus la qualité du sous-espace.

— la prise en compte de l'incertitude des hyper-paramètres dans la construction du GP.

Cependant, le coût prohibitif de l'estimation des hyper-paramètres en grande dimension limite l'intérêt des MGP aux fonctions comprenant un nombre modéré de variables de conception. En effet, on dénombre $d \cdot d_e$ hyper-paramètres qui sont fixés par optimisation de la fonction de log vraisemblance. Par exemple avec une fonction avec $d = 1000$ variables de conception et $d_e = 2$ directions efficaces, on doit réaliser une optimisation pour trouver $d \cdot d_e = 2000$ hyper-paramètres.

2.1.4.2 Processus Gaussiens combinés aux moindres carrés partiels

Avec le modèle KPLS [17], la matrice $\mathbf{A}$ est découverte grâce à la méthode des moindres carrés partiels, ou partial least squares (PLS) [50] qui cherche les d_e directions orthonormées $\boldsymbol{a}_{(i)} \in \mathbb{R}^d$, $i \in \{1,\dots,d_e\}$ de Ω les plus importantes au regard de l'influence des entrées $\mathbf{X}_{(0)}^{(l)} = \left[\left[\boldsymbol{x}^{(0)}\right]^\top,\dots,\left[\boldsymbol{x}^{(l)}\right]^\top\right]^\top \in \mathbb{R}^{d\times l}$ sur les sorties $\mathbf{Y}_{s,(0)}^{(l)} = \mathbf{Y}_s^{(l)} \in \mathbb{R}^l$. Ces directions sont définies récursivement comme suit :

$$\boldsymbol{a}'_{(i)} \in \arg\max_{\boldsymbol{a}'\in\mathbb{R}^d} \left\{ \boldsymbol{a}'^\top \left[\mathbf{X}_{(i)}^{(l)}\right]^\top \mathbf{Y}_{s,(i)}^{(l)} \left[\mathbf{Y}_{s,(i)}^{(l)}\right]^\top \mathbf{X}_{(i)}^{(l)} \boldsymbol{a}' \ ,\ \boldsymbol{a}'^\top \boldsymbol{a}' = 1 \right\}, \tag{2.22}$$

$$\boldsymbol{t}_{(i)} = \mathbf{X}_{(i)}^{(l)} \boldsymbol{a}'_{(i)}, \quad \boldsymbol{p}_{(i)} = \left[\mathbf{X}_{(i)}^{(l)}\right]^\top \boldsymbol{t}_{(i)}, \quad c_{(i)} = \left[\mathbf{Y}_{s,(i)}^{(l)}\right]^\top \boldsymbol{t}_{(i)} \tag{2.23}$$

$$\mathbf{X}_{(i+1)}^{(l)} = \mathbf{X}_{(i)}^{(l)} - \boldsymbol{t}_{(i)} \boldsymbol{p}_{(i)}, \quad \mathbf{Y}_{s,(i+1)}^{(l)} = \mathbf{Y}_{s,(i)}^{(l)} - c_{(i)} \boldsymbol{t}_{(i)}, \tag{2.24}$$

$$\mathbf{A}' = [\boldsymbol{a}'_{(1)},\dots,\boldsymbol{a}'_{(d_e)}], \quad \mathbf{P} = [\boldsymbol{p}'_{(1)},\dots,\boldsymbol{p}'_{(d_e)}], \quad \mathbf{A} = \mathbf{A}'(\mathbf{P}^\top\mathbf{A}')^{-1}, \tag{2.25}$$

où $\mathbf{X}_{(i+1)}^{(l)} \in \mathbb{R}^{d\times l}$ et $\mathbf{Y}_{s,(i+1)}^{(l)} \in \mathbb{R}^d$ sont les résidus de la projection de $\mathbf{X}_{(i)}^{(l)}$ et $\mathbf{Y}_{s,(i)}^{(l)}$ sur la i^e composante principale $\boldsymbol{t}_{(i)} \in \mathbb{R}^d$. Enfin, $\boldsymbol{p}_{(i)} \in \mathbb{R}^d$ et $c_{(i)} \in \mathbb{R}$ sont respectivement les coefficients de régression de $\mathbf{X}_{(i)}^{(l)}$ et $\mathbf{Y}_{s,(i)}^{(l)}$ sur la i^e composante principale $\boldsymbol{t}_{(i)}$ pour $i \in \{1,\dots,d_e\}$. En réalité, on a maximisé récursivement le carré de la covariance entre $\boldsymbol{a}'_{(i)}$ et $\left[\mathbf{X}_{(i)}^{(l)}\right]^\top \mathbf{Y}_{s,(i)}^{(l)}$.

La matrice $\mathbf{A}$ ainsi obtenue est utilisée dans le noyau de covariance $k_{\mathcal{A}}(\mathbf{A}\boldsymbol{x},\mathbf{A}\boldsymbol{x}')$ et dans la moyenne a priori $\mu_{\mathcal{A}}(\mathbf{A}\boldsymbol{x})$ pour tout $(\boldsymbol{x},\boldsymbol{x}') \in \Omega^2$. Pour construire le GP de s utilisant la matrice $\mathbf{A}$ découverte par PLS, il suffit d'estimer les hyper-paramètres de $k_{\mathcal{A}} : \mathbb{R}^{d_e} \mapsto \mathbb{R}$. Pour les noyaux

classiques (voir Tableau 2.1), le nombre d'hyper-paramètres est restreint de $d+1$ à d_e+1 ce qui réduit fortement le temps nécessaire à leur estimation comparé à un GP classique.

Le processus complet de construction d'un modèle KPLS est donné dans l'Algorithme 2.3.

Algorithme 2.3 Processus de construction d'un modèle KPLS.

entrée : le DoE initial, le nombre de composantes principales d_e

1: **pour** i dans $\{1,\dots,d_e\}$ **faire**
2: Résoudre (2.22) pour trouver la i^{e} direction de $\mathbf{A}$.
3: **fin pour**
4: Estimer $\hat{\boldsymbol{\theta}}_s^{(l)}$ par maximisation de la fonction de vraisemblance.
5: Calculer $\hat{\mu}_s^{(l)}$ (2.2) et $\hat{\sigma}_s^{(l)}$ (2.3).

sortie : le modèle KPLS

Pour conclure, les modèles KPLS offrent deux capacités intéressantes :

— la découverte du sous-espace linéaire, avec un nombre de directions efficaces fixé, qui représente le mieux la fonction à modéliser.

— la réduction du nombre d'hyper-paramètres ce qui limite le temps nécessaire à leur estimation comparé à un GP classique.

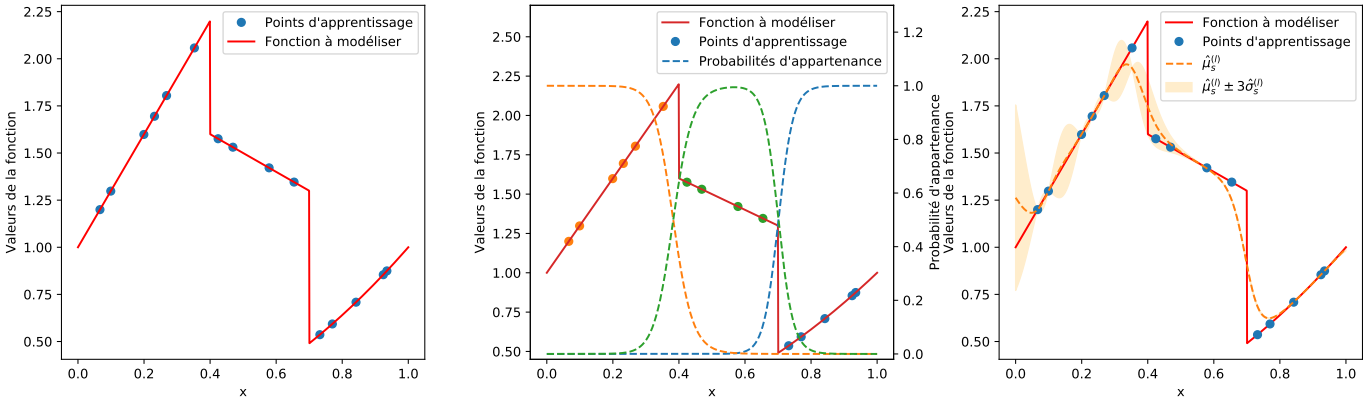
Ces deux avantages sont contrebalancés par l'absence de prise en compte de l'incertitude du sous-espace linéaire comme les MGP peuvent le faire.

Dans cette section sur les GP pour la grande dimension, on a introduit deux méthodes qui reposent sur l'hypothèse que la fonction à modéliser ne dépend que de d_e directions linéairement dépendantes des directions de l'espace de conception initial. On a d'abord présenté le MGP [40] qui apprend la distribution du sous-espace linéaire grâce à l'approximation de Laplace. On est ensuite capable de générer un GP qui prend en compte cette incertitude, c.-à-d. une distribution, dans la définition de sa moyenne et de son écart type. Toutefois, cette méthode est très coûteuse en temps de calcul à cause du grand nombre d'hyper-paramètres à estimer. On a aussi rappelé le modèle KPLS qui apprend le sous-espace linéaire grâce à la méthode des PLS. Ceci permet de réduire de manière drastique le nombre d'hyper-paramètres tout en trouvant le sous-espace le plus représentatif de la fonction à modéliser. Toutefois, cette dernière méthode ne considère pas l'incertitude du sous-espace linéaire découvert.

2.1.5 Le mélange d'experts, ou mixture of experts

Le mélange d'experts, ou mixture of experts (MOE) [12] cherche à modéliser les fonctions fortement multi-modales, voir discontinues, ce qui est difficilement réalisable par des modèles de substitution classiques. Par exemple, la Figure 2.4 montre une de ces fonctions. Elle nous servira d'exemple pour la mise en œuvre des MOE. L'idée des MOE est de diviser l'espace de conception en plusieurs sous-espaces, nommés clusters, qui rassemblent les points voisins de même comportement et de construire un modèle de substitution pour chacun de ces clusters. Ces modèles de substitution peuvent notamment être des GP.

La recherche des clusters est réalisée avec un modèle de mélange gaussien, ou Gaussian mixture model (GMM) construit grâce à l'algorithme de maximisation des attentes, ou expectation-maximization (EM). Le GMM trouve la meilleure répartition des clusters sur l'espace de conception en fonction d'un critère donné, de la valeur de la fonction s, du nombre $\mathrm{N}_{cluster}$ de clusters souhaités et des



(a) Fonction à modéliser. (b) Classement des points d'apprentissage. (c) Mélange d'experts de GP.

FIGURE 2.4 – Exemple de mise en oeuvre d'un mélange d'experts de GP.

points d'apprentissage fournis par un DoE $\mathcal{D}_s^{(l)}$. Après cette étape d'apprentissage des différents clusters, le GMM fournit une fonction de probabilité d'appartenance $\beta_i : \Omega \mapsto [0,1]$ pour chaque cluster $i \in \{1,\dots,N_c\}$ telle que $\sum_{i=1}^{N_c} \beta_i = 1$. Ainsi, chaque point $\boldsymbol{x}$ du domaine Ω est associé au cluster ayant la plus forte probabilité d'appartenance. La Figure 2.4b montre la fonction à modéliser en rouge, les points d'apprentissage repartis dans trois clusters en orange, vert et bleu, et les fonctions de probabilité d'appartenance à chaque cluster en pointillé. On constate bien que la somme des probabilité d'appartenance vaut 1 et que les points d'apprentissage sont bien classés.

Pour construire le MOE résultant de la division fournie par le GMM, on classe chaque point de $\mathcal{D}_s^{(l)}$ en fonction de leur appartenance aux clusters. Ensuite, un modèle de substitution est construit sur chaque cluster grâce aux points triés. On obtient donc N_c modèles de substitution indépendants. En particulier, si on utilise des GP comme modèles de substitution, une moyenne $\hat{\mu}_{s,i}^{(l)}$ et une variance $\hat{\sigma}_{s,i}^{(l)}$ sont définies sur chaque cluster. Par la suite, le modèle global est constitué en faisant une combinaison linéaire de chaque modèle en utilisant la probabilité d'appartenance de chaque point du domaine à chaque cluster. Par exemple, avec des GP sur chaque cluster, une distribution gaussienne $\mathcal{N}\left(\hat{\mu}_s^{(l)}(\boldsymbol{x}), \left[\hat{\sigma}_s^{(l)}(\boldsymbol{x})\right]^2\right)$ est définie en chaque point du domaine. On peut avoir une combinaison continue, ou smooth, telle que

$$\hat{\mu}_s^{(l)} = \sum_{i=1}^{N_c} \beta_i(\boldsymbol{x}) \hat{\mu}_{s,i}^{(l)}(\boldsymbol{x}), \tag{2.26}$$

$$\left[\hat{\sigma}_s^{(l)}\right]^2 = \sum_{i=1}^{N_c} \left[\beta_i(\boldsymbol{x}) \hat{\sigma}_{s,i}^{(l)}\right]^2; \tag{2.27}$$

ou une combinaison discontinue, ou hard, telle que :

$$\hat{\mu}_s^{(l)} = \sum_{i=1}^{N_c} \delta_i(\boldsymbol{x}) \hat{\mu}_{s,i}^{(l)}(\boldsymbol{x}), \tag{2.28}$$

$$\left[\hat{\sigma}_s^{(l)}\right]^2 = \sum_{i=1}^{N_c} \left[\delta_i(\boldsymbol{x}) \hat{\sigma}_{s,i}^{(l)}\right]^2, \tag{2.29}$$

avec $\delta_i(\boldsymbol{x}) = 0$ si $\beta_i(\boldsymbol{x}) < \beta_j(\boldsymbol{x})$ pour tout $j \in \{1,\dots,N_c\}\backslash\{i\}$, sinon $\delta_i(\boldsymbol{x}) = 1$. Avec une combinaison

smooth (2.26), (2.27), un MOE construit avec des GP interpolants n'est pas un modèle de substitution interpolant. Au contraire, avec une combinaison hard (2.28), (2.29), le MOE est interpolant. Toutefois, une discontinuité sera toujours présente alors qu'elle ne le sera pas avec une combinaison smooth. Par exemple, la Figure 2.4c montre la construction d'un MOE utilisant une combinaison smooth. On voit les deux phénomènes présentés, c.-à-d. la non-interpolation des points du DoE par le MOE et la disparition de la discontinuité.

2.1.6 Conclusion

Pour conclure cette section sur les GP, on a introduit leur formulation, les noyaux de covariance classiques ainsi que l'estimation des hyper-paramètres de ces noyaux par maximum de vraisemblance. Au-delà d'environ une vingtaine de variables, l'estimation de ces hyper-paramètres peut être coûteuse en temps de calcul. C'est pourquoi, on s'est intéressé aux extensions des GP à la grande dimension ($d > 20$) et plus particulièrement à deux méthodes. Ces méthodes font l'hypothèse que les fonctions de grande dimension dépendent uniquement de $d_e < d$ directions, appelées directions efficaces, qui sont linéairement dépendantes des directions de l'espace de départ. Le sous-espace qui représente le mieux la fonction en dimension réduite est appelé sous-espace linéaire. D'une part, le modèle MGP [40] construit le sous-espace linéaire par maximisation de la vraisemblance et considère une incertitude associée. Ceci permet notamment de définir une méthode d'enrichissement adaptatif pour améliorer la qualité du sous-espace. Le nombre d'hyper-paramètres nécessaires pour la construction d'un MGP est grand et rend leur estimation coûteuse en temps de calcul. D'autre part, le modèle KPLS [17] recherche le sous-espace linéaire grâce à la méthode des PLS. Pour construire un modèle KPLS, il suffit ensuite d'estimer les hyper-paramètres du noyau de covariance dans le sous-espace linéaire. Ceci réduit fortement le temps nécessaire à leur estimation puisqu'ils sont beaucoup moins nombreux. Le modèle KPLS obtenu ne prend pas en compte l'incertitude liée au sous-espace linéaire obtenu par PLS et qui peut être non négligeable avec un DoE comprenant peu de points. Finalement, on a présenté le MOE de GP qui permet de modéliser des fonctions fortement multi-modales par un unique modèle de substitution. Cependant, cette méthode de modélisation nécessite certains choix a priori de l'utilisateur sur certains hyper-paramètres comme le nombre de clusters.

2.2 Généralités sur l'optimisation Bayésienne sans contrainte

Le processus d'optimisation Bayésienne, ou Bayesian optimization (BO) [57, 71] a d'abord été introduit pour résoudre le problème d'optimisation sans contrainte suivant :

$$\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x}), \tag{2.30}$$

où $\Omega \subset \mathbb{R}^d$ est l'espace de conception et $f : \mathbb{R}^d \mapsto \mathbb{R}$ est une fonction objectif, dite boîte noire, qui est coûteuse à évaluer et dont l'unique information disponible est la valeur de la fonction aux points évalués. Par exemple, les dérivées, la régularité et la Hessienne sont inconnus. Dans notre cas, on s'intéresse plus particulièrement aux problèmes comportant plus d'une centaine de variables de conception.

Dans cette section, on introduit d'abord le principe de fonctionnement du processus de BO

sans contrainte. Puis, on définit les différentes fonctions d'acquisition nécessaires à la compréhension du document. Enfin, les travaux traitant du processus HDBO sont étudiés.

2.2.1 Principe de fonctionnement

Pour résoudre le problème d'optimisation non-contraint (2.30), la méthode BO [57, 71], dont l'efficient global optimization (EGO) est la première version, construit un GP [61, 98] de la fonction objectif f grâce à un DoE de l points d'échantillonnage de l'espace de conception. On peut, entre autres, utiliser tous les GP introduits dans la section précédente. Par exemple, BARTOLI et al. [9] utilisent des MOE de GP dans leur algorithme de BO. Ensuite, la solution optimale est obtenue en enrichissant itérativement le GP. Cet enrichissement est réalisé par le biais d'une stratégie de recherche qui équilibre l'exploration de l'espace de conception Ω et la minimisation de la prédiction du GP de f. En réalité, la stratégie de recherche nécessite la résolution d'un sous-problème de maximisation d'une fonction d'acquisition $\alpha_f^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}$ [9, 37, 117, 129, 135] qui code ce compromis exploration/exploitation. Ce sous-problème d'optimisation est donné comme suit :

$$\max_{\boldsymbol{x}\in\Omega} \alpha_f^{(l)}(\boldsymbol{x}). \tag{2.31}$$

Les fonctions d'acquisition se définissent uniquement à partir du GP. La résolution de (2.31) est donc peu coûteuse, en terme de temps de calcul, comparé au problème d'optimisation initial (2.30). Le DoE est ensuite enrichi avec la solution du sous-problème d'optimisation (2.31). Ce processus est finalement répété jusqu'à ce qu'un nombre maximal d'itérations max_nb_it soit atteint. Les principales étapes de la méthode BO sont résumées dans l'Algorithme 2.4.

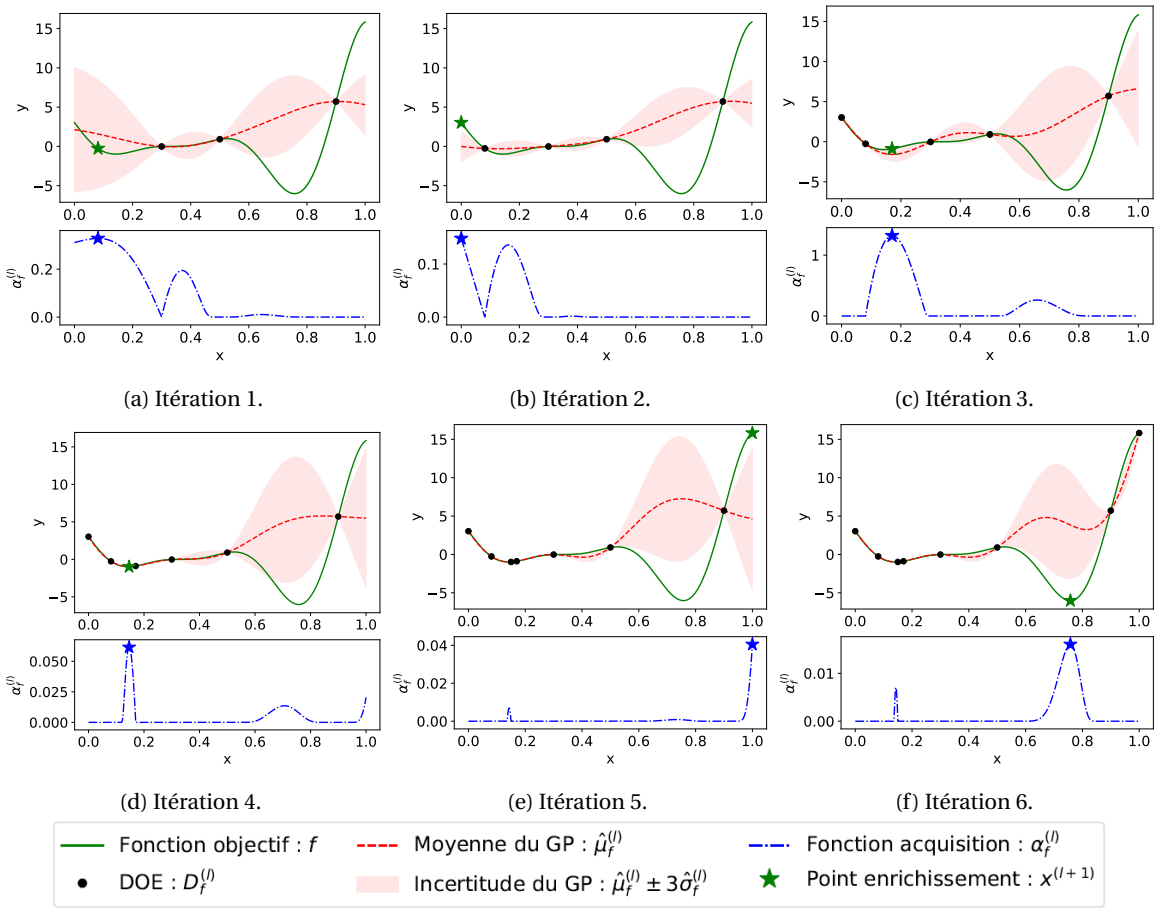
Algorithme 2.4 La méthode BO.

entrée : une fonction objectif, un DoE initial, un nombre maximal d'itérations max_nb_it
1: **pour** $l = 0$ **à** max_nb_it - 1 **faire**
2: Construire un GP de la fonction objectif
3: Trouver $\boldsymbol{x}^{(l+1)}$ la solution du sous-problème d'enrichissement (2.31)
4: Évaluer la fonction objectif en $\boldsymbol{x}^{(l+1)}$
5: Mettre à jour le DoE
6: **fin pour**
sortie : Le meilleur point du DoE

Par exemple, la Figure 2.5 montre six itérations d'un processus de BO sur la fonction unidimensionnelle $f(\boldsymbol{x}) = (6\boldsymbol{x}-2)^2 \sin(12x-4)$ [36]. Sur cette figure, on remarque d'une part que les itérations 3 et 4 (voir Figures 2.5c et 2.5d), correspondent à des étapes d'exploitation car la recherche se fait dans un domaine où l'incertitude est faible et la prédiction du GP est plus petite que la valeur du minimum courant du DoE. D'autre part, les itérations 5 et 6 (voir Figures 2.5e et 2.5f), montrent clairement deux étapes d'exploration où la moyenne $\hat{\mu}_f^{(l)}$ est plus grande que le minimum courant et l'écart type $\hat{\sigma}_f^{(l)}$ est grand.

2.2.2 Fonction d'acquisition

Les fonctions d'acquisition $\alpha_f^{(l)}$, dans le cadre d'un processus de BO, sont des mesures de l'information apportée par un point $\boldsymbol{x}$ du domaine de conception Ω sur le minimum de la fonction objectif f. L'idée est ensuite d'ajouter le point apportant le plus d'information, c.-à-d. qui



(a) Itération 1. (b) Itération 2. (c) Itération 3.

(d) Itération 4. (e) Itération 5. (f) Itération 6.

FIGURE 2.5 – Six itérations d'un processus de BO sur l'exemple unidimensionnel $f(\boldsymbol{x}) = (6\boldsymbol{x}-2)^2 \sin(12x-4)$.

maximise la fonction d'acquisition, dans le DoE $\mathcal{D}_f^{(l)}$ pour résoudre de manière itérative le problème d'optimisation (2.30). Il existe de nombreuses fonctions d'acquisition qui sont soit analytiques [9, 57, 62, 124, 137], c.-à-d. qu'elles ne nécessitent pas de méthodes d'estimation pour être calculées en chaque point du domaine Ω, soit non-analytiques [51, 83, 126, 133, 135]. Le lecteur peut se reporter à deux revues [37, 117] portant sur la BO pour plus d'informations sur les fonctions d'acquisition. Dans la suite, on introduit uniquement les fonctions d'acquisition utilisées dans ce manuscrit.

2.2.2.1 L'amélioration espérée ou expected improvement

L'amélioration espérée, ou expected improvement (EI) [71, 139] est une fonction d'acquisition qui cherche à estimer l'espérance d'amélioration de la valeur du minimum courant dans le DoE en ajoutant un point. Formellement, l'EI du point $\boldsymbol{x}$ par rapport à la valeur du minimum $f_{min}^{(l)} = \min \mathbf{Y}_f^{(l)}$ est donné par :

$$\alpha_{\mathrm{EI},f}^{(l)}(\boldsymbol{x}) = \mathbb{E}\left[\mathbb{I}(\boldsymbol{x})\right] = \mathbb{E}\left[\max\left(0, f_{min}^{(l)} - y_{f,\boldsymbol{x}}^{(l)}\right)\right]. \tag{2.32}$$

Dans le cas où $y^{(l)}_{f,\boldsymbol{x}}$ suit une loi gaussienne donnée par (2.1), cette dernière expression est analytique. Ainsi, on obtient l'expression suivante :

$$\alpha^{(l)}_{\mathrm{EI},f}(\boldsymbol{x}) = \begin{cases} 0, & \text{si } \hat{\sigma}^{(l)}_f(\boldsymbol{x}) = 0 \\ \left(f^{(l)}_{min} - \hat{\mu}^{(l)}_f(\boldsymbol{x})\right)\Phi\left(\frac{f^{(l)}_{min} - \hat{\mu}^{(l)}_f(\boldsymbol{x})}{\hat{\sigma}^{(l)}_f(\boldsymbol{x})}\right) + \hat{\sigma}^{(l)}_f(\boldsymbol{x})\phi\left(\frac{f^{(l)}_{min} - \hat{\mu}^{(l)}_f(\boldsymbol{x})}{\hat{\sigma}^{(l)}_f(\boldsymbol{x})}\right), & \text{sinon} \end{cases}, \tag{2.33}$$

où les fonctions $\Phi : \mathbb{R}^d \mapsto \mathbb{R}$ et $\phi : \mathbb{R}^d \mapsto \mathbb{R}$ sont respectivement les fonctions de répartition et de densité de la loi normale centrée réduite.

Grâce au second terme de (2.33), cette fonction d'acquisition permet d'explorer le domaine de conception où le GP n'est pas précis, c.-à-d. où l'écart type $\hat{\sigma}^{(l)}_f$ est grand. Au contraire, avec le premier terme de (2.33), les zones de faible valeur estimée sont approfondies et plus particulièrement les zones où la moyenne $\hat{\mu}^{(l)}_f$ est plus faible que le minimum connu $f^{(l)}_{min}$.

De nombreux travaux ont montré la capacité de cette fonction d'acquisition à résoudre des problèmes d'ingénierie [57] ou d'estimation d'hyper-paramètres en ML [121]. Elle est toutefois moins performante lorsque le nombre de points ajoutés dans le DoE devient trop grand. En effet, cela signifie que la moyenne $\hat{\mu}^{(l)}_f$ est plus grande que $f^{(l)}_{min}$ et que l'écart type $\sigma^{(l)}_f$ est faible pour un grand nombre de points du domaine de conception. Ceci crée un grand nombre de plateaux à valeur nulle et rend le sous-problème d'optimisation (2.31) difficile à résoudre. Par exemple, la Figure 2.6 montre une itération du processus BO pour la fonction introduite dans la Section 2.2.1. On remarque bien le plateau à valeur nulle pour $\boldsymbol{x} \in [0.8, 1]$. Un algorithme d'optimisation clas-

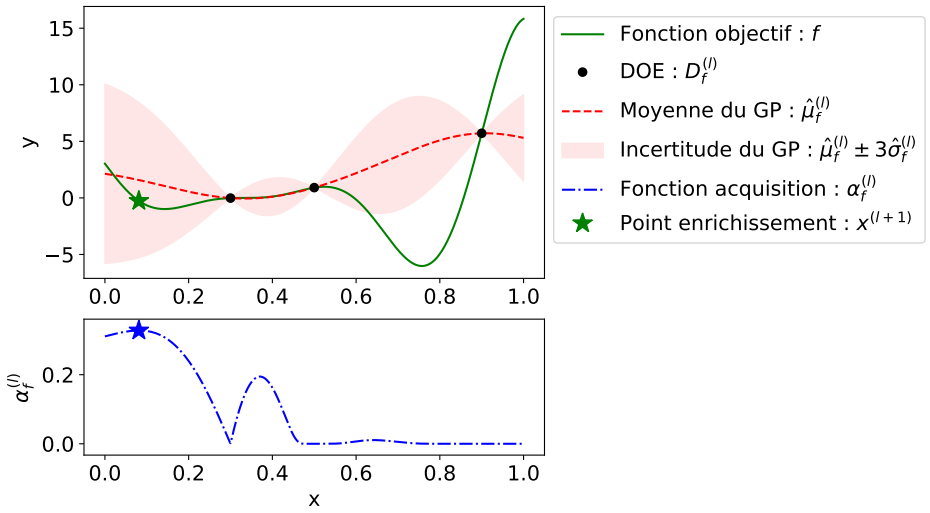


FIGURE 2.6 – Une itération du processus BO pour la fonction d'acquisition EI sur $f(\boldsymbol{x}) = (6\boldsymbol{x}-2)^2 \sin(12x-4)$.

sique a beaucoup de difficultés à gérer de telles zones. La sélection du point d'enrichissement est donc faussée.

2.2.2.2 La fonction d'acquisition de Watson et Barnes 2 normalisée

La fonction d'acquisition Watson Barnes 2 (WB2) [137] cherche à supprimer les zones de plateau à valeur nulle de l'EI. Pour cela, la moyenne $\hat{\mu}^{(l)}_f$ est ajoutée à l'EI pour produire $\alpha^{(l)}_{\mathrm{WB2},f}$ comme suit :

$$\alpha^{(l)}_{\mathrm{WB2},f}(\boldsymbol{x}) = \alpha^{(l)}_{\mathrm{EI},f}(\boldsymbol{x}) + \hat{\mu}^{(l)}_f(\boldsymbol{x}). \tag{2.34}$$

Ce changement permet d'avoir une fonction plus régulière et donc un sous-problème d'enrichissement (2.31) plus simple à résoudre.

Néanmoins, on rappelle que $\alpha^{(l)}_{\mathrm{EI},f}$ tend vers 0 lorsque le nombre de points augmente dans le DoE. On remarque donc que $\alpha^{(l)}_{\mathrm{EI},f}$ et $\hat{\mu}^{(l)}_f$ ne sont pas toujours du même ordre de grandeur ce qui peut réduire le caractère exploratoire de $\alpha^{(l)}_{\mathrm{WB2},f}$. Ainsi, il y a plus de chance que le sous-problème d'optimisation (2.31) utilisé avec WB2 donne des solutions locales malgré la suppression des plateaux à valeur nulle. Par exemple, en comparant les Figures 2.6 et 2.7a, on remarque d'une part que les plateaux à valeur nulle ne sont plus présents et que d'autre part, le point d'enrichissement sélectionné n'est pas le même entre l'EI et WB2. En effet, les variations de WB2 suivent les variations de $\hat{\mu}^{(l)}_f$ ce qui indique une tendance à favoriser l'exploitation de la moyenne du GP sans prendre en compte l'écart type.

Une extension de WB2 introduite par BARTOLI et al. [9] cherche le même compromis d'exploration/exploitation que l'EI tout en supprimant les plateaux à valeur nulle. Pour cela, la part de l'EI est rendue prépondérante dans WB2 par le biais d'un facteur de normalisation $s^{(l)} \in \mathbb{R}^+$. La fonction d'acquisition, nommée Watson Barnes 2 normalisée, ou scaled Watson Barnes (WB2S), est donnée par :

$$\alpha^{(l)}_{\mathrm{WB2S},f}(\boldsymbol{x}) = s^{(l)}\alpha^{(l)}_{\mathrm{EI},f}(\boldsymbol{x}) + \hat{\mu}^{(l)}_f(\boldsymbol{x}), \tag{2.35}$$

où $s^{(l)} = 1$ si $\alpha^{(l)}_{\mathrm{EI},f}\left(\boldsymbol{x}_{\mathrm{EI}_{max}}\right) = 0$, sinon

$$s^{(l)} = \frac{\beta\left|\hat{\mu}^{(l)}_f\left(\boldsymbol{x}_{\mathrm{EI}_{max}}\right)\right|}{\alpha^{(l)}_{\mathrm{EI},f}\left(\boldsymbol{x}_{\mathrm{EI}_{max}}\right)} \tag{2.36}$$

avec $\beta \in \mathbb{R}^+$, $\boldsymbol{x}_{\mathrm{EI}_{max}} \in \arg\max_{\boldsymbol{x}\in\mathbf{X}_{\mathrm{EI}}} \alpha^{(l)}_{\mathrm{EI},f}(\boldsymbol{x})$ et $\mathbf{X}_{\mathrm{EI}} \in \mathbb{R}^{d\times 100d}$ un ensemble de $100d$ points de l'espace de conception Ω échantillonnés avec la méthode échantillonnage par hypercube latin, latin hypercube sampling (LHS) que l'on évalue sur l'EI. Cette fonction d'acquisition normalise WB2 pour que la valeur de WB2S à $\boldsymbol{x}_{\mathrm{EI}_{max}}$ soit environ β fois plus forte que la valeur de la moyenne $\hat{\mu}^{(l)}_f$ au même point. Le but est de faire coïncider la valeur du maximum de l'EI avec le maximum de WB2S. En pratique, on utilise $\beta = 100$.

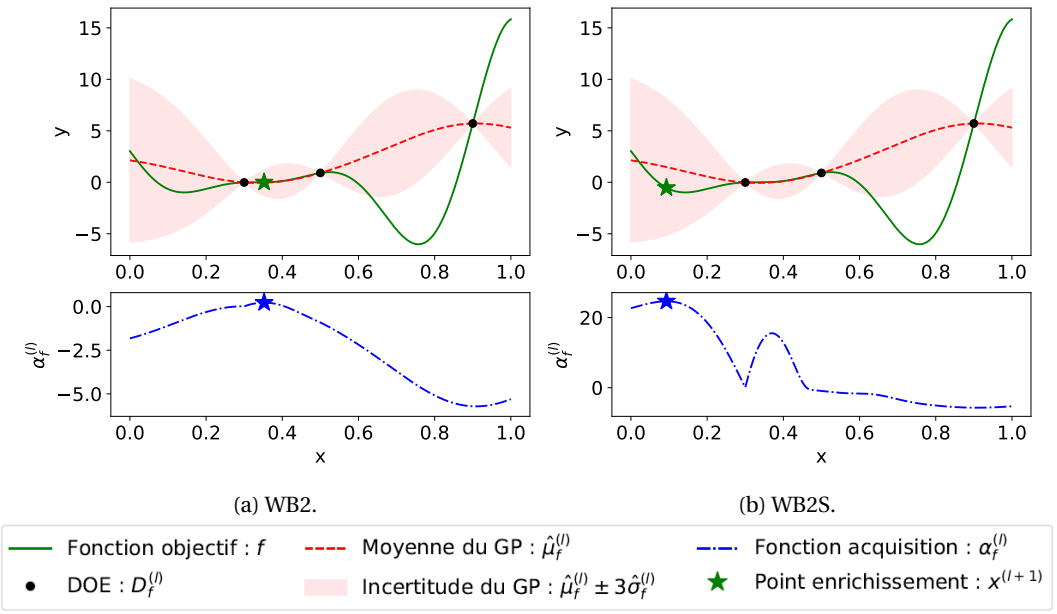
Les Figures 2.6, 2.7a et 2.7b montrent que WB2S et EI suivent les mêmes variations et permettent donc le même compromis exploration/exploitation. De plus, WB2S n'a pas de plateau à valeur nulle. La résolution du sous-problème d'optimisation (2.31) est donc facilitée.

2.2.2.3 Échantillonage de Thomson, ou Thomson sampling

La fonction d'acquisition échantillonage de Thomson, ou Thomson sampling (TS) [37, 117] repose sur le caractère aléatoire du GP. Elle utilise une réalisation du GP, noté $y^{(l)}_s : \mathbb{R}^d \mapsto \mathbb{R}$, sur l'ensemble du domaine comme fonction d'acquisition. En effet, une réalisation d'un GP sur l'ensemble du domaine est une fonction. Cette nouvelle fonction d'acquisition est donnée par :

$$\alpha^{(l)}_{\mathrm{TS}}(\boldsymbol{x}) = y^{(l)}_s(\boldsymbol{x}). \tag{2.37}$$

Comme $\alpha^{(l)}_{\mathrm{TS}}$ est une réalisation du GP de la fonction objectif, il est possible de générer $q \in \mathbb{N}^+$ réalisations et donc de créer q fonctions d'acquisition. Ainsi, à chaque itération du processus BO,



(a) WB2. (b) WB2S.

FIGURE 2.7 – Une itération du processus BO pour les fonctions d'acquisition WB2 et WB2S sur $f(\boldsymbol{x}) = (6\boldsymbol{x} - 2)^2 \sin(12x - 4)$.

on peut ajouter q points aux DoE.

Cependant, cette fonction d'acquisition est fortement exploratoire ce qui n'est pas enviable en grande dimension. En effet, en grande dimension, l'espace de recherche est très grand et une fonction d'acquisition trop exploratoire risque de se perdre dans l'immensité du domaine de conception.

2.2.2.4 La réduction progressive de l'incertitude ou stepwise uncertainty reduction

La fonction d'acquisition réduction progressive de l'incertitude, ou stepwise uncertainty reduction (SUR) [83] calcule le potentiel de chaque point $\boldsymbol{x}^+ \in \Omega$ pour réduire le volume espéré du GP sous $f_{min}^{(l)}$. Le volume espéré sous $f_{min}^{(l)}$ se traduit par :

$$\mathbb{E}_\Omega\left[y_{s,\boldsymbol{x}}^{(l)} \leq f_{min}^{(l)}\right] = \int_\Omega \Phi\left(\frac{f_{min}^{(l)} - \hat{\mu}_f^{(l)}(\boldsymbol{x})}{\hat{\sigma}_f^{(l)}(\boldsymbol{x})}\right) \mathrm{d}\boldsymbol{x}. \tag{2.38}$$

Le but est maintenant d'estimer ce volume si on ajoute un point $\boldsymbol{x}^+$ au DoE, c.-à-d.

$$\mathbb{E}_\Omega\left[y_{f,\boldsymbol{x}}^{(l)+} \leq \min\left(f_{min}^{(l)}, f(\boldsymbol{x}^+)\right)\right], \tag{2.39}$$

où $y_{f,\boldsymbol{x}}^{(l)+}$ est la variable aléatoire suivant la loi définie par le GP construit à partir du DoE $\mathcal{D}_f^{(l)}$ auquel on a ajouté le point $\{\boldsymbol{x}^+, f(\boldsymbol{x}^+)\}$. Comme on ne souhaite pas calculer $f(\boldsymbol{x}^+)$ dans le sous-problème d'enrichissement (2.31), on cherche à estimer l'espérance du volume sous $f_{min}^{(l),+} = \min\left(f_{min}^{(l)}, y_{f,\boldsymbol{x}^+}^{(l)}\right)$. La fonction d'acquisition SUR est finalement donnée par l'opposé de l'espérance de la loi du volume sous $f_{min}^{(l),+}$:

$$\alpha_{\mathrm{SUR},f}^{(l)}(\boldsymbol{x}^+) = -\mathbb{E}_\Omega\left[\mathrm{Y}_{f,\boldsymbol{x}}^{(l)} \leq f_{min}^{(l),+}\right]. \tag{2.40}$$

Contrairement à EI, à WB2 et à WB2S, la fonction d'acquisition SUR n'est pas analytique à cause de l'intégration sur le domaine Ω. On peut néanmoins l'estimer en utilisant une méthode

de Monte-Carlo.

Cette fonction d'acquisition a montré de bons résultats sur des problèmes de faible dimension [83]. Toutefois, elle ne peut pas être utilisée pour des problèmes en grande dimension à cause de la phase d'estimation par Monte-Carlo nécessaire à chaque évaluation de la fonction d'acquisition. En effet, seuls des problèmes en dimension $d = 2$ et $d = 4$ ont été testés par PICHENY [83].

2.2.2.5 La recherche par prédiction d'entropie ou predictive entropy search

La fonction d'acquisition de recherche entropique prédictive, ou predictive entropy search (PES) [51] mesure la quantité d'information apportée sur la localisation du minimum $\boldsymbol{x}^*$ par l'ajout d'un point $\boldsymbol{x}$ au DoE. De plus, en notant

$$\boldsymbol{x}^{(l),*} \in \arg\min_{\boldsymbol{x}\in\Omega} \; y^{(l)}_{f,\boldsymbol{x}} \tag{2.41}$$

la variable aléatoire de la localisation du minimum, $p\left(\boldsymbol{x}^*|\mathcal{D}^{(l)}_f\right)$ sa fonction de densité et $\mathbb{P}\left(\boldsymbol{x}^*|\mathcal{D}^{(l)}_f\right)$ sa loi de probabilité, cette mesure peut se calculer grâce à l'opposé de l'entropie différentielle de $\mathbb{P}\left(\boldsymbol{x}^*|\mathcal{D}^{(l)}_f\right)$ que l'on note $\mathrm{H}\left[\mathbb{P}\left(\boldsymbol{x}^*|\mathcal{D}^{(l)}_f\right)\right]$ avec

$$\mathrm{H}\left[\mathbb{P}(\boldsymbol{x})\right] = -\int_\Omega p(\boldsymbol{x}) \log\left(p(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}. \tag{2.42}$$

On ajoute au DoE le point qui maximise l'espérance de la réduction de cette quantité, c.-à-d. qui maximise

$$\alpha^{(l)}_{\mathrm{PES},f}(\boldsymbol{x}) = \mathrm{H}\left[\mathbb{P}\left(\boldsymbol{x}^*|\mathcal{D}^{(l)}_f\right)\right] - \mathbb{E}_{y^{(l)}_{f,\boldsymbol{x}}}\left[\mathrm{H}\left[\mathbb{P}\left(\boldsymbol{x}^*|\mathcal{D}^{(l)}_f \cup (\boldsymbol{x}, y^{(l)}_{f,\boldsymbol{x}})\right)\right]\right], \tag{2.43}$$

où $\mathbb{E}_X[f(X)] = \int_\Omega f(\boldsymbol{x})p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$. L'évaluation de la PES dans l'état n'est possible qu'après de nombreuses approximations. Pour les éviter, HERNÁNDEZ-LOBATO et al. [51] reformulent (2.43) comme suit :

$$\alpha^{(l)}_{\mathrm{PES},f}(\boldsymbol{x}) = \mathrm{H}\left[\mathbb{P}\left(y_f|\mathcal{D}^{(l)}_f, \boldsymbol{x}\right)\right] - \mathbb{E}_{\boldsymbol{x}^{(l),*}}\left[\mathrm{H}\left[\mathbb{P}\left(y_f|\mathcal{D}^{(l)}_f, \boldsymbol{x}, \boldsymbol{x}^{(l),*}\right)\right]\right], \tag{2.44}$$

avec $\mathbb{P}\left(y_f|\mathcal{D}^{(l)}_f, \boldsymbol{x}, \boldsymbol{x}^{(l),*}\right)$ la loi du GP en $\boldsymbol{x}$ conditionné par $\mathcal{D}^{(l)}_f$ et pour lequel on impose que $\boldsymbol{x}^{(l),*}$ soit le minimum global. Pour estimer cette expression, on réalise une étape d'estimation de la moyenne par échantillonnage de $\boldsymbol{x}^{(l),*}$ suivant la loi $\mathbb{P}\left(\boldsymbol{x}^*|\mathcal{D}^{(l)}_f\right)$. Celle-ci est suivie d'une étape de propagation des attentes, ou expectation propagation [70]. Enfin, une marginalisation des hyperparamètres du GP est réalisée. Le lecteur peut se référer aux travaux de HERNÁNDEZ-LOBATO et al. [51] pour plus d'information sur les trois étapes succinctement introduites ici.

Ces trois étapes d'estimation rendent impossible l'utilisation de la PES en grande dimension. Toutefois, cette fonction d'acquisition a montré de très bons résultats dans le domaine du ML, notamment dans son application à l'algorithme d'AI AlphaGO [22, 119].

On a introduit les cinq principales fonctions d'acquisition de la littérature (EI, WB2, WB2S, SUR et PES). Elles peuvent être de deux types : soit analytiques (EI, WB2, WB2S) ce qui permet leur utilisation pour des problèmes de grande dimension ; soit non-analytiques (SUR, PES), c.-à-d. qui nécessitent une phase d'estimation, ce qui ne permet pas leur utilisation pour des problèmes de grande dimension. En effet, les phases d'estimation nécessaires à leur calcul seraient beaucoup trop coûteuses en temps de calcul. De plus, on a vu que ces fonctions d'acquisition cherchent toutes à ajouter des points qui apportent le plus d'information sur le minimum de la fonction. Par

exemple, en ajoutant des points qui maximisent l'amélioration de la valeur du minimum connu ou en ajoutant des points qui apportent le plus d'information sur la localisation du minimum. Elles ont toutes montré de très bons résultats sur des problèmes d'optimisation académiques, industriels et de ML. Toutefois, elles ne sont pas utilisables pour l'optimisation en grande dimension à cause du temps nécessaire à la création du GP. On remarque aussi un manque de corrélation entre les points du DoE et les autres points du domaine Ω due à l'augmentation exponentielle de la taille du domaine. De plus, une fonction d'acquisition trop exploratoire n'exploiterait pas suffisamment la prédiction du GP et nécessiterait beaucoup de points pour converger vers le minimum. Dans la section suivante, on s'intéresse à cinq méthodes qui tentent de surmonter ces défis.

2.2.3 Optimisation Bayésienne en grande dimension

Les algorithmes de BO classiques peinent à résoudre des problèmes de grande dimension (plus de 20 variables de conception). Ceci est dû à trois manques majeurs de la BO :

1. Le GP n'apporte pas suffisamment d'information. En effet, en grande dimension, la distance entre la plupart des points de Ω et les points du DoE $\mathcal{D}_f^{(l)}$ est grande et ne permet pas une corrélation efficace entre ces points. Ainsi, les prédictions réalisées par le GP ne prennent pas en compte les données du DoE et sont donc inutilisables. Par exemple, avec le noyau exponentiel, le terme $\exp\left(-\theta_{s,i}^{(l)} m_i\right)$ est inférieur à 1 pour tout $i \in \{1,\dots,d\}$ tant que $m_i \neq 0$ (voir Tableau 2.1). Ainsi pour tous les points du domaine Ω qui ne sont pas dans le DoE, la valeur de la corrélation avec les points du DoE est proche de 0 à cause du produit.
2. La construction du GP est lente à cause du nombre d'hyper-paramètres à estimer. De plus, en grande dimension, le DoE contient plus de points qu'en petite dimension pour capter le comportement de la fonction objectif. Or, ceci augmente considérablement la taille de la matrice de covariance ce qui ralentit davantage l'estimation des hyper-paramètres.
3. L'exploration complète de l'espace de conception n'est pas raisonnable au vue de sa taille. Ceci rend les méthodes trop exploratoires inefficaces.

Les problématiques 1 et 2 de la BO ont partiellement été résolues en utilisant les GP pour la grande dimension (voir Section 2.1.4) dans le processus de BO décrit dans l'Algorithme 2.4 [18, 39, 59, 73, 77, 103, 135, 136]. Ceci permet en effet de faciliter la construction du GP mais ne permet pas de réduire l'espace de recherche. Ainsi, le processus d'optimisation est toujours réalisé dans un très grand espace.

L'ensemble de ces problématiques est aussi traité en réalisant une réduction de dimension avec d'autres outils (PCA [138], sous-espace actif [25], variétés [125], etc.). On réduit ainsi l'espace d'entrée et on construit un GP dans l'espace de petite dimension. Ceci limite le temps d'estimation des hyper-paramètres, permet au GP d'apporter plus d'information et restreint la taille du domaine à explorer. De nombreux travaux reposent sur cette hypothèse [14, 15, 20, 60, 66, 74, 134]. Néanmoins, la plupart considèrent l'application de réduction de dimension comme fixe et apprise/choisie uniquement au début du processus d'optimisation. Ceci peut être néfaste au processus d'optimisation puisque l'optimum peut ne pas se trouver dans le sous-espace défini par l'application de réduction.

Enfin, une troisième méthode est utilisée pour résoudre les problématiques 1 et 3. Elle repose sur la décomposition de l'espace de conception en sous-espaces de conception de plus faible di-

mension qui sont traités parallèlement [33]. Ainsi, la taille de ces sous-espaces permet de régler la quantité d'exploration désirée afin de favoriser ou non la convergence vers un minimum local. Ceci est généralement réalisé de la même manière que pour les algorithmes d'optimisation à région de confiance. Néanmoins, les GP sont toujours construits dans l'espace de départ ce qui peut s'avérer très long lorsque le nombre de dimensions augmente fortement.

Dans la suite, on introduit plus en détails les méthodes qui sont utilisées dans ce manuscrit. Plus particulièrement, on introduit les méthodes :

— Efficient global optimization enhanced by a KPLS surrogate model (EGO-KPLS) [18] qui utilise un GP de grande dimension,

— Random embedding Bayesian optimization (REMBO) [134] ainsi que ses extensions hashing-enhanced subspace Bayesian optimization (HeSBO) [74] et R random embedding Bayesian optimization (RREMBO) [15] qui utilisent une réduction de dimension aléatoire préalable au processus d'optimisation.

— Trust region Bayesian optimization (TuRBO) [33] qui utilise un découpage de l'espace de conception en plusieurs sous-espaces de taille réduite associé à une méthode de région de confiance.

2.2.3.1 Efficient global optimization enhanced by a KPLS surrogate model

L'algorithme EGO-KPLS [18] est le même que l'algorithme BO classique excepté que le modèle de substitution utilisé est un modèle KPLS, décrit dans la section 2.1.4.2. Néanmoins, on rappelle la méthode EGO-KPLS dans l'Algorithme 2.5.

Algorithme 2.5 La méthode EGO-KPLS.

entrée : une fonction objectif, un DoE initial, un nombre maximal d'itérations max_nb_it

1: **pour** $l = 0$ **à** max_nb_it - 1 **faire**
2: Construire un modèle KPLS de la fonction objectif
3: Trouver $\boldsymbol{x}^{(l+1)}$ la solution du sous-problème d'enrichissement (2.31)
4: Évaluer la fonction objectif en $\boldsymbol{x}^{(l+1)}$
5: Mettre à jour le DoE
6: **fin pour**

sortie : Le meilleur point du DoE

Cette méthode HDBO a notamment permis de réduire le temps nécessaire à la construction du GP ce qui permet également de réduire fortement le temps total d'optimisation. De plus, la réduction de dimension réalisée dans le noyau de covariance permet une corrélation non-nulle des points du domaine Ω avec les points du DoE. Ainsi la prédiction du GP portera les informations du DoE ce qui n'est pas le cas d'un GP classique de grande dimension. Par exemple avec le noyau exponentiel, il reste uniquement d_e termes du type $\exp\left(-\theta_{s,i}^{(l)} m_i\right)$ (voir Tableau 2.1). Si on choisi d_e suffisamment petit, le produit de ces termes ne sera pas proche de 0 et permettra une corrélation des points.

2.2.3.2 Optimisation Bayésienne à sous-espace aléatoire, ou random embedding Bayesian optimization et ses extensions

On suppose dans [15, 74, 134] que la fonction objectif ne dépend en réalité que de $d_e \ll d$ directions, appelées directions efficaces. Comme dans la Section 2.1.4, on a supposé qu'il existe une fonction $f_{\mathcal{A}} : \mathbb{R}^{d_e} \mapsto \mathbb{R}$ telle que $f_{\mathcal{A}}(\mathbf{A}\boldsymbol{x}) = f(\boldsymbol{x})$ avec $\mathbf{A} \in \mathbb{R}^{d_e \times d}$ et $\mathcal{A} = \{\boldsymbol{u} = \mathbf{A}\boldsymbol{x} \; ; \; \forall \boldsymbol{x} \in \Omega\}$. On rappelle également que $\Omega = [-1, 1]^d$. L'idée est ensuite de réaliser directement l'optimisation dans le sous-espace linéaire $\mathcal{A}$ afin de réduire le nombre d'hyper-paramètres à estimer lors de la construction du GP et de réduire l'espace de recherche du minimum grâce au passage de la dimension d à d_e. Ceci permet, entre autre, de réduire le temps nécessaire à la construction du GP et de faciliter l'optimisation de la fonction d'acquisition. De plus, si le nombre de directions efficaces d_e choisi est suffisamment faible, l'optimisation peut être réalisée avec n'importe quelle fonction d'acquisition, qu'elle soit analytique ou non. Cependant, pour pouvoir réaliser cette optimisation, il est nécessaire de définir :

1. la matrice $\mathbf{A}$,
2. les bornes de l'hypercube de recherche $\mathcal{B} \subset \mathbb{R}^{d_e}$ de dimension reduite,
3. une application $\gamma : \mathcal{A} \mapsto \Omega$ de retour du sous-espace linéaire vers l'espace initial Ω
4. le noyau de covariance du GP.

Ces 4 grandes étapes sont reprises et détaillées ci-dessous.

Matrice de réduction : Dans REMBO [134], on génère aléatoirement la matrice $\mathbf{A}^+ \in \mathbb{R}^{d \times d_e}$ qui permet d'obtenir un point $\boldsymbol{x} \in \mathbb{R}^d$ à partir d'un point $\boldsymbol{u} \in \mathbb{R}^{d_e}$ tel que $\boldsymbol{x} = \mathbf{A}^+ \boldsymbol{u}$. Les colonnes de la matrice $\mathbf{A}^+ \in \mathbb{R}^{d \times d_e}$ sont échantillonnées en suivant une distribution normale centrée réduite. La matrice $\mathbf{A}$ est donc la pseudo-inverse de $\mathbf{A}^+$ donnée par $\mathbf{A} = \left[\mathbf{A}^+\right]^\top \left(\mathbf{A}^+ \left[\mathbf{A}^+\right]^\top\right)^{-1}$. Dans RREMBO [15], la matrice $\mathbf{A}^+$ est générée de la même manière mais on est capable de retrouver facilement la matrice $\mathbf{A}$ de passage de Ω à $\mathcal{A}$. En effet, en réalisant une orthogonalisation de Gram-Schmidt de $\mathbf{A}^+$, on a $\mathbf{A} = \left[\mathbf{A}^+\right]^\top$. Dans HeSBO [74], la matrice $\mathbf{A}$ est composée de 0, 1 et -1. Pour cela, un vecteur $h \in \mathbb{R}^d$ tel que $h_i \in \{1, \ldots, d\}$ et un vecteur $\eta \in \mathbb{R}^d$ tel que $\eta_i \in \{-1, 1\}$ sont générés aléatoirement. On oblige toutefois qu'il y ait au moins une occurrence de chaque nombre de $\{1, \ldots, d_e\}$ dans h. Ainsi, on a $\mathrm{A}_{i,j} = \eta_i \delta_j^{h_i}$ avec $\delta_j^{h_i}$ le symbole de Kronecker.

Bornes de l'espace de recherche : Les bornes de l'espace $\mathcal{B}$ sont définies de manière différente dans REMBO, RREMBO et HeSBO. Elles dépendent fortement de la manière dont sont définies $\mathbf{A}$ ou $\mathbf{A}^+$. En effet, il est impossible de définir avec certitude les bornes de $\mathcal{B}$ pour REMBO et RREMBO car on travaille avec $\mathbf{A}^+$. WANG et al. [134] démontrent qu'il y a de fortes probabilités que le minimum se trouve dans

$$\mathcal{B}_{\mathrm{W}} = \left[-\sqrt{d_e}, \sqrt{d_e}\right]^{d_e}. \tag{2.45}$$

Néanmoins, cette définition ne permet pas toujours de considérer tous les points de $\mathcal{A}$. En réalité, on a souvent $\mathcal{B} \subset \mathcal{A}$ ou $\mathcal{A} \subset \mathcal{B}$. Par exemple, la Figure 2.8 montre que le sous-espace $\mathcal{B}_{\mathrm{W}}$ ne prend en compte que les points représentés par la ligne noire de Ω. Dans ce cas précis, on pourrait prendre $\mathcal{B}_{\mathrm{W}}$ plus grand pour considérer plus de points de Ω dans le problème d'optimisation. En

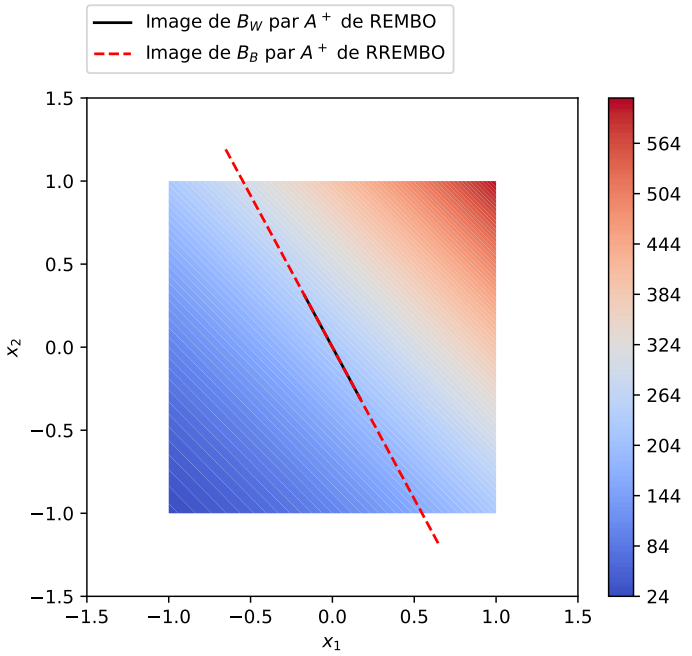


FIGURE 2.8 – Image du segment $\mathcal{B}$ par $\mathbf{A}^+$ en utilisant les définitions de WANG et al. [134] en noir et BINOIS et al. [15] en rouge pour le même sous-espace linéaire.

choisissant $\mathbf{A} = \left[\mathbf{A}^+\right]^\top$, BINOIS et al. [15] peuvent définir le plus petit hyper-rectangle

$$\mathcal{B}_\mathrm{B} = \left[-\sum_{i=1}^{d}\left|\mathrm{A}_{1,i}\right|, \sum_{i=1}^{d}\left|\mathrm{A}_{1,i}\right|\right] \times \cdots \times \left[-\sum_{i=1}^{d}\left|\mathrm{A}_{d_e,i}\right|, \sum_{i=1}^{d}\left|\mathrm{A}_{d_e,i}\right|\right] \tag{2.46}$$

qui contient $\mathcal{A}$. Aucun point $\boldsymbol{x} \in \Omega$ qui est solution directe du problème $\boldsymbol{x} = \mathbf{A}^+\boldsymbol{u}$ pour tout $\boldsymbol{u} \in \mathbb{R}^{d_e}$ n'est oublié dans l'optimisation. On a donc $\mathcal{A} \subset \mathcal{B}_\mathrm{B}$. Néanmoins, cette définition utilise $\mathbf{A} = \left[\mathbf{A}^+\right]^\top$ ce qui implique que certains points de $\boldsymbol{u} \in \mathcal{B}_\mathrm{B}$ ne permettent pas d'avoir $\boldsymbol{x} = \mathbf{A}^+\boldsymbol{u}$ avec $\boldsymbol{x} \in \Omega$. Par exemple, la Figure 2.8 montre l'image de $\mathcal{B}_\mathrm{B}$ par $\mathbf{A}^+$. On remarque que RREMBO définit un espace dont les images par $\mathbf{A}^+$ ne sont pas dans Ω avec des points en dehors de $[-1,1]^2$. Cette propriété est en réalité utilisée pour prendre en compte les valeurs de la fonction objectif sur les bords les plus proches de l'image de $\mathcal{B}_\mathrm{B}$ par $\left[\mathbf{A}^+\right]^\top$ du domaine Ω. Ceci est réalisé grâce à l'application de retour de $\mathcal{B}_\mathrm{B}$ à Ω qui est introduite plus loin. Pour finir, avec la construction particulière de la matrice $\mathbf{A}$, NAYEBI et al. [74] considèrent qu'ils peuvent directement travailler dans

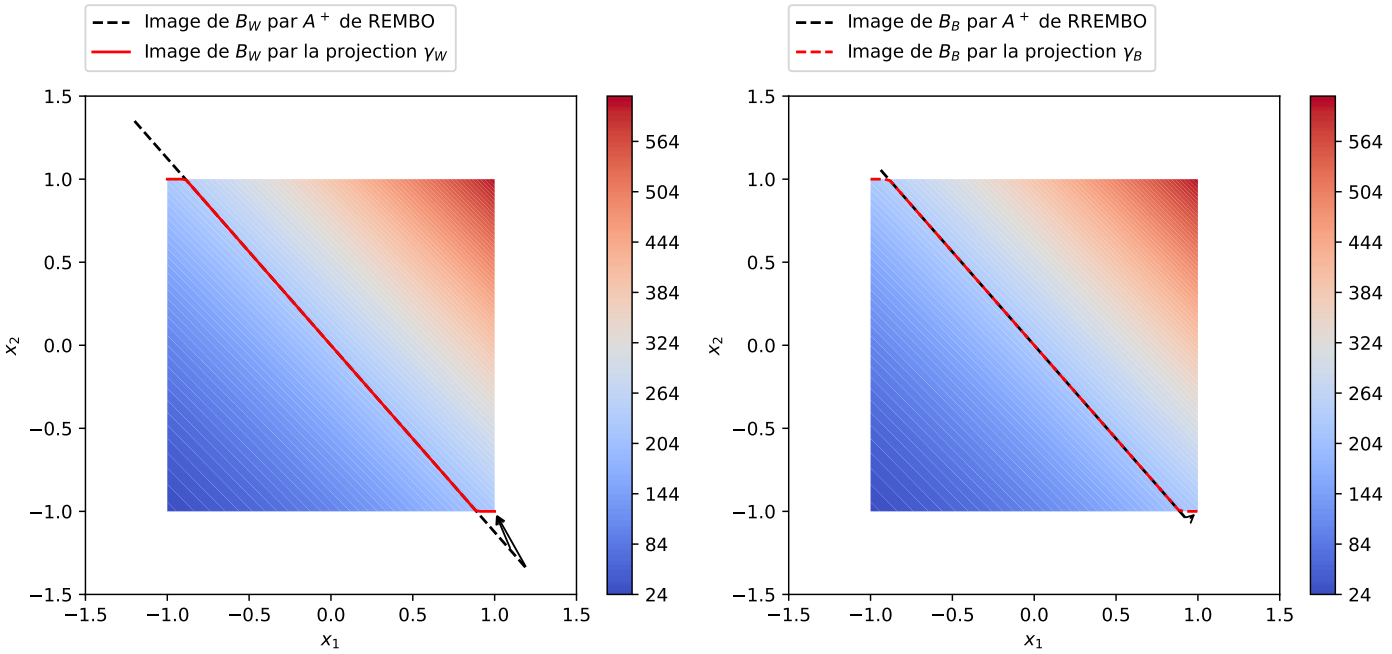
$$\mathcal{B}_\mathrm{N} = [-1,1]^{d_e}. \tag{2.47}$$

Application de retour : Ensuite, on introduit les différentes applications γ de retour de $\mathcal{A}$ vers Ω. Dans REMBO, on cherche le point de $\boldsymbol{x}^* \in \Omega$ qui est le plus proche de $\mathbf{A}^+\boldsymbol{u}$ en résolvant un problème d'optimisation quadratique. L'application correspondante est donnée par

$$\boldsymbol{x}^* = \gamma_\mathrm{W}(\boldsymbol{u}) \in \arg\min_{\boldsymbol{x}\in\Omega} \|\boldsymbol{x} - \mathbf{A}^+\boldsymbol{u}\|^2. \tag{2.48}$$

Cette définition de l'application est simple mais peut donner le même $\boldsymbol{x}^*$ pour plusieurs $\boldsymbol{u} \in \mathcal{B}_\mathrm{W}$. Ainsi la fonction objectif dans le sous-espace comprend des zones de valeur constante qui correspondent à des zones donnant le même $\boldsymbol{x}^*$. De plus, ce phénomène est difficilement pris en

compte par les GP. Par exemple, la Figure 2.9a montre l'image de $\mathcal{B}_\mathrm{W}$ par $\mathbf{A}^+$ en noir et l'image de $\mathcal{B}_\mathrm{W}$ par γ_W en rouge. On constate que les points aux extrémités sont projetés sur le même point



(a) Image de $\mathcal{B}_\mathrm{W}$ par $\mathbf{A}^+$ ou γ_W.

(b) Image de $\mathcal{B}_\mathrm{B}$ par $\mathbf{A}^+$ ou γ_B.

FIGURE 2.9 – Comparaison entre les espaces $\mathcal{B}$ et la projection γ de REMBO et RREMBO. La flèche noire représente la projection d'un point donné avec en queue, un point de $\mathbb{R}^d$ image de $\mathcal{B}$ par $\mathbf{A}^+$; et en tête, la projection de ce point par γ sur Ω.

du bord de Ω. De plus, le sous-espace aléatoire choisi ne permet pas d'atteindre le minimum de la fonction objectif situé en $\boldsymbol{x} = [-1,-1]$. Dans RREMBO, l'application est choisie pour être bijective entre $\mathcal{A}$ et Ω, c.-à-d. qu'il y a un unique $\boldsymbol{x} \in \Omega$ pour chaque $\boldsymbol{u} \in \mathcal{A}_\mathrm{B}$ et qu'il y a un unique $\boldsymbol{u} \in \mathcal{A}_\mathrm{B}$ pour chaque $\boldsymbol{x} \in \Omega$. L'application est finalement donnée par

$$\gamma_\mathrm{B}(\boldsymbol{u}) \in \arg\min_{\boldsymbol{x}\in\Omega} \left\{ \|\boldsymbol{x} - \mathbf{A}^+\boldsymbol{u}\|^2, \text{ s.c. } \mathbf{A}\boldsymbol{x} = \boldsymbol{u} \right\}, \tag{2.49}$$

qui nécessite la résolution d'un problème quadratique d'optimisation. Par exemple, la Figure 2.9b représente les images de l'espace $\mathcal{B}_\mathrm{B}$ par $\mathbf{A}^+$ et par la projection γ_B. On remarque que la projection de RREMBO définit une injection de $\mathcal{B}_\mathrm{B}$ dans Ω [15] ce qui ne permet pas d'avoir le même point de Ω pour plusieurs points de $\mathcal{B}_\mathrm{B}$. De même que pour REMBO, on constate que RREMBO n'atteint pas la valeur du minimum de la fonction objectif. Cependant comme $\mathcal{A} \subset \mathcal{B}_\mathrm{B}$, certains points de $\mathcal{B}_\mathrm{B}$ n'ont pas d'image dans Ω. Par exemple, la Figure 2.10 montre un domaine $\Omega \subset \mathbb{R}^3$ projeté dans un domaine $\mathcal{B}_\mathrm{B} \subset \mathbb{R}^2$. Le domaine $\mathcal{A}$ est représenté en blanc. En réalité, tous les points noirs de $\mathcal{B}_\mathrm{B}$ n'ont pas d'image dans Ω par γ_B. Enfin, grâce à la construction spécifique de la matrice $\mathbf{A}$ dans HeSBO, l'application de retour est donnée par

$$\gamma_\mathrm{N}(\boldsymbol{u}) = \left[\eta_i \, u_{h_i}\right]_{i=1,\dots,d}, \tag{2.50}$$

où η_i et h_i sont définis dans la Section 2.2.3.2.

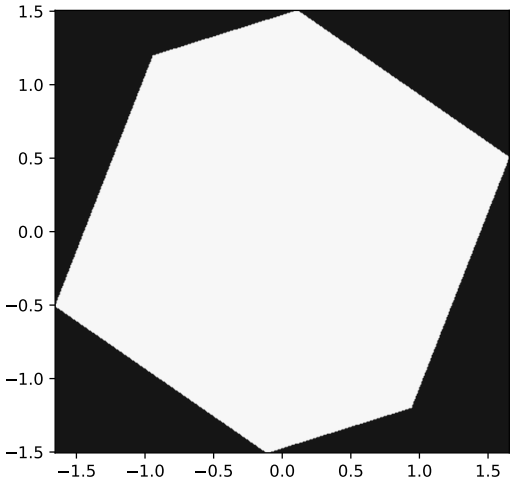


FIGURE 2.10 – Représentation de $\mathcal{B}_\mathrm{B}$ et de $\mathcal{A}$. En noir, les points de $\mathcal{B}_\mathrm{B}$ qui n'ont pas d'image dans Ω par γ_B ; en blanc, les points de $\mathcal{B}_\mathrm{B}$ qui ont une image dans Ω par γ_B.

Noyaux de covariance : Finalement, BINOIS et al. [15], WANG et al. [134] ont introduit de nouveaux noyaux de covariance pour mieux représenter la fonction objectif en prenant en compte le sous-espace linéaire et l'application retour. Le premier noyau de covariance utilise les distances entre les points dans le sous-espace linéaire $\mathcal{A}$. Celui-ci est noté $k_\mathcal{A}$ et est sélectionné parmi les noyaux de covariance déjà présentés dans le Tableau 2.1. Il est simple à mettre en place mais il ne prend pas en compte l'application de retour γ (c.-à-d., γ_W ou γ_B ou γ_N suivant l'algorithme choisi). Un deuxième noyau de covariance considère les distances dans Ω en utilisant l'application de retour γ pour être plus représentatif des données dans l'espace initial. Il s'exprime par $k_\gamma(\boldsymbol{u},\boldsymbol{u}') = k\big(\gamma(\boldsymbol{u}),\gamma(\boldsymbol{u}')\big)$ où k est un noyau de covariance du Tableau 2.1. Enfin, un troisième noyau peut être défini en travaillant avec l'application γ couplée à une projection orthogonale $p_\mathrm{A} : \mathbb{R}^d \mapsto \mathbb{R}^d$ de Ω sur $\{\boldsymbol{x} = \mathbf{A}^+\boldsymbol{u}, \forall \boldsymbol{u} \in \mathbb{R}^{d_e}\}$. Afin de contrebalancer l'effet de la projection orthogonale sur les distances en grande dimension, une distorsion est aussi introduite. Le noyau s'exprime donc comme $k_\Psi(\boldsymbol{u},\boldsymbol{u}') = k\big(\Psi(\boldsymbol{u}),\Psi(\boldsymbol{u}')\big)$ avec

$$\Psi(\boldsymbol{u}) = \left(1 + \frac{\|\gamma(\boldsymbol{u}) - \boldsymbol{x}'\|}{\|\boldsymbol{x}'\|}\right)\boldsymbol{x}', \quad \boldsymbol{x}' = \boldsymbol{x}\left[\max\left(1, \max_{1\le i\le d}(|x_i|)\right)\right]^{-1} \tag{2.51}$$

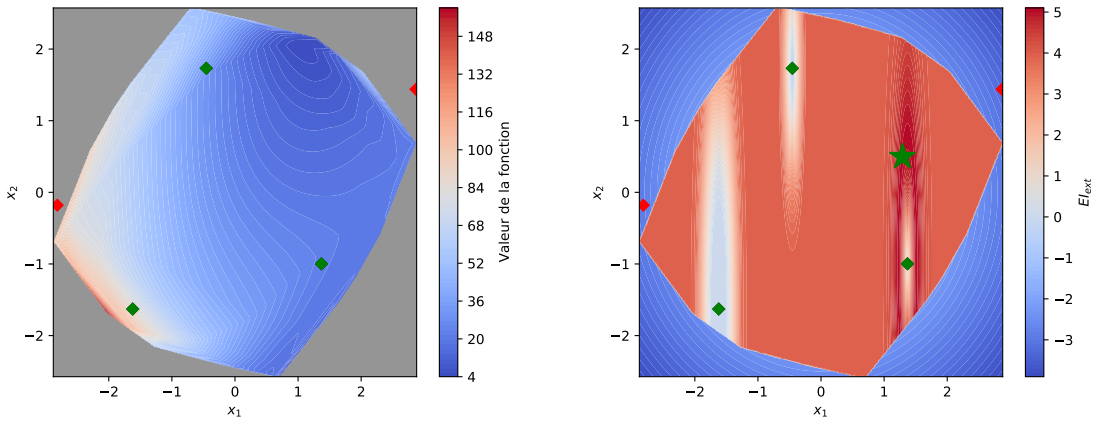
et $\boldsymbol{x} = p_\mathbf{A}\big(\gamma(\boldsymbol{u})\big)$. On remarque que $\boldsymbol{x} = p_\mathbf{A}\big(\gamma(\boldsymbol{u})\big)$ peut se réécrire $\boldsymbol{x} = \mathbf{A}^+\boldsymbol{u}$ dans le cas où on prend $\gamma = \gamma_\mathrm{B}$.

Optimisation de la fonction d'acquisition : Ce paragraphe ne concerne que la fonction γ_B introduite précédemment puisqu'elle est la seule à ne pas définir de points dans Ω pour tous les points de $\mathcal{B}_\mathrm{B}$. Pour pouvoir travailler dans l'hypercube $\mathcal{B}_\mathrm{B}$, BINOIS et al. [15] n'utilisent que les points du DoE ayant une image dans Ω pour construire les GP. De plus, ils définissent une fonction d'acquisition $\alpha_{\mathrm{EI}_{ext}}$ particulière dont le maximum est obligatoirement dans $\mathcal{B}_\mathrm{B}$. L'idée est d'avoir une fonction positive sur $\mathcal{A}$ et négative sur $\mathcal{B}_\mathrm{B} \setminus \mathcal{A}$. Ainsi, l'algorithme de maximisation ne proposera que des points de $\mathcal{A}$ ayant une fonction d'acquisition positive. Plus particulièrement, BINOIS et al. [15] utilise une pénalisation de l'EI donnée par $\alpha^{(l)}_{\mathrm{EI}_{ext}}(\boldsymbol{u}) = \alpha^{(l)}_\mathrm{EI}(\boldsymbol{u})$ si $\boldsymbol{u} \in \mathcal{A}$, c.-à-d. que le problème

quadratique (2.49) a une solution faisable. Sinon $\alpha^{(l)}_{\mathrm{EI}_{ext}}(\boldsymbol{u}) = -\|\boldsymbol{u}\|_2$. Elle s'écrit également sous la forme suivante :

$$\alpha^{(l)}_{\mathrm{EI}_{ext}}(\boldsymbol{u}) = \begin{cases} \alpha^{(l)}_{\mathrm{EI}}(\boldsymbol{u}) & , \text{si } \boldsymbol{u} \in \mathcal{A} \\ -\|\boldsymbol{u}\|_2 & , \text{sinon} \end{cases} \tag{2.52}$$

En effet, $\alpha^{(l)}_{\mathrm{EI}_{ext}}$ est positive si $\boldsymbol{u}$ appartient à $\mathcal{A}$ et négative sinon. Par exemple, La Figure 2.11a représente une fonction objectif projetée dans un sous-espace de dimension $d_e = 2$. Les zones grises



(a) Fonction objectif projetée.

(b) $\alpha_{\mathrm{EI}_{ext}}$.

FIGURE 2.11 – Projection d'une fonction objectif dans un sous-espace de dimension $d_e = 2$ et la représentation de la fonction d'acquisition correspondante $\alpha_{\mathrm{EI}_{ext}}$. La zone grise représente les points de $\mathcal{B} \notin \mathcal{A}$, les carrés verts sont les points du DoE dans $\mathcal{A}$, les carrés rouges sont les points du DoE n'appartenant pas à $\mathcal{A}$, l'étoile verte est le maximum de $\alpha_{\mathrm{EI}_{ext}}$.

montrent clairement qu'une grande partie du domaine $\mathcal{B}_{\mathrm{B}}$ n'a pas d'image dans l'espace de départ Ω. Dans la Figure 2.11b, on constate que les zones grises sont remplacées par des zones à valeurs négatives et croissantes vers l'intérieur du domaine ce qui permet une convergence vers des points ayant une image dans Ω. Ainsi, la maximisation de cette fonction d'acquisition donne un point qui a une image dans Ω (l'étoile verte sur la Figure 2.11b) ce qui permet l'évaluation sur la fonction objectif. Pour finir, on souligne que ce phénomène n'apparaît pas avec l'utilisation de γ_{N} et γ_{W} car on force l'appartenance à Ω dans la définition.

Algorithme : Pour finir, ces trois méthodes reposent sur une même forme d'algorithme dont chaque étape correspond à un des paragraphes précédents. L'algorithme 2.6 rappelle le processus BO étendu à la grande dimension dans le cadre de REMBO, RREMBO et HeSBO.

Ces trois méthodes ont montré de bons résultats sur des problèmes de grande dimension pouvant aller jusqu'au million de variables [15, 74, 134]. Parmi ces trois processus d'optimisation, NAYEBI et al. [74] ont montré que HeSBO était le choix le plus judicieux pour obtenir des convergences rapides et précises. De plus, BINOIS et al. [15] démontrent que l'utilisation de γ_{B}, k_Ψ et $\mathcal{B}_{\mathrm{B}}$ est préférable par rapport à γ_{W}, $k_{\mathcal{A}}$, k_γ et $\mathcal{B}_{\mathrm{W}}$. Cependant, la génération aléatoire de $\mathbf{A}^+$ peut conduire à un sous espace qui ne comprend pas le minimum de la fonction. Par exemple, la Figure 2.9 montre un sous-espace linéaire possible pour un problème d'optimisation en deux dimensions. On remarque clairement que ce sous-espace ne contient pas le minimum de la fonction objectif. Pour cela, WANG et al. [134] proposent de réaliser n processus d'optimisation en parallèle

Algorithme 2.6 Les méthodes REMBO, RREMBO ou HESBO.

entrée : une fonction objectif, un nombre maximal d'itérations max_nb_it, une dimension effective d_e

1: Générer la matrice $\mathbf{A}^+ \in \mathbb{R}^{d \times d_e}$ (Paragraphe 2.2.3.2)
2: Calculer le sous-espace de conception $\mathcal{B}$ (Paragraphe 2.2.3.2)
3: Générer le DoE $\mathcal{D}^{(0)}_{\mathbf{A},f}$ dans $\mathcal{B}$ grâce à γ
4: **pour** $l = 0$ **à** max_nb_it - 1 **faire**
5: Construire un GP avec $\mathcal{D}^{(l)}_{\mathbf{A},f}$ (Paragraphe 2.2.3.2)
6: Trouver $\boldsymbol{u}^{(l+1)}$ la solution du sous-problème d'enrichissement (2.31) (Paragraphe 2.2.3.2)
7: Estimer $\boldsymbol{x}^{(l+1)} = \gamma(\boldsymbol{u}^{(l+1)})$ (Paragraphe 2.2.3.2)
8: Évaluer la fonction objectif en $\boldsymbol{x}^{(l+1)}$
9: Mettre à jour le DoE
10: **fin pour**

sortie : Le meilleur point du DoE

avec une matrice $\mathbf{A}^+$ différente pour chaque instance et un nombre maximal d'évaluations global de max_nb_it. En outre, ces travaux considèrent également que la dimension efficace est connue ce qui n'est pas forcément le cas en général.

2.2.3.3 Optimisation Bayésienne reposant sur des régions de confiance, ou Trust region Bayesian optimization

Lorsque le nombre de variables de conception augmente, le nombre de points nécessaires à la construction du GP donnant de bonnes prédictions augmente exponentiellement. En réalité, il est impossible de prendre un compte un tel nombre de points. Avec un nombre de points plus restreint, la distance entre eux est plus grande et rend l'écart type du GP important. Ainsi, la plupart des fonctions d'acquisition vont se concentrer sur l'exploration du domaine ce qui ne permet pas de converger vers le minimum.

Pour résoudre cela, la méthode trust region implementation in kriging-based optimization with expected improvement (TRIKE) [101] et plus récemment la méthode TuRBO [33] intègrent une partie locale dans l'algorithme classique BO. TRIKE considère une unique région de confiance dont la taille est gérée avec un rapport entre l'EI du point ajouté et la réelle amélioration apportée par ce point. De plus, un critère d'arrêt de l'algorithme est aussi défini avec l'EI du point ajouté. Plus récemment, la méthode TuRBO, reposant aussi sur l'idée des régions de confiance, a été développée pour résoudre des problèmes de grande dimension. De plus, elle considère plusieurs régions de confiance en simultanée. C'est pourquoi, on s'intéressera plus particulièrement à TuRBO dans ce manuscrit.

Pour résoudre un problème d'optimisation avec TuRBO, on génère un hyper-rectangle, communément appelé région de confiance, de taille de base $\mathrm{L}^{(l)} \in \mathbb{R}^+$ autour du point du DoE ayant la plus faible valeur de fonction objectif $\boldsymbol{x}^{(l)}_{min}$ à l'itération l. La taille $\mathrm{L}^{(l)}_i$ de l'hyper-rectangle pour la i^e variable de conception est liée à la longueur de corrélation du GP $l^{(l)}_{f,i}$ et est donnée par

$$\mathrm{L}^{(l)}_i = \frac{l^{(l)}_{f,i}\mathrm{L}^{(l)}}{\left[\prod_{j=1}^{d} l^{(l)}_{f,i}\right]^{\frac{1}{d}}}. \tag{2.53}$$

Un processus BO classique est ensuite réalisé dans cette région de confiance. On note que la taille

de la région de confiance est très importante pour la convergence de l'algorithme. C'est pourquoi elle est mise à jour à chaque itération en fonction du nombre de succès τ_{suc} ou d'échecs τ_{fail} successifs à ajouter un point améliorant le minimum du DoE. Ainsi, après τ_{suc} succès consécutifs, on double la taille de la région de confiance (c.-à-d., $L^{(l+1)} = \min(L_{max}, 2L^{(l)})$). Sinon après τ_{fail} échecs consécutifs, on divise par deux la taille de la région de confiance (c.-à-d., $L^{(l+1)} = L^{(l)}/2$). Si $L^{(l+1)}$ devient plus petite que la taille minimale L_{min}, alors on régénère une région de confiance avec une taille $L^{(l+1)} = L^{(0)}$. Si aucun des deux cas n'est réalisé, la taille de la région de confiance reste la même (c.-à-d., $L^{(l+1)} = L^{(l)}$). Les valeurs de $L^{(0)}, L_{min}, L_{max}, \tau_{fail}$ et τ_{suc} sont des paramètres à choisir par l'utilisateur.

De plus, afin de conserver un caractère exploratoire, TuRBO maintient simultanément m régions de confiance en parallèle de taille de base $L_{min} \geq L_t \geq L_{max}$ avec un GP local par région $t \in \{1, \ldots, m\}$. Pour chaque région de confiance TR_t, on résout le sous-problème d'optimisation. Parmi les m points proposés, on garde celui proposant la fonction d'acquisition la plus forte, c.-à-d. le point solution du sous-problème d'optimisation suivant :

$$\left(t_{max}, x^{(l+1)}\right) \in \arg \max_{(t,\boldsymbol{x}) \in \{1,\ldots,m\} \times TR_t} \alpha_{f,t}^{(l)}(\boldsymbol{x}), \tag{2.54}$$

où $\alpha_{f,t}^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}$ est la fonction d'acquisition construite à partir du GP local de la région de confiance TR_t. Plus particulièrement, TuRBO utilise TS (voir (2.37)) comme fonction d'acquisition. Il est donc possible de générer $q \in \mathbb{N}^+$ points à ajouter au DoE à chaque itération en utilisant q réalisations du GP de la fonction objectif pour construire q fonctions d'acquisition différentes.

Pour résumer, la méthode itérative TuRBO est donnée par l'Algorithme 2.7. Cette méthode

Algorithme 2.7 La méthode TURBO.

entrée : une fonction objectif, un nombre maximal d'itérations max_nb_it, un nombre de régions de confiance m, une taille initiale $L^{(0)}$, une taille minimale L_{min} et maximale L_{max}, un taux de succès τ_{suc} et d'échecs τ_{fail}

1: Générer les m régions de confiance
2: Générer le DoE pour chaque région de confiance
3: Initialiser $n_{suc} = 0$ et $n_{fail} = 0$
4: **pour** $l = 0$ **à** max_nb_it - 1 **faire**
5: Construire le GP pour chaque région de confiance TR_t
6: Trouver $\boldsymbol{x}^{(l+1)}$ la solution du sous-problème d'enrichissement (2.54)
7: Évaluer la fonction objectif en $\boldsymbol{x}^{(l+1)}$
8: Mettre à jour le DoE de chaque région de confiance
9: Mettre à jour n_{suc} et n_{fail}
10: Calculer $L_t^{(l+1)}$ avec n_{suc}, n_{fail}, τ_{suc}, τ_{fail}, L_{min}, L_{max} et $L_t^{(0)}$
11: Mettre à jour le centrage des régions de confiance
12: **fin pour**

sortie : Le meilleur point du DoE global

permet de prendre en compte des problèmes de grande dimension en ne traitant que des régions de confiance avec un processus BO. Ceci permet de favoriser l'exploitation par rapport à l'exploration qui est privilégiée par les fonctions d'acquisition classiques. Néanmoins, la construction des GP se fait toujours dans l'espace initial de grande dimension. La méthode TuRBO ne permet malheureusement pas de prendre en compte des problèmes de très grande dimension, c.-à-d. plus d'une centaine de variables de conception, à la différence les méthodes RREMBO, REMBO et

HeSBO.

Pour conclure cette section sur la HDBO, on a introduit les trois grands défis s'opposant à l'utilisation de la méthode BO pour des problèmes d'optimisation de grande dimension. Premièrement, les GP classiques de grande dimension souffrent du manque de corrélation des points du DoE avec les points de Ω. Deuxièmement, l'estimation des hyper-paramètres peut s'avérer longue à cause de leur estimation par maximum de vraisemblance. En effet, leur nombre augmente avec la dimension ce qui rend cette estimation difficile. Troisièmement, l'exploration complète de l'espace de conception n'est pas possible de par sa taille. Les fonctions d'acquisition classiques favorisent trop l'exploration et ne permettent pas toujours de converger vers un optimum.

Ces défis sont en partie résolus grâce à des hypothèses sur la fonction objectif à optimiser ou en restreignant l'espace de recherche. On a introduit cinq méthodes existantes pour résoudre des problèmes d'optimisation de grande dimension grâce à la méthode BO. EGO-KPLS utilise un noyau de covariance spécifique pour réduire le temps nécessaire à l'estimation des hyper-paramètres mais l'optimisation est réalisée dans l'espace initial. REMBO, RREMBO et HeSBO font l'hypothèse que la fonction objectif ne dépend que de $d_e \ll d$ directions efficaces mais le sous-espace est aléatoirement choisi ce qui limite les performances de l'optimisation si il est mal choisi. Enfin, TuRBO considère des régions de confiance autour des meilleurs points du DoE mais réalise l'optimisation et la construction des GP dans l'espace initial ce qui peut s'avérer coûteux.

2.2.4 Conclusion

Dans cette section sur la BO, on a d'abord introduit le processus itératif d'enrichissement reposant sur les GP. Pour réaliser cet enrichissement adaptatif, on a présenté les principales fonctions d'acquisition qui permettent de mener le compromis exploration/exploitation. Ce processus ne peut cependant pas être utilisé tel quel pour des problèmes de grande dimension (c.-à.-d. souvent plus de 20 dimensions) à cause du temps nécessaire à la construction des GP et le peu d'information qu'ils comportent dans ce cas. On a donc introduit cinq méthodes de HDBO qui reposent soit sur une réduction linéaire de dimension réalisée de manière aléatoire, soit sur des régions de confiance définies dans l'espace de recherche. Ces méthodes ont montré de bons résultats mais la construction des GP reste parfois lente. De plus, à cause de la sélection aléatoire du sous-espace linéaire, certaines méthodes peuvent générer un sous espace ne contenant pas l'optimum. Toutefois, aucune des méthodes introduites dans cette section ne prend en compte les problèmes d'optimisation sous contraintes. C'est pourquoi, on introduit dans la section suivante les méthodes BO capables de résoudre des problèmes comprenant des contraintes.

2.3 Généralité sur l'optimisation Bayésienne avec contraintes

Dans cette section, on s'intéresse plus particulièrement aux problèmes d'optimisation sous contraintes mixtes, c.-à-d. qui comportent des contraintes d'égalité et d'inégalité comme suit :

$$\min_{\boldsymbol{x}\in\Omega} \left\{ f(\boldsymbol{x}) \text{ s.c. } \boldsymbol{g}(\boldsymbol{x}) \geq 0 \text{ et } \boldsymbol{h}(\boldsymbol{x}) = 0 \right\}, \tag{2.55}$$

où $f : \mathbb{R}^d \mapsto \mathbb{R}$ est la fonction objectif, $\boldsymbol{g} : \mathbb{R}^d \mapsto \mathbb{R}^m$ sont les fonctions des contraintes d'inégalité, $\boldsymbol{h} : \mathbb{R}^d \mapsto \mathbb{R}^p$ correspondent aux fonctions des contraintes d'égalité et Ω est un hypercube

de $\mathbb{R}^d$ représentant le domaine de conception considéré. On note $\boldsymbol{c} = [\boldsymbol{g}^\top, \boldsymbol{h}^\top]^\top$ l'ensemble des contraintes. Les fonctions f, $\boldsymbol{g}$ et $\boldsymbol{h}$ sont coûteuses à évaluer et ne possèdent aucune propriété utilisable autre que les valeurs $f(\boldsymbol{x})$, $\boldsymbol{g}(\boldsymbol{x})$ et $\boldsymbol{h}(\boldsymbol{x})$ pour tout $\boldsymbol{x} \in \Omega$. De la même manière que dans la Section 2.2, on construit $m+p+1$ GP pour $f, \boldsymbol{g}$ et $\boldsymbol{h}$ qui sont utilisés dans un processus BO d'enrichissement itératif jusqu'à convergence de l'algorithme. L'algorithme de base de la CBO est le même que celui de la BO classique (voir Algorithme 2.4) en remplaçant le sous-problème d'optimisation (2.31) par un sous-problème d'optimisation prenant en compte les contraintes. Dans la littérature [2, 9, 37, 43, 51, 63, 83, 84, 108, 112, 117], ce sous-problème d'optimisation peut être de deux types. D'une part, le sous-problème d'optimisation peut être sans contrainte [51, 83, 84, 112]. Il vise à optimiser une fonction de mérite qui rassemble les contraintes et la fonction objectif en une unique fonction. D'autre part, le sous-problème peut être contraint [2, 9, 63, 108]. Il aspire donc à optimiser l'une des fonctions d'acquisition introduites dans la Section 2.2 tout en respectant un critère de faisabilité pour chaque contrainte. La plupart des méthodes sont développées pour résoudre des problèmes d'optimisation comportant uniquement des contraintes d'inégalité. Pour prendre en compte des contraintes d'égalité, on remplace chaque contrainte d'égalité $h(\boldsymbol{x}) = 0$ par deux contraintes d'inégalité $h(\boldsymbol{x}) \geq \epsilon_c$ et $-h(\boldsymbol{x}) \leq \epsilon_c$. Toutefois, l'ajout de ces deux contraintes est problématique pour le processus d'optimisation car le nombre de contraintes est doublé et des contraintes antagonistes doivent être prises en compte [76].

Dans la suite, on va introduire et discuter des principales méthodes liées aux fonctions de mérite et aux critères de faisabilité. Ensuite, le couplage de ces méthodes avec les algorithmes HDBO est mis en avant.

2.3.1 Méthodes liées aux fonctions de mérite

On s'intéresse ici aux méthodes CBO à fonction de mérite, c.-à-d. que le sous-problème d'optimisation peut s'écrire comme :

$$\boldsymbol{x}^{(l+1)} \in \arg\max_{\boldsymbol{x}\in\Omega} \alpha_m^{(l)}\boldsymbol{x} \tag{2.56}$$

où $\alpha_m^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}$ est la fonction de mérite qui rassemble les informations du GP de la fonction objectif et des GP modélisant les contraintes $\boldsymbol{g}$ et $\boldsymbol{h}$. Cette formulation du sous-problème d'optimisation permet d'utiliser des algorithmes d'optimisation sans contrainte. Les quatre méthodes principales de cette catégorie de méthode CBO [51, 83, 84, 112] sont introduites ci-après. Elles prennent en compte des problèmes d'optimisation qui sont soit sous contraintes d'inégalité ou soit sous contraintes mixtes.

2.3.1.1 Amélioration faisable espérée ou expected feasible improvement

On cherche ici à trouver les zones de l'espace de conception qui ont le plus de chance d'être faisables grâce aux informations des GP. Pour les contraintes d'inégalité, l'amélioration faisable espérée, ou expected feasible improvement (EFI) [111] réalise une pénalisation de EI dans les zones de faible probabilité de faisabilité. Cette probabilité de faisabilité est analytique et est donnée,

pour chaque composante g_i de $\boldsymbol{g}$, par

$$\alpha^{(l)}_{\mathrm{PF},g_i}(\boldsymbol{x}) = \begin{cases} 0 & , \text{si } \hat{\sigma}^{(l)}_{g_i}(\boldsymbol{x}) = 0 \text{ et } \hat{\mu}^{(l)}_{g_i}(\boldsymbol{x}) < 0 \\ 1 & , \text{si } \hat{\sigma}^{(l)}_{g_i}(\boldsymbol{x}) = 0 \text{ et } \hat{\mu}^{(l)}_{g_i}(\boldsymbol{x}) \geq 0 \\ \mathbb{P}\left(y^{(l)}_{g_i,\boldsymbol{x}} \geq 0\right) = \Phi\left(\frac{\hat{\mu}^{(l)}_{g_i}(\boldsymbol{x})}{\hat{\sigma}^{(l)}_{g_i}(\boldsymbol{x})}\right) & , \text{sinon} \end{cases} \tag{2.57}$$

où $\Phi : \mathbb{R} \mapsto [0,1]$ est la fonction de répartition de la loi normale centrée réduite. En supposant que les composantes g_i de $\boldsymbol{g}$ sont indépendantes, on peut calculer la probabilité de faisabilité globale du point $\boldsymbol{x}$ comme suit :

$$\begin{aligned} \alpha^{(l)}_{\mathrm{PF},\boldsymbol{g}}(\boldsymbol{x}) &= \mathbb{P}\left(y^{(l)}_{g_1,\boldsymbol{x}} \geq 0 \cap \cdots \cap y^{(l)}_{g_m,\boldsymbol{x}} \geq 0\right) \\ &= \prod_{i=1}^{m} \mathbb{P}\left(y^{(l)}_{g_i,\boldsymbol{x}} \geq 0\right) \\ &= \prod_{i=1}^{m} \alpha^{(l)}_{\mathrm{PF},g_i}(\boldsymbol{x}) \end{aligned} \tag{2.58}$$

Finalement, la fonction de mérite EFI est une pénalisation de la fonction d'acquisition EI par la probabilité de faisabilité :

$$\alpha^{(l)}_{\mathrm{EFI},f,\boldsymbol{g}}(\boldsymbol{x}) = \alpha^{(l)}_{\mathrm{EI},f}(\boldsymbol{x})\alpha^{(l)}_{\mathrm{PF},\boldsymbol{g}}(\boldsymbol{x}) \tag{2.59}$$

De plus, on doit pouvoir définir la meilleure valeur de la fonction objectif dans le DoE pour pouvoir calculer EI. Pour cela, deux méthodes sont classiquement utilisées :

1. Si il n'y a pas de point faisable dans le DoE, alors $f^{(l)}_{min} = +\infty$, sinon $f^{(l)}_{min} = \min_{\boldsymbol{x} \in \mathcal{D}^{(l)}_f} \left\{ f(\boldsymbol{x}) \quad \text{s.c.} \quad \boldsymbol{g}(\boldsymbol{x}) \geq 0 \right\}$.

2. Si il n'y a pas de point faisable dans le DoE, alors $f^{(l)}_{min} = f\left(\boldsymbol{x}^{(l)}_{min}\right)$ où $\boldsymbol{x}^{(l)}_{min} \in \arg\min_{\boldsymbol{x} \in \mathcal{D}^{(l)}_f} \sum_{i=1}^{m} \left|\min\left(0, g_i(\boldsymbol{x})\right)\right|$, sinon $f^{(l)}_{min} = \min_{\boldsymbol{x} \in \mathcal{D}^{(l)}_f} \left\{ f(\boldsymbol{x}) \quad \text{s.c.} \quad \boldsymbol{g}(\boldsymbol{x}) \geq 0 \right\}$.

La méthode 1 cherche à supprimer la prise en compte de la fonction objectif tant qu'il n'y a pas de point faisable dans le DoE. Cela peut mener vers des zones faisables à forte valeur de fonction objectif. De plus, si les contraintes sont multi-modales, il est possible que l'on reste bloqué dans une zone sous-optimale. Au contraire, la méthode 2 prend en compte la valeur de la fonction objectif tout en minimisant la violation des contraintes. Ceci permet notamment de guider la recherche de zone faisable vers les zones à valeur de fonction objectif faible. En pratique, la méthode 1 est majoritairement utilisée.

La fonction de mérite EFI est analytique ce qui permet de l'utiliser plus simplement dans des problèmes de grande dimension. Toutefois, le produit dans l'expression de $\alpha^{(l)}_{\mathrm{PF},\boldsymbol{g}}$ peut être problématique lorsque le nombre de contraintes augmente. En effet, $\alpha^{(l)}_{\mathrm{PF},\boldsymbol{g}} < 1$ sur tout le domaine de conception hormis pour les points faisables du DoE. Ainsi, le produit d'un grand nombre de valeurs inférieures à 1 tend vers 0. La fonction de mérite s'annule donc presque partout ce qui rend l'optimisation de cette fonction impossible à réaliser. Malgré ce problème, cette fonction de mérite a été testée de nombreuses fois avec succès sur des problèmes d'optimisation industriels [49, 89]. Pour finir, elle est uniquement utilisable sur les problèmes d'optimisation sous contraintes d'inégalité.

2.3.1.2 La réduction progressive de l'incertitude ou stepwise uncertainty reduction

PICHENY [83] a étendu la fonction d'acquisition SUR pour résoudre les problèmes d'optimisation sous contraintes d'inégalité. Comme pour SUR sans contrainte, on cherche à ajouter le point faisable qui réduit le plus le volume espéré sous $f_{min}^{(l)}$. De la même manière que pour EFI, la méthode 1 est utilisée pour définir $f_{min}^{(l)}$. Avec un domaine faisable $\Omega_{\boldsymbol{g}}$ connu, ce volume est donné par

$$\mathbb{E}_{\Omega_{\boldsymbol{g}}}\left[y_{f,\boldsymbol{x}}^{(l)} \leq f_{min}^{(l)}\right]. \tag{2.60}$$

Comme $\Omega_{\boldsymbol{g}}$ n'est pas connu, on utilise l'information des GP modélisant les contraintes. Le volume espéré faisable sous $f_{min}^{(l)}$ s'exprime grâce aux probabilités de faisabilité des GP modélisant les contraintes :

$$\mathbb{E}_{\Omega}\left[\mathbb{P}\left(y_{f,\boldsymbol{x}}^{(l)} \leq f_{min}^{(l)} \cap y_{g_1,\boldsymbol{x}}^{(l)} \geq 0 \cap \cdots \cap y_{g_m,\boldsymbol{x}}^{(l)} \geq 0\right)\right]. \tag{2.61}$$

Grâce à l'indépendance des GP entre eux, ce volume peut s'écrire

$$\mathbb{E}_{\Omega}\left[\mathbb{P}\left(y_{f,\boldsymbol{x}}^{(l)} \leq f_{min}^{(l)}\right)\prod_{i=1}^{m}\mathbb{P}\left(y_{g_i,\boldsymbol{x}}^{(l)} \geq 0\right)\right]. \tag{2.62}$$

Finalement, comme pour SUR sans contrainte (2.40), il faut exprimer l'espérance de la distribution du volume sous

$$f_{min}^{(l),+} = \begin{cases} \min\left(f_{min}^{(l)}, y_{f,\boldsymbol{x}^+}^{(l)}\right), & \text{si } \forall i \in \{1,\dots,m\},\ y_{g_i,\boldsymbol{x}^+}^{(l)} \geq 0 \\ f_{min}^{(l)}, & \text{sinon} \end{cases} \tag{2.63}$$

si on ajoute le point $\boldsymbol{x}^+ \in \Omega$ au DoE. Finalement, la fonction de mérite SUR avec contraintes est donnée par l'opposé de l'espérance de ce volume :

$$\alpha_{\mathrm{SUR},f}^{(l)}(\boldsymbol{x}^+) = -\mathbb{E}_{\Omega}\left[\mathbb{P}\left(y_{f,\boldsymbol{x}}^{(l)} \leq f_{min}^{(l),+}\right)\prod_{i=1}^{m}\mathbb{P}\left(y_{g_i,\boldsymbol{x}}^{(l)} \geq 0\right)\right]. \tag{2.64}$$

A l'instar de la fonction d'acquisition SUR sans contrainte (2.40), cette fonction de mérite nécessite une intégration sur Ω pour être estimée. Ceci ne permet donc pas son utilisation pour des problèmes en grande dimension. De plus, le produit des probabilités de faisabilité devient problématique lorsque le nombre de contraintes augmente. En pratique, on utilise SUR avec moins de 4 contraintes d'inégalité [83, 105].

2.3.1.3 La recherche par prédiction d'entropie avec contraintes ou predictive entropy search with constraints

La fonction d'acquisition PES [51] est aussi étendue aux contraintes d'inégalité. Comme pour la PES sans contrainte, on tente d'estimer l'information sur la localisation du minimum faisable. On mesure cette information grâce à l'opposé de l'entropie différentielle de la variable aléatoire $\boldsymbol{x}^{(l),*}$, $\mathrm{H}\left[\mathbb{P}\left(\boldsymbol{x}^*|\mathcal{D}^{(l)}\right)\right]$, où

$$\boldsymbol{x}^{(l),*} \in \arg\max_{\boldsymbol{x}\in\Omega}\left\{y_{f,\boldsymbol{x}}^{(l)} \text{ s.c. } y_{g_i,\boldsymbol{x}}^{(l)} \geq 0\ \forall i \in \{1,\dots,m\}\right\}, \tag{2.65}$$

$\mathcal{D}^{(l)} = \mathcal{D}_f^{(l)} \cup \mathcal{D}_{g_1}^{(l)} \cup \cdots \cup \mathcal{D}_{g_m}^{(l)}$ et $\mathbb{P}\left(\boldsymbol{x}^*|\mathcal{D}^{(l)}\right)$ la loi de probabilité de $\boldsymbol{x}^{(l),*}$.

Ensuite, on ajoute au DoE le point $\boldsymbol{x}$ qui maximise l'espérance de la réduction de cette quantité, c.-à-d. qui maximise

$$\alpha^{(l)}_{\text{PESC}}(\boldsymbol{x}) = \text{H}\left[\mathbb{P}\left(\boldsymbol{x}^*|\mathcal{D}^{(l)}\right)\right] - \mathbb{E}_{\boldsymbol{y}^{(l)}_{\boldsymbol{x}}}\left[\text{H}\left[\mathbb{P}\left(\boldsymbol{x}^*|\mathcal{D}^{(l)} \cup (\boldsymbol{x}, \boldsymbol{y}^{(l)}_{\boldsymbol{x}})\right)\right]\right], \tag{2.66}$$

où $\boldsymbol{y}^{(l)}_{\boldsymbol{x}} = \left[y^{(l)}_{f,\boldsymbol{x}}, y^{(l)}_{g_1,\boldsymbol{x}}, \cdots, y^{(l)}_{g_m,\boldsymbol{x}}\right]^\top$. En réalité, il est plus simple de travailler avec l'expression équivalente donnée par :

$$\alpha^{(l)}_{\text{PESC}}(\boldsymbol{x}) = \text{H}\left[\mathbb{P}\left(\boldsymbol{y}|\mathcal{D}^{(l)}, \boldsymbol{x}\right)\right] - \mathbb{E}_{\boldsymbol{x}^{(l),*}}\left[\text{H}\left[\mathbb{P}\left(\boldsymbol{y}|\mathcal{D}^{(l)}, \boldsymbol{x}, \boldsymbol{x}^{(l),*}\right)\right]\right], \tag{2.67}$$

avec $\mathbb{P}\left(\boldsymbol{y}|\mathcal{D}^{(l)}, \boldsymbol{x}, \boldsymbol{x}^{(l),*}\right)$ la loi du vecteur des GP de f et $\boldsymbol{g}$ en $\boldsymbol{x}$ conditionnés par $\mathcal{D}^{(l)}$ et pour lequel on impose que $\boldsymbol{x}^{(l),*}$ soit le minimum global. Pour estimer la fonction de mérite recherche entropique prédictive avec contraintes, ou predictive entropy search with constraints (PESC), on réalise une étape d'estimation de la moyenne par échantillonnage de $\boldsymbol{x}^*$ suivant la loi $\mathbb{P}\left(\boldsymbol{x}^*|\mathcal{D}^{(l)}\right)$. Celle-ci est suivie d'une étape de propagation des attentes, ou (expectation propagation) [70]. Enfin, une marginalisation des hyper-paramètres du GP est réalisée. Le lecteur peut se référer aux travaux de HERNÁNDEZ-LOBATO et al. [51] pour plus informations sur les trois étapes succinctement introduites ici.

PESC a montré de bons résultats comparés à EFI et SUR mais il n'est pas possible de l'utiliser en grande dimension à cause des phases d'estimation nécessaires à son calcul. De plus, elle ne prend en compte que les contraintes d'inégalité ce qui limite son utilisation aux seuls problèmes d'optimisation sous contraintes d'inégalité.

2.3.1.4 Lagrangien augmenté pour l'optimisation Bayésienne, ou augmented Lagrangian for Bayesian optimization

Dans la contexte de l'optimisation sous contraintes mixtes, la méthode du Lagrangien augmenté, ou augmented Lagrangian (AL) [76] est classique. On introduit d'abord l'expression du AL :

$$\text{AL}(\boldsymbol{x}, \boldsymbol{\lambda_h}, \boldsymbol{\lambda_g}, \boldsymbol{s}, \rho) = f(\boldsymbol{x}) + \boldsymbol{\lambda_h}^\top \boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{\lambda_g}^\top \left(\boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{s}\right) + \frac{1}{2\rho}\left[\sum_{j=1}^{p} h_j(\boldsymbol{x})^2 + \sum_{j=1}^{m}\left(s_j - g_j(\boldsymbol{x})\right)^2\right] \tag{2.68}$$

où $\boldsymbol{\lambda_h} \in \mathbb{R}^{+^p}$ et $\boldsymbol{\lambda_g} \in \mathbb{R}^{+^m}$ sont les multiplicateurs de Lagrange, $\rho \in \mathbb{R}^+$ est un paramètre de pénalisation et $\boldsymbol{s} \in \mathbb{R}^{+^m}$ sont des variables d'écart changeant les contraintes d'inégalité en contraintes d'égalité. En définissant $\boldsymbol{c} = \left[-\boldsymbol{g}^\top, \boldsymbol{h}^\top\right]^\top$, $\boldsymbol{\lambda} = \left[\boldsymbol{\lambda_g}^\top, \boldsymbol{\lambda_h}^\top\right]^\top$ et en augmentant la dimension de $\boldsymbol{s}$ tel que $s_{m+1} = \cdots = s_{m+p} = 0$, on écrit le AL plus simplement :

$$\text{AL}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{s}, \rho) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \left(\boldsymbol{c}(\boldsymbol{x}) + \boldsymbol{s}\right) + \frac{1}{2\rho}\sum_{j=1}^{m+p}\left(c_j(\boldsymbol{x}) + s_j\right)^2. \tag{2.69}$$

Finalement, on résout le problème d'optimisation sous contraintes de manière itérative en utilisant le AL et en ajoutant une tolérance $\epsilon \in \mathbb{R}^+$ sur les contraintes d'égalité. Cette méthode itérative est donnée par l'Algorithme 2.8 pour lequel on estime les variables d'écart à chaque itération.

Dans la méthode du Lagrangien augmenté pour l'optimisation Bayésienne, ou augmented Lagrangian for Bayesian optimization (ALBO) [84], l'idée est d'étendre la méthode du AL dans le cadre de la CBO en minimisant le AL grâce à une méthode dérivée de la BO. Pour cela, la distribu-

Algorithme 2.8 La méthode AL.

entrée : une fonction objectif f, les contraintes $\boldsymbol{c}$, les multiplicateurs de Lagrange $\boldsymbol{\lambda}^{(0)}$, le paramètre de pénalisation $\rho^{(0)}$, un nombre d'itérations max_nb_it, une tolérance ϵ

1: **pour** $l = 0$ **à** max_nb_it - 1 **faire**
2: Définir la fonction des variables d'écart $\boldsymbol{s}^{(l)}(\boldsymbol{x}) = \max\left(\boldsymbol{0}, -\boldsymbol{\lambda}^{(l)}\rho^{(l)} - \boldsymbol{c}(\boldsymbol{x})\right)$
3: Soit $\boldsymbol{x}^{(l+1)}$ la solution de $\arg\min_{\boldsymbol{x}\in\Omega} \mathrm{AL}\left(\boldsymbol{x}, \boldsymbol{\lambda}^{(l)}, \boldsymbol{s}^{(l)}, \rho^{(l)}\right)$
4: Mettre à jour les multiplicateurs de Lagrange

$$\lambda_i^{(l+1)} = \lambda_i^{(l)} + \frac{1}{\rho^{(l)}}\left(c_i\left(\boldsymbol{x}^{(l+1)}\right) + s_i^{(l)}\right);\ \forall i \in \{1, \ldots, m+p\}$$

5: Mettre à jour le paramètre de pénalisation
6: **si** $\boldsymbol{c}_{1:m}(\boldsymbol{x}^{(l+1)}) \leq 0$ **et** $|\boldsymbol{c}_{m+1:m+p}(\boldsymbol{x}^{(l+1)})| \leq \epsilon$ **alors**
7: $\rho^{(l+1)} = \rho^{(l)}$
8: **sinon**
9: $\rho^{(l+1)} = \frac{1}{2}\rho^{(l)}$
10: **fin si**
11: **fin pour**

sortie : le dernier point trouvé $\boldsymbol{x}^{\text{max\_nb\_it}}$

tion du AL en $\boldsymbol{x}$ est définie à partir des GP modélisant la fonction objectif et des contraintes :

$$\begin{aligned}\mathrm{AL}_{\boldsymbol{x}}^{(l)} \sim \mathbb{P}\left(\mathrm{AL}|\mathcal{D}^{(l)}, \boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{s}, \rho\right) = \mathbb{P}\left(y_f|\mathcal{D}_f^{(l)}, \boldsymbol{x}\right) + \sum_{i=1}^{m+p} \lambda_i \left(\mathbb{P}\left(y_{c_i}|\mathcal{D}_{c_i}^{(l)}, \boldsymbol{x}\right) + s_i\right) \\ + \frac{1}{2\rho}\sum_{i=1}^{m+p}\left(\mathbb{P}\left(y_{c_i}|\mathcal{D}_{c_i}^{(l)}, \boldsymbol{x}\right) + s_i\right)^2.\end{aligned} \tag{2.70}$$

On souligne que la distribution du AL en $\boldsymbol{x}$ n'est pas gaussienne à cause du carré du GP modélisant les contraintes présent dans cette expression.

On reprend ensuite l'Algorithme 2.8 en remplaçant, à chaque itération, la minimisation du AL par la maximisation de l'amélioration espérée de la variable aléatoire $\mathrm{AL}_{\boldsymbol{x}}^{(l)}$. Cette mesure définit la fonction de mérite suivante :

$$\alpha_{\mathrm{ALBO}}^{(l)}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{s}, \rho) = \mathbb{E}\left[\mathbb{I}(\boldsymbol{x})\right] = \mathbb{E}\left[\max\left(0, \mathrm{AL}_{min}^{(l)} - \mathrm{AL}_{\boldsymbol{x}}^{(l)}\right)\right], \tag{2.71}$$

où $\mathrm{AL}_{min}^{(l)}$ est le minimum du AL dans le DoE. Cette méthode CBO, nommée ALBO, est finalement donnée par l'Algorithme 2.9 pour lequel l'estimation des variables d'écart est exprimée grâce aux moyennes des GP modélisant les contraintes. L'utilisation de l'amélioration espérée du AL dans cet algorithme permet, comme pour l'EI, d'explorer les zones de forte incertitude des GP. Toutefois, l'expression de $\alpha_{\mathrm{ALBO}}^{(l)}$ n'est pas analytique et nécessite une phase d'intégration pour être estimée. De plus, PICHENY et al. [84] considère la fonction objective comme connue et non couteuse. Couplé à la phase d'estimation, cette dernière caractéristique interdit son utilisation en grande dimension lorsque la fonction objective est aussi modélisée par un GP, la méthode ALBO a montré de très bons résultats sur des cas tests d'optimisation à contraintes mixtes comparés aux autres fonctions de mérite précédemment introduites [84].

On a présenté ici les quatre principales méthodes CBO à fonction de mérite. EFI est la seule qui soit analytique et donc utilisable en grande dimension. Toutefois, le produit des probabili-

Algorithme 2.9 La méthode ALBO.

entrée : une fonction objectif f, les contraintes $\boldsymbol{c}$, les multiplicateurs de Lagrange $\boldsymbol{\lambda}^{(0)}$, le paramètre de pénalisation $\rho^{(0)}$, un nombre d'itérations max_nb_it, une tolérance ϵ, un DOE initial des contraintes et de la fonction objectif $\mathcal{D}^{(l)}$

1: **pour** $l = 0$ **à** max_nb_it - 1 **faire**
2: Construire les GP modélisant les contraintes et la fonction objectif
3: Définir la fonction des variables d'écart $\boldsymbol{s}^{(l)}(\boldsymbol{x}) = \max\left(\boldsymbol{0}, -\boldsymbol{\lambda}^{(l)}\rho^{(l)} - \hat{\boldsymbol{\mu}}_{\boldsymbol{c}}(\boldsymbol{x})\right)$
4: Soit $\boldsymbol{x}^{(l+1)} \in \arg\max_{\boldsymbol{x}\in\Omega} \alpha^{(l)}_{\text{ALBO}}(\boldsymbol{x}, \boldsymbol{\lambda}^{(l)}, \boldsymbol{s}^{(l)}, \rho^{(l)})$
5: Évaluer la fonction objectif et les contraintes en $\boldsymbol{x}^{l+1}$
6: Ajouter le point $\boldsymbol{x}^{(l+1)}$ au DoE
7: Mettre à jour les multiplicateurs de Lagrange

$$\lambda_i^{(l+1)} = \lambda_i^{(l)} + \frac{1}{\rho^{(l)}}\left(c_i\left(\boldsymbol{x}^{(l+1)}\right) + s_i^{(l)}\right);\ \forall i \in \{1, \dots, m+p\}$$

8: Mettre à jour le paramètre de pénalisation
9: **si** $\boldsymbol{c}_{1:m}(\boldsymbol{x}^{(l+1)}) \leq 0$ **et** $|\boldsymbol{c}_{m+1:m+p}(\boldsymbol{x}^{(l+1)})| \leq \epsilon$ **alors**
10: $\rho^{(l+1)} = \rho^{(l)}$
11: **sinon**
12: $\rho^{(l+1)} = \frac{1}{2}\rho^{(l)}$
13: **fin si**
14: **fin pour**

sortie : le meilleur point du DoE

tés de faisabilité n'autorise pas son utilisation avec un grand nombre de contraintes. De plus, cette fonction de mérite n'est définie que pour les contraintes d'inégalité. Les autres fonctions de mérite SUR, PESC et ALBO ne sont pas analytiques et ne peuvent donc pas être utilisées en grande dimension. En outre, seul ALBO est défini pour des problèmes à contraintes mixtes, c.-à-d. comportant des contraintes d'égalité et d'inégalité. Pour conclure, on constate qu'aucune des fonctions de mérite introduites ici ne peut être utilisée sur des problèmes de grande dimension comportant un grand nombre de contraintes.

2.3.2 Méthodes liées à un critère de faisabilité

On s'intéresse ici aux méthodes CBO à critère de faisabilité, c.-à-d. que le sous-problème d'optimisation s'écrit sous la forme :

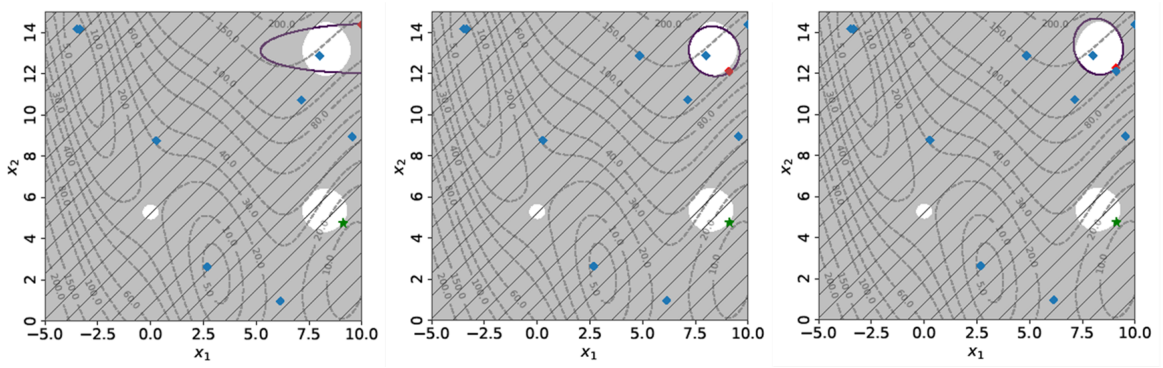
$$\boldsymbol{x}^{(l+1)} \in \arg\max_{\boldsymbol{x}\in\Omega}\left\{\alpha_f^{(l)} \quad \text{s.c.} \quad \boldsymbol{x} \in \Omega_{\boldsymbol{h}}^{(l)} \cap \Omega_{\boldsymbol{g}}^{(l)}\right\} \tag{2.72}$$

où $\alpha_f^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}$ est l'une des fonctions d'acquisition présentées dans la Section 2.2. De la même manière que pour EFI et SUR, $f_{min}^{(l)}$ est défini grâce à la méthode 1 ou 2 introduite dans la Section 2.3.1.1. $\Omega_{\boldsymbol{g}}^{(l)}$ et $\Omega_{\boldsymbol{h}}^{(l)}$ sont respectivement les domaines faisables définis par les critères de faisabilité $\boldsymbol{\alpha}_{\boldsymbol{g}}^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}^m$ et $\boldsymbol{\alpha}_{\boldsymbol{h}}^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}^p$. On souligne que les critères de faisabilité de $\boldsymbol{\alpha}_{\boldsymbol{g}}^{(l)}$ et $\boldsymbol{\alpha}_{\boldsymbol{h}}^{(l)}$ ne sont pas forcément les mêmes que ceux de $\boldsymbol{g}$ et $\boldsymbol{h}$. Par exemple, on peut avoir $\Omega_{\boldsymbol{h}}^{(l)} = \left\{\boldsymbol{x}, \boldsymbol{\alpha}_{\boldsymbol{h}}^{(l)}(\boldsymbol{x}) \leq 0\right\}$. En utilisant des fonctions d'acquisition et des critères de faisabilité analytiques, l'utilisation de ces méthodes pour résoudre des problèmes de grande dimension est plus simple que pour les méthodes CBO à fonction de mérite non-analytique comme PESC, SUR et ALBO.

Les trois principaux critères de faisabilité [2, 9, 63, 108] sont introduits ci-après. Ils prennent en compte les problèmes sous contraintes d'inégalité ou mixtes.

2.3.2.1 Super efficient global optimization et Super efficient global optimization with mixture of experts

Les méthodes super efficient global optimization (SEGO) et super efficient global optimization coupled with mixture of experts (SEGOMOE) [9, 108] utilisent l'information des moyennes des GP modélisant les contraintes comme critère de faisabilité $\boldsymbol{\alpha}_{\boldsymbol{c}}^{(l)}(\boldsymbol{x}) = \hat{\boldsymbol{\mu}}_{\boldsymbol{c}}^{(l)}(\boldsymbol{x})$ avec $\Omega_{\boldsymbol{h}}^{(l)} = \left\{x, \boldsymbol{\alpha}_{\boldsymbol{h}}^{(l)}(\boldsymbol{x}) = 0\right\}$ et $\Omega_{\boldsymbol{g}}^{(l)} = \left\{x, \boldsymbol{\alpha}_{\boldsymbol{g}}^{(l)}(\boldsymbol{x}) \geq 0\right\}$. En effet, plus on ajoute de points au DoE, plus la moyenne des GP est proche de la fonction objectif. Ainsi, en ajoutant de manière itérative des points aux GP, on peut trouver un domaine faisable grâce à ce critère de faisabilité. Par exemple, sur la Figure 2.12, on montre les domaines faisables (le réel et le prédit) et les courbes de niveau de la fonction objectif pour le problème problème de Branin modifié, ou modified Branin problem (MB) [79]. Le



(a) Itération 1. (b) Itération 4. (c) Itération Finale.

FIGURE 2.12 – Représentation du domaine faisable de SEGO (pour trois itérations) pour le problème Branin modifié. La zone hachurée est le domaine non faisable de SEGO, la zone grise montre le domaine non faisable réel et les courbes en pointillé sont les courbes de niveau de la fonction objectif. Les carrés bleus représentent le DOE courant tandis que l'étoile verte indique l'optimum global. Le carré rouge est le point ajouté au DOE.

domaine non faisable réel est représenté par la couleur grise, c.-à-d. que le domaine faisable est formé des trois boules blanches distinctes, et le domaine non faisable prédit est représenté par l'aire hachurée. On remarque ici que le domaine faisable prédit existe et permet de détecter la boule du domaine faisable en haut à droite du domaine de conception. Les trois itérations de la Figure 2.12 montrent l'évolution du domaine faisable prédit qui permet à terme de trouver un point faisable dans l'espace de conception. En effet, le dernier point ajouté au DoE (voir Figure 2.12c) est dans la zone faisable en haut à droite de l'espace de conception. Cette méthode a été testée avec succès sur de nombreux cas tests aéronautiques [7, 8, 9, 108].

Malgré ces bons résultats, la restriction du critère de faisabilité aux moyennes des GP dans le sous-problème d'optimisation (2.72) peut tromper le processus d'optimisation. En réalité, pendant les premières phases de l'optimisation, le DoE est encore pauvre en information, c.-à-d. qu'il ne comporte que peu de points, et ne permet pas de construire des GP dont les moyennes représentent bien les contraintes. A cause de la grande incertitude sur les GP modélisant $\boldsymbol{g}$ et $\boldsymbol{h}$, l'utilisation seule des fonctions de moyenne $\hat{\boldsymbol{\mu}}_{\boldsymbol{g}}^{(l)}$ et $\hat{\boldsymbol{\mu}}_{\boldsymbol{h}}^{(l)}$ peut conduire à considérer que la plus grande partie du domaine de conception est non-faisable. Ainsi, une grande partie du domaine de

conception peut ne pas être explorée. Dans ce cas, le processus d'enrichissement itératif devient très local et peut ignorer certaines zones du domaine de conception. Par exemple, en utilisant de nouveau la Figure 2.12, on remarque que le domaine faisable prédit ne couvre pas deux des trois zones faisables. A cause de cela, SEGO, ou SEGOMOE, ne va ajouter des points que dans la zone faisable prédite ce qui ne permet pas d'explorer la zone faisable où se trouve le minimum global. Pour résoudre ce problème, LAM et al. [63] ont introduit des critères de faisabilité prenant en compte l'incertitude du GP.

2.3.2.2 Super efficient global optimization with expected violation

AUDET et al. [2] introduisent un critère de faisabilité pour les contraintes d'inégalité utilisant l'écart type des GP modélisant les contraintes $\hat{\boldsymbol{\sigma}}_{\boldsymbol{g}}^{(l)}$. Ils prennent ainsi en compte l'incertitude de construction des GP dans le critère de faisabilité. Ce critère définit une zone faisable lorsque la valeur de la violation espérée, ou expected violation (EV) est proche de zéro. Le critère EV est donné par :

$$\alpha_{\mathrm{EV},g_i}^{(l)}(\boldsymbol{x}) = \begin{cases} 0, & \text{si } \hat{\sigma}_{g_i}^{(l)}(\boldsymbol{x}) = 0 \text{ et } \hat{\mu}_{g_i}^{(l)} \geq 0 \\ -\hat{\mu}_{g_i}^{(l)}, & \text{si } \hat{\sigma}_{g_i}^{(l)}(\boldsymbol{x}) = 0 \text{ et } \hat{\mu}_{g_i}^{(l)} \leq 0 \\ \left(0 - \hat{\mu}_{g_i}^{(l)}(\boldsymbol{x})\right)\Phi\left(\frac{0-\hat{\mu}_{g_i}^{(l)}(\boldsymbol{x})}{\hat{\sigma}_{g_i}^{(l)}(\boldsymbol{x})}\right) + \hat{\sigma}_{g_i}^{(l)}(\boldsymbol{x})\phi\left(\frac{0-\hat{\mu}_{g_i}^{(l)}(\boldsymbol{x})}{\hat{\sigma}_{g_i}^{(l)}(\boldsymbol{x})}\right), & \text{sinon} \end{cases} \tag{2.73}$$

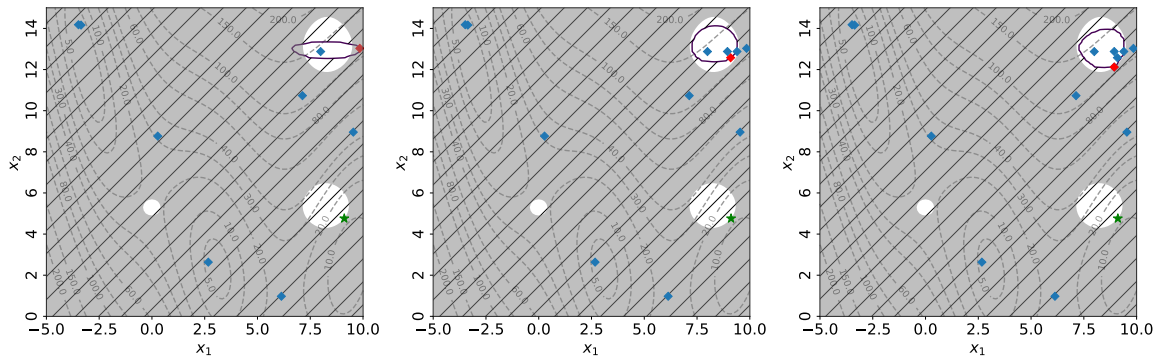
où g_i représente la i$^{\text{e}}$ composante de $\boldsymbol{g}$, $\Phi : \mathbb{R} \mapsto [0,1]$ est la fonction de répartition et $\phi : \mathbb{R} \mapsto \mathbb{R}$ est la densité de probabilité de la loi normale centrée réduite. En réalité, on définit un domaine faisable prédit par $\Omega_{\boldsymbol{g}}^{(l)} = \left\{\boldsymbol{x}\ ;\ \boldsymbol{\alpha}_{\mathbf{EV},\boldsymbol{g}}^{(l)}(\boldsymbol{x}) - \boldsymbol{\epsilon}_{\boldsymbol{g}} \leq 0\right\}$ où $\boldsymbol{\epsilon}_{\boldsymbol{g}} \in \mathbb{R}^{+^d}$ est choisi par l'utilisateur. Classiquement, on choisit $\boldsymbol{\epsilon}_{\boldsymbol{g}} = 10^{-3}$. En analysant ce critère, on remarque que les points avec une grande incertitude, c.-à-d. un grand écart type, sont écartés ainsi que les points avec une moyenne ne respectant pas la contrainte. Ainsi, ce critère utilise l'incertitude pour ne calculer des points que s'ils ont une grande probabilité d'être faisable. Comme SEGO, super efficient global optimization using the expected violation (SEGO-EV) va se concentrer sur une unique zone faisable sans faire d'exploration du domaine.

Par exemple, la Figure 2.13 montre trois itérations de la méthode SEGO-EV. On constate que la zone faisable prédite est très restreinte. Ceci est dû au manque de points dans le DOE qui engendre une grande incertitude dans le domaine. De plus, en ajoutant des points, on remarque que la zone faisable prédite est incluse dans la zone faisable réelle. Ceci montre bien que cette méthode est adaptée à l'optimisation prudente, c.-à-d. qui cherche à évaluer le moins possible de points non faisables. Toutefois, cette méthode n'explore pas le domaine de conception ce qui ne permet pas de trouver la zone faisable optimale, c.-à-d. qui contient l'optimum global, en bas à droite du domaine sur cet exemple.

2.3.2.3 Super efficient global optimization with the upper trust bounds

Le critère de faisabilité borne supérieure de confiance, ou upper trust bound (UTB) [63] tente de favoriser l'exploration du domaine de conception en relâchant les contraintes. LAM et al. [63] ont définit UTB pour les contraintes d'inégalité comme suit :

$$\alpha_{\mathrm{UTB},g_i}^{(l)}(\boldsymbol{x}) = \hat{\mu}_{g_i}^{(l)}(\boldsymbol{x}) + \tau_{g_i}^{(l)}\hat{\sigma}_{g_i}^{(l)}(\boldsymbol{x}) \tag{2.74}$$



(a) Itération 1. (b) Itération 4. (c) Itération Finale.

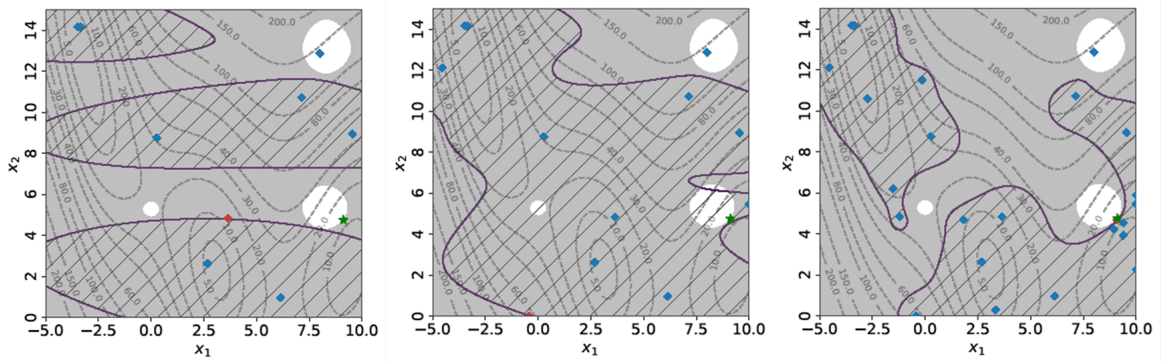
FIGURE 2.13 – Représentation du domaine faisable de SEGO-EV (pour trois itérations) pour le problème Branin modifié. La zone hachurée est le domaine non faisable de SEGO-EV, la zone grise montre le domaine non faisable réel et les courbes en pointillé sont les courbes de niveau de la fonction objectif. Les carrés bleus représentent le DOE courant tandis que l'étoile verte indique l'optimum global. Le carré rouge est le nouveau point ajouté au DOE.

où $\tau_{g_i} \in \mathbb{R}^+$. Le domaine de faisabilité prédit s'exprime donc comme $\Omega_{\boldsymbol{g}}^{(l)} = \left\{\boldsymbol{x}, \boldsymbol{\alpha}_{\mathbf{UTB},\boldsymbol{g}}^{(l)}(\boldsymbol{x}) \geq 0\right\}$. En général, on choisit $\tau_{g_i} = 3$ ce qui correspond approximativement à intervalle de confiance à 99% pour chaque point $\boldsymbol{x}$ du domaine Ω. On remarque également que UTB est optimiste sur le domaine de faisabilité, c.-à-d. que la zone de faisabilité est grande comparée à celle définie par SEGO. Cette propriété permet à la méthode super efficient global optimization using the upper trust bound (SEGO-UTB) d'être plus exploratoire et donc plus adaptée aux problèmes à zones de faisabilité disjointes. Toutefois, ce caractère exploratoire ralentit la convergence en diminuant l'exploitation des données.

De plus, le paramètre τ_{g_i} s'interprète comme un seuil de doute en la fonction de moyenne $\hat{\mu}_{g_i}^{(l)}$. En effet, si τ_{g_i} est proche de 0 alors on retrouve le critère SEGO qui ne prend pas en compte l'incertitude des GP. Ainsi on fait totalement confiance aux prédictions fournies pas les GP. Sinon plus τ_{g_i} est grand, plus on prend en compte l'incertitude et plus on autorise l'exploration. On considère donc que la moyenne ne fournit pas une bonne prédiction et il est nécessaire d'explorer le domaine de conception.

Par exemple, la Figure 2.14 montre trois itérations de l'algorithme SEGO-UTB. On remarque clairement que la zone faisable prédite est beaucoup plus grande que la zone faisable de SEGO ou SEGO-EV (voir Figures 2.12 et 2.13) ce qui accorde une plus grande exploration. On voit également qu'à la dernière itération, SEGO-UTB converge vers l'optimum global ce qui n'est pas le cas de SEGO et SEGO-EV. Toutefois, SEGO et SEGO-EV convergent plus rapidement que SEGO-UTB vers un optimum local comme le montre le nombre de points plus important dans la Figure 2.14c comparé aux Figures 2.12c et 2.13c.

On a introduit les trois principales méthodes CBO à critère de faisabilité qui ne poursuivent pas le même but. Premièrement, SEGO et SEGOMOE réalisent une exploitation massive de la moyenne des GP modélisant les contraintes. Ils ne réalisent pas d'exploration des contraintes et peuvent donc être bloqués dans une zone sous-optimale, c.-à-d. une zone faisable qui ne contient pas le minimum global. Deuxièmement, SEGO-EV utilise l'information d'incertitude pour empêcher l'algorithme d'explorer dans les zones avec une grande incertitude. Cette méthode a une tendance encore plus locale que SEGO et SEGOMOE et ne fait pas d'exploration du domaine. Toutefois, ce critère de faisabilité est utile pour réaliser une BO dite prudente, c.-à-d. qui ne doit pas évaluer



(a) Itération 1. (b) Itération 4. (c) Itération Finale.

FIGURE 2.14 – Représentation du domaine faisable de SEGO-UTB (pour trois itérations) pour le problème Branin modifié. La zone hachurée est le domaine non faisable de SEGO-UTB, la zone grise montre le domaine non faisable réel et les courbes en pointillé sont les courbes de niveau de la fonction objectif. Les carrés bleus représentent le DOE courant tandis que l'étoile verte indique l'optimum global. Le carré rouge est le nouveau point ajouté au DOE.

des points trop infaisables, e.g. pour ne pas endommager le matériel. Troisièmement, SEGO-UTB tire aussi profit de l'information d'incertitude pour aller visiter les zones du domaine qui peuvent cacher une zone faisable. Cependant, la caractère exploratoire de ce critère peut demander beaucoup de points pour converger vers un optimum. Pour finir, on rappelle que seul SEGO et SEGOMOE peuvent être employés pour résoudre des problèmes d'optimisation à contraintes mixtes.

2.3.3 Optimisation Bayésienne sous contraintes de grande dimension

Dans cette section, on présente les extensions des méthodes HDBO introduites en Section 2.2.3 pour résoudre des problèmes d'optimisation sous contraintes de grande dimension. Tout d'abord, la méthodes super efficient global optimization enhanced by a KPLS surrogate model (SEGO-KPLS) [18] est introduite. Ensuite, on expliquera en quoi une extension aux problèmes contraints de REMBO [134], RREMBO [15] et HeSBO [74] est difficile à mettre en oeuvre. Finalement, on introduira la méthode scalable constrained Bayesian optimization (SCBO) [34] qui étend TuRBO [33] aux problèmes contraints.

2.3.3.1 Super efficient global optimization enhanced by a KPLS surrogate models

L'idée de BOUHLEL et al. [18] est très simple : les GP modélisant les contraintes et de la fonction objectif sont remplacés par des modèles KPLS, présentés dans le section 2.1.4.2. Ainsi, le temps nécessaire à la construction des GP est restreint en réduisant le nombre d'hyper-paramètres à estimer. On peut également coupler l'utilisation des modèles KPLS et de SEGOMOE en adoptant des modèles KPLS pour chaque GP du MOE. Ainsi, on définit une méthode de HDCBO pour des fonctions fortement multi-modales. Enfin, on utilise le critère de faisabilité de SEGO ce qui permet de résoudre le sous-problème d'optimisation dans l'espace de grande dimension. Cependant, cette optimisation du sous-problème d'enrichissement dans l'espace de grande dimension peut s'avérer difficile à réaliser au vue de la complexité du problème à résoudre. Ces méthodes ont montré de très bons résultats notamment sur le problème industriel automobile Mopta08 [18, 56] ou le cas test aérodynamique ADODG case 6 [9, 16].

2.3.3.2 Random embedding Bayesian optimization pour les problèmes avec contraintes

Les méthodes reposant sur la procédure REMBO sont difficilement adaptables aux problèmes avec contraintes. En effet, le sous-espace linéaire définit aléatoirement pourrait manquer le domaine faisable. Par exemple, la Figure 2.15 montre un problème d'optimisation sous contraintes d'inégalité pour lequel on a généré plusieurs sous-espaces linéaires. On remarque que les deux

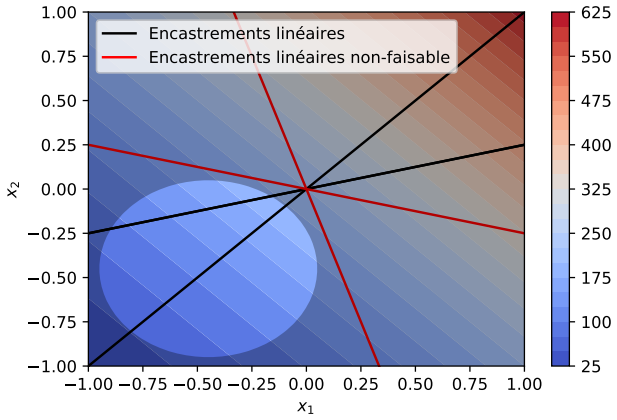


FIGURE 2.15 – Sous-espace linéaire de la fonction $s(\boldsymbol{x}) = 3(x_1 + x_2 + 3) + 24(x_1 + x_2 + 3)^2$ dont la boule de domaine faisable est donnée par $\left((x_1 + 0.45)^2 + (x_2 + 0.45)^2 - 0.25\right) \leq 0$. La zone grisée est la zone non faisable du domaine de conception.

sous-espaces linéaires rouge ne contiennent aucun point faisable. En effet, ils ne coupent pas la boule faisable (en bas à gauche du domaine). Par contre, les deux sous-espaces noirs contiennent des points faisables. On peut clairement imaginer que ce phénomène devienne prépondérant lorsque le nombre de contraintes augmente.

2.3.3.3 Scalable constrained Bayesian optimization

La méthode SCBO [34] étend la méthode TuRBO pour les problèmes comportant des contraintes. Le seul changement par rapport à la méthode initiale est la prise en compte des contraintes dans le sous-problème d'enrichissement. Ils ont choisi d'utiliser une méthode liée à un critère de faisabilité analogue à SEGO. Ceci autorise la résolution du sous-problème d'enrichissement sous contraintes dans chaque région de confiance. Cette extension a été testée avec succès sur le problème industriel automobile Mopta08 [56]. Cependant, comme pour SEGO-KPLS et SEGOMOE, l'optimisation du sous-problème d'enrichissement est opérée dans l'espace initial ce qui peut être complexe à réaliser.

Dans cette section discutant de la HDCBO, on a introduit les possibles extensions des méthodes HDBO à la CBO. D'une part, deux de ces méthodes (SEGO-KPLS et SCBO) s'étendent facilement aux problèmes contraints car elles réalisent l'optimisation du sous-problème d'enrichissement dans l'espace de conception initial. Ces algorithmes ont montré de bons résultats sur des problèmes industriels automobiles et aéronautiques. Cependant, le travail dans l'espace de conception initial peut rendre la construction des GP lente ou l'exploration trop importante. D'autre part, les méthodes reposant sur REMBO sont difficilement applicables aux problèmes avec contraintes à cause de leur caractère aléatoire. En effet, la zone faisable peut être complètement absente du sous-espace linéaire utilisé.

2.3.4 Conclusion

Pour conclure cette partie sur la CBO, on rappelle que la prise en compte des contraintes s'accomplit en modifiant le sous-problème d'optimisation (2.31) pour qu'il considère les contraintes. On a donc introduit les deux méthodes principalement utilisées dans la littérature : la méthode liée à une fonction de mérite et la méthode liée à un critère de faisabilité.

D'une part, la première méthode repose sur une fonction qui rassemble les informations des contraintes et de la fonction objectif en une unique fonction. Celle-ci est ensuite maximisée pour trouver le point suivant à calculer. Cependant, la plupart de ces fonctions de mérite ne sont pas analytiques (PESC, SUR, ALBO) ce qui ne permet pas de les utiliser facilement en grande dimension. Quant à la fonction EFI, son problème réside dans le produit des probabilités de faisabilité de chaque contrainte ce qui la rend inutilisable si le nombre de contraintes augmente fortement.

D'autre part, les méthodes à critère de faisabilité se concentrent sur la définition d'un critère qui permet de définir les zones du domaine que l'on a le droit d'échantillonner à chaque itération. Ceci sert notamment à favoriser ou non l'exploration du domaine suivant la prudence requise pour chaque évaluation du problème d'optimisation. En fait, uniquement la méthode SEGO-UTB est capable d'explorer le domaine ce qui la rend plus efficace pour des problèmes à contraintes multi-modales. Cependant, sa force d'exploration peut ralentir sa convergence. Ces méthodes à critère de faisabilité reposent toutes sur des fonctions analytiques ce qui rend possible leur utilisation en grande dimension. Néanmoins, la plupart de ces méthodes sont uniquement définies pour des problèmes à contraintes d'inégalité. Seulement ALBO, SEGO et SEGOMOE sont capables de prendre en compte des contraintes d'égalité sans la création de contraintes antagonistes. Pour finir, deux méthodes HDCBO ont été présentées et elles reposent sur la définition d'un critère de faisabilité. Toutefois, elles travaillent sur le domaine de conception initial ce qui peut rendre la construction des GP lente et l'exploration du domaine trop importante.

2.4 Synthèse du chapitre

Pour conclure ce chapitre, on rappelle que la formulation et l'estimation des hyper-paramètres des GP à partir d'un DoE a été introduit. Néanmoins, ces GP ne sont pas capables de prendre en compte des fonctions de grande dimension. En effet, la corrélation entre les points du DoE et du domaine de conception est faible et le nombre d'hyper-paramètres à estimer devient grand. C'est pourquoi, les principales extensions des GP à la grande dimension ont aussi été présentées (MGP, KPLS). Elles cherchent à définir un sous-espace linéaire de la fonction initiale qui représente le mieux la fonction. Ainsi, on peut corréler les points du domaine de conception avec les points du DoE et estimer les hyper-paramètres des GP plus rapidement. Cependant, aucune information sur la qualité de ce sous espace n'est disponible dans un temps de calcul raisonnable. De plus, on a exposé les MOE de GP pour la prise en compte de fonctions fortement multi-modales.

Ensuite, le processus de BO sans contrainte est introduit afin de résoudre des problèmes d'optimisation dont la fonction objectif est coûteuse à évaluer et dont l'unique information disponible est la valeur de la fonction au point évalué. La méthode BO repose sur la construction d'un GP modélisant la fonction objectif dont le DoE est enrichi itérativement dans le but de trouver l'optimum. Cette stratégie d'enrichissement est gérée par un sous problème d'optimisation qui réalise un compromis entre l'exploration des zones du domaine où le GP est de mauvaise qualité et l'ex-

ploitation des zones où le GP est de bonne qualité. Ce sous problème d'optimisation cherche le maximum d'une fonction d'acquisition qui renseigne sur la quantité d'information qu'un point apporte sur le minimum. Les principales fonctions d'acquisition ont été introduites (EI, WB2, WB2S, SUR, TS et PES). Seules les fonctions d'acquisition analytiques sont capables de prendre en compte des problèmes de grande dimension dans le cadre de BO classique. Toutefois, la méthode BO utilisée en grande dimension souffre de trois grands maux : la construction du GP en grande dimension peut d'avérer lente ; les fonctions d'acquisition favorisent trop l'exploration du domaine ; et les GP classiques ne permettent pas de corréler les points du domaine avec les points du DoE. Les méthodes qui permettent une meilleure prise en compte de la grande dimension reposent, pour la plupart, sur une réduction de la dimension a priori en sélectionnant aléatoirement quelques directions efficaces (REMBO, RREMBO et HeSBO). Cette sélection aléatoire ne permet pas toujours d'inclure le minimum dans le sous-espace. D'autres se fondent sur l'utilisation de GP pour la grande dimension (EGO-KPLS). Elles travaillent donc dans le domaine de conception initial ce qui favorise la part d'exploration des fonctions d'acquisition. D'autres encore se concentrent sur des zones restreintes du domaine de conception pour contrebalancer la sur-exploration du domaine (TuRBO). Malgré les bons résultats de ces méthodes, la construction des GP peut être lente.

Enfin, le processus de CBO est présenté pour résoudre des problèmes d'optimisation dont la fonction objectif et les contraintes sont coûteuses à évaluer. Les méthodes principales reposent sur une reformulation du sous-problème d'optimisation soit en un problème d'optimisation d'une fonction de mérite sans contrainte (EFI, SUR et PESC), soit en un problème d'optimisation d'une fonction d'acquisition respectant un critère de faisabilité pour chaque contrainte (SEGO, SEGOMOE, SEGO-EV, SEGO-UTB). Seules trois de ces méthodes prennent en compte les contraintes d'égalité et d'inégalité sans transformer les contraintes d'égalité (SEGO, SEGOMOE et ALBO). ALBO n'est pas utilisable en grande dimension à cause de la phase d'estimation nécessaire à chaque évaluation. SEGO et SEGOMOE ne prennent pas en compte l'incertitude des GP modélisant les contraintes, et peuvent donc rester bloqués dans une zone faisable qui ne contient pas le minimum global. Pour finir, on présente des méthodes permettant de prendre en compte les contraintes en grande dimension mais elles souffrent des mêmes pathologies que les méthodes HDBO sans contrainte.

Récapitulatif du chapitre

On a répondu aux objectifs introduit en début de chapitre grâce aux points suivants :

- Introduction de la théorie sur les GP classiques comprenant les noyaux de covariance, la fonction de vraisemblance et son optimisation.
- Présentation de deux grandes familles de GP capables de modéliser des fonctions de grandes dimensions : les MGP et les modèles KPLS.
- Introduction de la théorie sur la BO à travers le principe de fonctionnement et les fonctions d'acquisition.
- Présentation des principales méthodes HDBO, comme TuRBO, EGO-KPLS, REMBO RREMBO et HeSBO, et de leur limites.
- Introduction des méthodes CBO, comme ALBO, PESC, SUR, EFI, SEGO, SEGO-UTB et SEGO-EV, et de leur limites.
- Présentation des méthodes HDCBO, comme SEGO-KPLS et SCBO, et de leur limites.

Chapitre 3

Développement et étude d'un algorithme reposant sur la recherche de sous-espaces linéaires pour l'optimisation Bayésienne en grande dimension

« *On a toujours le choix, même si toutes les options paraissent mauvaises.* »

Robin Hobb

Sommaire

Objectifs du chapitre

- Proposer une méthode d'optimisation Bayésienne pour les problèmes d'optimisation comprenant un grand nombre de variables de conception.
- Évaluer l'impact des paramètres de la méthode proposée sur la rapidité et la robustesse de convergence.
- Comparer la méthode proposée aux algorithmes d'optimisation Bayésienne de la littérature.

Dans ce chapitre, on s'intéresse à la résolution de problèmes d'optimisation coûteux à évaluer et comprenant un grand nombre de variables de conception et pouvant comprendre quelques contraintes. Ils s'écrivent de la manière suivante

$$\min_{\boldsymbol{x}\in\Omega} \left\{ f(\boldsymbol{x}) \;\text{ s.c. }\; \boldsymbol{g}(\boldsymbol{x}) \geq 0 \text{ et } \boldsymbol{h}(\boldsymbol{x}) = 0 \right\}, \tag{3.1}$$

où $f : \mathbb{R}^d \mapsto \mathbb{R}$ est la fonction de performance ou fonction objectif, $\boldsymbol{g} : \mathbb{R}^d \mapsto \mathbb{R}^m$ sont les contraintes d'inégalité, $\boldsymbol{h} : \mathbb{R}^d \mapsto \mathbb{R}^p$ sont les contraintes d'égalité, $\Omega \subset \mathbb{R}^d$ est le domaine de conception et d le nombre de variables de conception de l'ordre de plusieurs centaines $d \geq 100$. Toutefois, on peut supposer de l'on supprime les contraintes $\boldsymbol{g}(\boldsymbol{x})$ et $\boldsymbol{h}(\boldsymbol{x})$ de ce type de problèmes grâce à des méthodes de pénalisation qui modifient la fonction objectif. Au final, on créer un nouveau problème d'optimisation sans contrainte comprenant un grand nombre de variables de conception $d \geq 100$ et qui est de la forme

$$\min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x}). \tag{3.2}$$

De plus, on n'a aucune information sur la régularité ou sur les dérivées de la fonction objectif. On cherche donc à résoudre un problème d'optimisation de type boîte noire en grande dimension et sans contrainte.

Comme introduit dans le Chapitre 2, on se place dans le contexte de la HDBO pour résoudre ce problème. Toutefois, les méthodes HDBO présentées dans le Chapitre 2 luttent pour résoudre ce type de problèmes soit parce qu'elles ne traitent que les problèmes de taille moyenne ($d \leq 100$), soit parce qu'elles ne travaillent que dans un sous-espace aléatoire de faible dimension qui peut ne pas contenir le minimum de la fonction objectif.

Pour surmonter ces obstacles, on propose une extension de la méthode RREMBO [15] présentée dans la Section 2.2.3. On argumente d'abord le choix de la méthode RREMBO. Ensuite, on présente le nouvel algorithme efficient global optimisation coupled with random and supervised embeddings (EGORSE) reposant sur un apprentissage adaptatif de sous-espaces linéaires. Puis on évalue l'impact des différents paramètres sur la rapidité et la robustesse de convergence de cette méthode sur un ensemble de problèmes de tailles croissantes. Finalement, on compare EGORSE avec les méthodes HDBO de la littérature sur des cas tests analytiques.

3.1 Optimisation Bayésienne à base d'apprentissage adaptatif de sous-espaces linéaires

Dans cette section, on cherche à proposer un algorithme HDBO capable de résoudre efficacement les problèmes d'optimisation comportant un grand nombre de variables de conception ($d \geq 100$).

Pour cela, on détermine d'abord la méthode BO de la littérature qui est la plus appropriée pour résoudre ce type de problèmes. Ensuite, on étend cette méthode pour qu'elle résolve plus efficacement ces problèmes, c.-à-d. plus rapidement et/ou de manière plus robuste.

3.1.1 Choix de la méthode d'optimisation de travail

On cherche ici la méthode HDBO qui correspond le plus à nos spécifications, c.-à-d. qui est capable de résoudre des problèmes d'optimisation sans contrainte comportant un grand nombre de variables de conception ($d \geq 100$).

D'après la Section 2.2.3, la méthode TuRBO [33] peut être utilisée à cette fin. Elle réalise l'optimisation dans des sous-domaines de Ω qui sont régis par un centre et une taille. Ceci permet de contrôler le domaine de recherche afin d'éviter une trop forte exploration du domaine de conception. TuRBO utilise toutefois un GP classique pour modéliser la fonction objectif ce qui devient très coûteux lorsque le nombre de variables de conception est important.

La méthode EGO-KPLS [18] repose sur une optimisation dans l'espace Ω. La création du GP modélisant la fonction objective est possible grâce à l'utilisation d'un modèle KPLS. Cependant, l'optimisation de la fonction d'acquisition est réalisée sur la globalité de l'espace Ω ce qui favorise l'exploration par rapport à l'exploitation des données et ce qui peut s'avérer coûteux en temps de calcul lorsque le nombre de variables de conception augmente.

D'autres algorithmes d'optimisation (tels que REMBO [134], RREMBO [15] et HeSBO [74]) reposent sur l'hypothèse que la fonction objectif dépend linéairement de quelques directions efficaces $d_e \ll d$, c.-à-d. qu'il existe une fonction $f_{\mathcal{A}} : \mathbb{R}^{d_e} \mapsto \mathbb{R}$ telle que $f_{\mathcal{A}}(\mathbf{A}\boldsymbol{x}) = f(\boldsymbol{x})$, où $\mathbf{A} \in \mathbb{R}^{d \times d_e}$ est la matrice de passage de Ω vers $\mathcal{A} = \{u = \mathbf{A}\boldsymbol{x}, \boldsymbol{x} \in \Omega\}$. En pratique, ces méthodes ne recherchent pas le meilleur sous-espace linéaire mais travaillent avec un sous-espace généré aléatoirement. Chacune de ces méthodes est accompagnée d'une application retour γ de $\mathcal{A}$ vers Ω et de la définition d'un hypercube de recherche $\mathcal{B}$ de dimension d_e. Seule RREMBO propose une application de retour bijective entre $\mathcal{A}$ et Ω ce qui la rend plus représentative de la fonction objectif. Cependant, l'hypercube $\mathcal{B}$ utilisé inclut des points qui n'ont pas d'image dans Ω par γ. Pour résoudre ce problème, BINOIS et al. [15] donnent une valeur singulière à la fonction d'acquisition dans ces zones ce qui permet de n'évaluer que des points de $\mathcal{A}$. Le calcul de la fonction d'acquisition nécessite donc la résolution d'un problème quadratique à chaque évaluation. En résumé, l'optimisation est réalisée dans un sous-espace linéaire $\mathcal{B}$ de dimension d_e, ce qui réduit le temps de calcul et la zone de recherche. Mais le calcul de la fonction d'acquisition peut s'avérer coûteux à cause de la résolution du problème quadratique associé. De plus, l'espace de recherche est choisi aléatoirement ce qui ne permet pas d'affirmer qu'il est représentatif de l'ensemble des valeurs de la fonction objectif.

En conclusion, aucune des méthodes de la littérature ne permet une résolution efficace des problèmes d'optimisation comprenant un grand nombre de variables de conception. On remarque

que seule la méthode RREMBO permet une optimisation avec un coût raisonnable bien que l'évaluation de la fonction d'acquisition nécessite la résolution d'un problème quadratique et que le sous-espace linéaire de recherche est aléatoirement choisi. On va donc modifier cette méthode afin de pouvoir choisir et apprendre le sous-espace linéaire de recherche tout en supprimant la résolution du problème quadratique à chaque évaluation de la fonction d'acquisition.

3.1.2 Extension de RREMBO à l'apprentissage adaptatif de sous-espaces linéaires

Dans cette section, on propose une extension de RREMBO afin de résoudre plus efficacement les problèmes d'optimisation sans contrainte ayant un grand nombre de variables de conception ($d \gg 100$). Pour cela, on commence par rappeler plus en détails la méthode RREMBO. Ensuite, on étend cette méthode pour qu'elle recherche itérativement le meilleur sous-espace linéaire de recherche et ne nécessite plus la résolution d'un problème quadratique dans l'évaluation de la fonction d'acquisition.

3.1.2.1 Rappels sur la méthode RREMBO

En travaillant avec la méthode RREMBO, on se place dans le cadre des méthodes faisant l'hypothèse que la fonction objectif ne dépend que de quelques directions efficaces $d_e < d$. En réalité, cela signifie qu'il existe une fonction $f_{\mathcal{A}} : \mathbb{R}^{d_e} \mapsto \mathbb{R}$ telle que $f_{\mathcal{A}}(\mathbf{A}\boldsymbol{x}) = f(\boldsymbol{x})$ où $\mathbf{A} \in \mathbb{R}^{d_e \times d}$ est la matrice de passage de Ω vers $\mathcal{A} = \{\boldsymbol{u} = \mathbf{A}\boldsymbol{x}, \boldsymbol{x} \in \Omega\}$. On rappelle également qu'on travaille avec $\Omega = [-1,1]^d$.

BINOIS et al. [15] choisissent de travailler avec une matrice $\mathbf{A}^+ \in \mathbb{R}^{d \times d_e}$ formée de vecteurs aléatoires gaussiens telle que $\boldsymbol{x} = \mathbf{A}^+ \boldsymbol{u}$. On souligne qu'avec une sélection aléatoire de $\mathbf{A}^+$, on n'est pas certain que le minimum de la fonction objectif est contenu dans le sous-espace comme le montre la Figure 2.9. En réalisant une orthogonalisation de la matrice $\mathbf{A}^+$, on peut retrouver la matrice $\mathbf{A} = \left[\mathbf{A}^+\right]^\top$. Comme $\Omega = [-1,1]^d$, il est possible de calculer facilement la plus petite boîte $\mathcal{B}_{\mathrm{B}}$ contenant $\mathcal{A}$. Elle est donnée par

$$\mathcal{B}_{\mathrm{B}} = \left[-\sum_{i=1}^{d} \left|\mathrm{A}_{1,i}\right|, \sum_{i=1}^{d} \left|\mathrm{A}_{1,i}\right|\right] \times \cdots \times \left[-\sum_{i=1}^{d} \left|\mathrm{A}_{d_e,i}\right|, \sum_{i=1}^{d} \left|\mathrm{A}_{d_e,i}\right|\right]. \tag{3.3}$$

On a donc que $\mathcal{A} \subset \mathcal{B}_{\mathrm{B}} \subset \mathbb{R}^{d_e}$. Afin de pouvoir associer une valeur à chaque point de $\mathcal{A}$, BINOIS et al. [15] introduisent une application $\gamma_{\mathrm{B}} : \mathbb{R}^{d_e} \mapsto \mathbb{R}^d$ de $\mathcal{A}$ dans Ω qui est donnée par :

$$\gamma_{\mathrm{B}}(\boldsymbol{u}) \in \arg\min_{\boldsymbol{x} \in \Omega} \left\{ \|\boldsymbol{x} - \mathbf{A}^+ \boldsymbol{u}\|^2, \text{ s.c. } \mathbf{A}\boldsymbol{x} = \boldsymbol{u} \right\}. \tag{3.4}$$

Remarque 3.1.1 *La faisabilité du problème quadratique* (3.4) *est capitale pour la suite du raisonnement. En effet, le problème quadratique* (3.4) *n'admet pas de solution faisable pour tous les points* $\boldsymbol{u} \in \mathcal{B}_{\mathrm{B}}$ *mais il admet toujours une solution faisable pour les points* $\boldsymbol{u} \in \mathcal{A}$*. Ainsi, si la solution du problème quadratique est faisable alors* $\boldsymbol{u}$ *appartient à* $\mathcal{A}$*, sinon* $\boldsymbol{u}$ *appartient à* $\mathcal{B}_{\mathrm{B}} \setminus \mathcal{A}$*.*

En choisissant $\gamma_{\mathrm{B}}^{-1}(\boldsymbol{x}) = \mathbf{A}\boldsymbol{x}$, on obtient une bijection de $\mathcal{A}$ dans Ω (voir Figure 2.9b). On peut donc réécrire le problème d'optimisation dans le sous-espace $\mathcal{A}$

$$\min_{\boldsymbol{u} \in \mathcal{A}} f\left(\gamma_{\mathrm{B}}(\boldsymbol{u})\right). \tag{3.5}$$

Toutefois, ce sous-espace $\mathcal{A}$ n'est pas trivial est peut être complexe à délimiter. De plus, la définition de bornes de l'espace de recherche est nécessaire pour l'utilisation d'un algorithme BO. C'est pourquoi on cherche à résoudre ce problème dans l'hypercube $\mathcal{B}_B$ qui est le plus petit hypercube à contenir $\mathcal{A}$.

$$\min_{\boldsymbol{u}\in\mathcal{B}_B} f\left(\gamma_B(\boldsymbol{u})\right). \tag{3.6}$$

On note que γ_B n'est pas une bijection de $\mathcal{B}_B$ dans Ω et donc que certains points de $\mathcal{B}_B$ n'ont pas d'image dans Ω (voir Figure 2.10).

L'optimisation du problème (3.6) est réalisée grâce à un algorithme BO classique qui cherche à résoudre itérativement un sous-problème d'optimisation du type

$$u^{(l+1)} \in \arg\max_{\boldsymbol{u}\in\mathcal{B}_B} \alpha_f^{(l)}(\boldsymbol{u}), \tag{3.7}$$

où $\alpha_f^{(l)} : \mathbb{R}^{d_e} \mapsto \mathbb{R}$ est une fonction d'acquisition (voir Section 2.2.1) et l l'indice de l'itération du processus BO. Cependant, on doit empêcher ce sous-problème d'optimisation de proposer des points de $\mathcal{B}_B$ qui n'appartiennent pas à $\mathcal{A}$ afin de pouvoir évaluer la fonction objectif dans un domaine où elle existe (voir Figure 2.11). Pour cela, BINOIS et al. [15] créent un sous-problème d'optimisation ayant une fonction d'acquisition qui est positive pour les points de $\mathcal{A}$ et négative pour les points de $\mathcal{B}_B$ qui ne sont pas dans $\mathcal{A}$. Ainsi, un algorithme d'optimisation réalisant la maximisation de ce sous-problème ne proposera que des points avec une fonction d'acquisition positive, c.-à-d. des points dans $\mathcal{A}$. En pratique, BINOIS et al. [15] utilisent et pénalisent l'EI (voir (2.33)) :

$$\alpha_{\mathrm{EI}_{ext}}^{(l)}(\boldsymbol{u}) = \begin{cases} \alpha_{\mathrm{EI}}^{(l)}(\boldsymbol{u}) & \text{, si } \boldsymbol{u}\in\mathcal{A} \\ -\|\boldsymbol{u}\|_2 & \text{, sinon} \end{cases} \tag{3.8}$$

Ainsi $\alpha_{\mathrm{EI}_{ext}}^{(l)}(\boldsymbol{u})$ est positive pour tous les points $\boldsymbol{u}\in\mathcal{A}$ et négative pour tous les points $\boldsymbol{u}\notin\mathcal{A}$ (voir Figure 2.11b). Toutefois, pour savoir si $\boldsymbol{u}$ appartient à $\mathcal{A}$ ou non, on doit résoudre le problème quadratique (3.4). On résout donc un problème quadratique à chaque évaluation de la fonction d'acquisition ce qui peut s'avérer coûteux en temps de calcul. On rappelle qu'une explication détaillée de la méthode RREMBO est donnée dans la Section 2.2.3.

3.1.2.2 Apprentissage adaptatif d'un sous-espace linéaire

On propose, dans cette section, une extension de RREMBO qui évite la résolution du problème d'optimisation quadratique nécessaire au calcul de la fonction d'acquisition dans RREMBO. De plus, on cherche aussi à découvrir et utiliser le sous-espace linéaire de la fonction objectif pour éviter une recherche aléatoire qui peut ne pas aboutir à une solution satisfaisante du problème.

Principe de la méthode : Pour faciliter la compréhension de la méthode, on en introduit d'abord les grands principes qui sont détaillés dans les paragraphes suivants.

A chaque itération de notre approche, on réalise simultanément $T\in\mathbb{N}^+$ optimisations sous contraintes dans T hypercubes $\mathcal{B}_B^{(t)}\subset\mathbb{R}^{d_e}$ où $t\in\{1,\dots,T\}$. Ces domaines appartiennent à T sous-espaces linéaires générés à partir d'une liste $\mathcal{R}=\{\mathcal{R}^{(1)},\dots,\mathcal{R}^{(T)}\}$ de T méthodes de réduction de dimension. Ainsi, la méthode $\mathcal{R}^{(t)}$ produit une matrice $\mathbf{A}^{(t)}\in\mathbb{R}^{d_e\times d}$ de passage de Ω à $\mathcal{A}^{(t)}=\{\boldsymbol{u}=\mathbf{A}^{(t)}\boldsymbol{x},\boldsymbol{x}\in\Omega\}$ et un hypercube de recherche $\mathcal{B}_B^{(t)}$.

Pour chacune de ces T méthodes, on modifie légèrement le problème d'optimisation considéré par RREMBO afin qu'il contienne la contrainte d'appartenance des points de $\mathcal{B}_{\mathrm{B}}^{(t)}$ à $\mathcal{A}^{(t)}$ dans la fonction objectif et/ou la contrainte considérée. On crée donc une fonction objectif $f^{(t)} : \mathbb{R}^{d_e} \mapsto \mathbb{R}$, qui existe sur l'ensemble de l'hypercube $\mathcal{B}_{\mathrm{B}}^{(t)}$, et une fonction de contrainte $g^{(t)} : \mathbb{R}^{d_e} \mapsto \mathbb{R}$, qui représente la contrainte d'appartenance des points de $\mathcal{B}_{\mathrm{B}}^{(t)}$ à $\mathcal{A}^{(t)}$. Ensuite, on résout le problème d'optimisation sous contrainte suivant :

$$\min_{\boldsymbol{u}\in\mathcal{B}_{\mathrm{B}}^{(t)}} \left\{ f^{(t)}(\boldsymbol{u}) \quad \text{s.c.} \quad g^{(t)}(\boldsymbol{u}) \geq 0 \right\}. \tag{3.9}$$

On doit donc résoudre un problème quadratique à chaque évaluation de ce problème pour savoir si le point $\boldsymbol{u}$ de $\mathcal{B}_{\mathrm{B}}^{(t)}$ appartient à $\mathcal{A}^{(t)}$. On rappelle que pour RREMBO, on cherche plutôt à résoudre :

$$\min_{\boldsymbol{u}\in\mathcal{B}_{\mathrm{B}}^{(t)}} f(\boldsymbol{u}). \tag{3.10}$$

Ce dernier problème d'optimisation ne considère pas la contrainte d'appartenance des points de $\mathcal{B}_{\mathrm{B}}^{(t)}$ à $\mathcal{A}^{(t)}$. En réalité, BINOIS et al. [15] reportent cette contrainte d'appartenance dans le problème de maximisation de la fonction d'acquisition (voir (3.8)). Ainsi, la maximisation de la fonction d'acquisition nécessite un grand nombre de résolutions de problèmes quadratiques. Ceci n'est pas le cas de la méthode qu'on propose.

De plus, RREMBO utilise une méthode à base de sous-espaces aléatoires. On propose d'utiliser différentes méthodes de réduction de dimension supervisées, comme les PLS [50] ou les MGP [40], afin d'orienter la recherche vers des sous-espaces de fortes variations de la fonction objectif.

Enfin, on met en place une recherche adaptative de ces sous-espaces de fortes variations en utilisant l'ensemble des points déjà évalués. On réalise un nombre maximum de fois (max_nb_it) l'ensemble des T optimisations simultanées avec la liste $\mathcal{R}$ des différentes méthodes de réduction de dimension en utilisant les points déjà évalués par la fonction objectif. Au final, le processus proposé est résumé par l'Algorithme 3.1.

Dans les paragraphes suivants, on détaille les trois idées principales de la méthode proposée.

Reformulation du critère d'appartenance à $\mathcal{A}$: Soit $\mathcal{R}$ un ensemble de $\mathrm{T} \in \mathbb{N}^+$ méthodes de réduction de dimension. Chaque méthode $\mathcal{R}^{(t)}$ produit une matrice $\mathbf{A}^{(t)}$, avec $t \in \{1,\dots,\mathrm{T}\}$, telle que $\mathbf{A}^{(t)} \in \mathbb{R}^{d\times d_e}$ une matrice de passage de l'espace Ω au sous-espace $\mathcal{A}^{(t)} = \left\{\boldsymbol{u} = \mathbf{A}^{(t)}\boldsymbol{x}, \boldsymbol{x} \in \Omega\right\}$. On peut donc définir l'application $\gamma_{\mathrm{B}}^{(t)}$ avec (3.4) et l'hypercube $\mathcal{B}_{\mathrm{B}}^{(t)}$ avec (3.3).

Pour éviter le problème quadratique au moment de l'évaluation de la fonction d'acquisition, on reporte le problème d'existence de l'image des points de $\mathcal{B}_{\mathrm{B}}^{(t)}$ dans Ω dans le problème d'optimisation à résoudre dans le sous-espace. L'idée est de créer un problème d'optimisation ayant une contrainte qui soit faisable lorsque le point $\boldsymbol{u} \in \mathcal{A}^{(t)}$, c.-à-d. lorsque le problème quadratique (3.4) admet une solution faisable, et qui soit non-faisable sinon.

Par exemple, on peut choisir l'opposé de la norme du point $\boldsymbol{u}$ s'il n'appartient pas à $\mathcal{A}^{(t)}$, ce qui fait tendre la contrainte vers zéro lorsque l'on se rapproche du centre du domaine $\mathcal{B}_{\mathrm{B}}^{(t)}$. La valeur de la contrainte est négative et se rapproche de zéro lorsque l'on se rapproche de $\mathcal{A}^{(t)}$.

Pour définir la zone faisable, on s'appuie sur (3.3) et sur $\Omega = [-1,1]^d$. Cette équation montre que les points du bord de $\mathcal{B}_{\mathrm{B}}^{(t)}$ qui ont une image dans Ω par $\gamma_{\mathrm{B}}^{(t)}$ sont des angles du domaine Ω. On entend par angles du domaine, les points dont toutes les composantes sont 1 ou -1. Or les

Algorithme 3.1 Principe du processus proposé.

entrée : une fonctions objectif f, un DoE initial $\mathcal{D}_f^{(0)}$, un nombre maximum d'itérations max_nb_it, un nombre maximum d'itérations par sous-espace max_nb_it_sub, un nombre de composantes d_e, une liste $\mathcal{R} = \{\mathcal{R}^{(1)}, \ldots, \mathcal{R}^{(\mathrm{T})}\}$ de $\mathrm{T} \in \mathbb{N}^+$ méthodes de réduction de dimension

1: **pour** $i = 0$ **à** max_nb_it - 1 **faire**
2: **pour** $t = 1$ **à** T **faire**
3: Construire $\mathbf{A}^{(t)} \in \mathbb{R}^{d_e \times d}$ avec $\mathcal{R}^{(t)}$ nécessitant ou non $\mathcal{D}_f^{(i)}$
4: Établir $\mathcal{B}_{\mathrm{B}}^{(t)}$
5: Construire $f^{(t)}$ qui existe sur l'ensemble du domaine $\mathcal{B}_{\mathrm{B}}^{(t)}$
6: Construire une contrainte $g^{(t)}$ faisable sur $\mathcal{A}^{(t)}$ et non-faisable sur $\mathcal{B}_{\mathrm{B}}^{(t)} \setminus \mathcal{A}^{(t)}$.
7: Résoudre le problème d'optimisation

$$\min_{\boldsymbol{u} \in \mathcal{B}_{\mathrm{B}}^{(t)}} \{ f^{(t)}(\boldsymbol{u}) \quad \text{s.c.} \quad g^{(t)}(\boldsymbol{u}) \geq 0 \}$$

avec une méthode CBO utilisant au maximum max_nb_it_sub itérations.
8: **fin pour**
9: $\mathcal{D}_f^{(i+1)} = \mathcal{D}_f^{(i)} \cup \{\text{Les points déjà évalués par } f \text{ à l'itération } i\}$
10: **fin pour**

sortie : Le meilleur point en terme de valeur de f dans $\mathcal{D}_f^{(\text{max\_nb\_it})}$

angles du domaine Ω sont les points les plus éloignés de son centre $x_c = \mathbf{0} \in \mathbb{R}^d$ et sont de même norme. Toutes les images des points de $\mathcal{B}_{\mathrm{B}}^{(t)}$ par $\gamma_{\mathrm{B}}^{(t)}$ ont donc une norme plus faible, ou égale, à la norme des angles du domaine Ω. Il est simple de choisir comme valeur de contrainte pour la zone faisable, la différence entre la norme des angles du domaine Ω avec la norme de $\gamma_{\mathrm{B}}^{(t)}(\boldsymbol{u})$ si $\boldsymbol{u} \in \mathcal{A}^{(t)}$. Ainsi, tous les points de $\boldsymbol{u}$ appartenant à $\mathcal{A}^{(t)}$ ont une valeur de contrainte positive. En outre, la valeur de la contrainte se rapproche de zéro lorsque l'on se rapproche des zones de $\mathcal{B}_{\mathrm{B}}^{(t)}$ qui n'appartiennent pas à $\mathcal{A}^{(t)}$. A titre d'exemple, la Figure 3.1 montre la fonction à deux variables de conception déjà utilisée dans la Section 2.2.3. Elle représente l'image de $\mathcal{B}_{\mathrm{B}}^{(t)}$ par $\gamma_{\mathrm{B}}^{(t)}$ en rouge et indique également l'emplacement de la projection des bords de $\mathcal{B}_{\mathrm{B}}^{(t)}$ qui se trouvent aux angles du domaine Ω. On souligne qu'ici $\mathcal{B}_{\mathrm{B}}^{(t)} = \mathcal{A}^{(t)}$. On constate bien ici que tous les points de l'image de $\mathcal{B}_{\mathrm{B}}^{(t)}$ par $\gamma_{\mathrm{B}}^{(t)}$ ont une norme plus faible que la norme des angles du domaine Ω.

Enfin, on normalise la contrainte pour qu'elle soit comprise entre -1 et 1. Cette symétrie sur les bornes de la contrainte permet de ne pas rendre prépondérante une zone du domaine de conception par rapport à une autre. Ceci serait préjudiciable au processus d'optimisation. Cette contrainte est finalement donnée par $g^{(t)}(\boldsymbol{u}) \geq 0$ où $g^{(t)} : \mathcal{B}_{\mathrm{B}}^{(t)} \mapsto \mathbb{R}$ s'écrit :

$$g^{(t)}(\boldsymbol{u}) = \begin{cases} 1 - \dfrac{\left\| \gamma_{\mathrm{B}}^{(t)}(\boldsymbol{u}) \right\|_2^2}{d} & , \text{si } \boldsymbol{u} \in \mathcal{A}^{(t)} \\ -\left\| \boldsymbol{u}^{\mathbf{A}^{(t)}} \right\|_2^2 & , \text{sinon} \end{cases} \tag{3.11}$$

avec $u_i^{\mathbf{A}^{(t)}} = u_i / \sum_{j=1}^{d} \left| \mathrm{A}_{i,j}^{(t)} \right|$ où $u_i^{\mathbf{A}^{(t)}}$ et u_i sont respectivement les i^{e} composantes des vecteurs $\boldsymbol{u}^{\mathbf{A}^{(t)}}$ et $\boldsymbol{u}$. Elle est positive pour les points de $\mathcal{A}^{(t)}$ et négative pour les points de $\mathcal{B}_{\mathrm{B}}^{(t)} \setminus \mathcal{A}^{(t)}$.

Comme le montre la Figure 2.11a, la fonction $f^{(t)}(\boldsymbol{u}) = f\left(\gamma_{\mathrm{B}}^{(t)}(\boldsymbol{u})\right)$ n'est pas définie sur l'ensemble du domaine $\mathcal{B}_{\mathrm{B}}^{(t)}$. Les valeurs de cette fonction n'existent pas dans les zones grises du domaine $\mathcal{B}_{\mathrm{B}}^{(t)}$. C'est pourquoi on cherche à lui donner une valeur dans cette zone afin de la définir

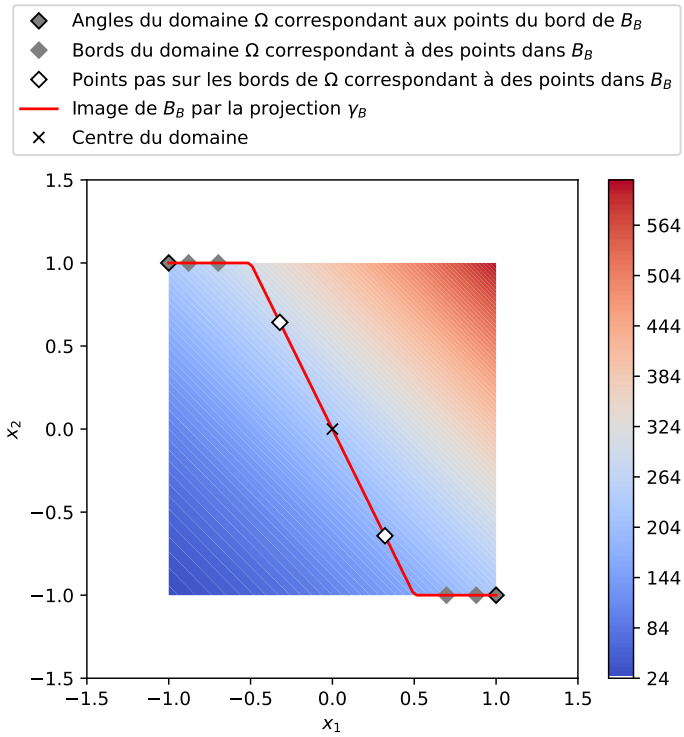


FIGURE 3.1 – Image du segment $\mathcal{B}_{\mathrm{B}}^{(t)}$ par $\gamma_{\mathrm{B}}^{(t)}$.

sur l'ensemble de $\mathcal{B}_{\mathrm{B}}^{(t)}$. On choisit donc $f^{(t)}(\boldsymbol{u}) = f\left(\gamma_{\mathrm{W}}^{(t)}(\boldsymbol{u})\right)$ si $\boldsymbol{u} \notin \mathcal{A}^{(t)}$ où

$$\gamma_{\mathrm{W}}^{(t)}(\boldsymbol{u}) \in \arg\min_{\boldsymbol{x}\in\Omega} \|\boldsymbol{x} - \left[\mathbf{A}^{(t)}\right]^{+} \boldsymbol{u}\|^2, \tag{3.12}$$

où $\left[\mathbf{A}^{(t)}\right]^{+} = \left[\mathbf{A}^{(t)}\right]^{\top}\left(\mathbf{A}^{(t)}\left[\mathbf{A}^{(t)}\right]^{\top}\right)^{-1}$ est le pseudo-inverse de $\mathbf{A}^{(t)}$. En effet, l'application $\gamma_{\mathrm{W}}^{(t)}(\boldsymbol{u})$ existe pour l'ensemble des points de $\mathcal{B}_{\mathrm{B}}^{(t)}$ bien qu'elle puisse donner le même $\boldsymbol{x} \in \Omega$ pour plusieurs $\boldsymbol{u} \in \mathcal{B}_{\mathrm{B}}^{(t)}$. De plus, il est possible d'atteindre ces points $\boldsymbol{x}$ avec des points $\boldsymbol{u}$ appartenant à $\mathcal{A}^{(t)}$ (voir Figure 2.9a). Ainsi, le minimum de la fonction objectif se trouve obligatoirement dans $\mathcal{A}^{(t)}$. En outre, tous ces points sont non-faisables ce qui réduit leur chance d'être évalués. Finalement, la fonction objective dans l'hypercube $\mathcal{B}_{\mathrm{B}}^{(t)}$ est donnée par :

$$f^{(t)}(\boldsymbol{u}) = \begin{cases} f\left(\gamma_{\mathrm{B}}^{(t)}(\boldsymbol{u})\right) & \text{, si } \boldsymbol{u} \in \mathcal{A}^{(t)} \\ f\left(\gamma_{\mathrm{W}}^{(t)}(\boldsymbol{u})\right) & \text{, sinon} \end{cases} \tag{3.13}$$

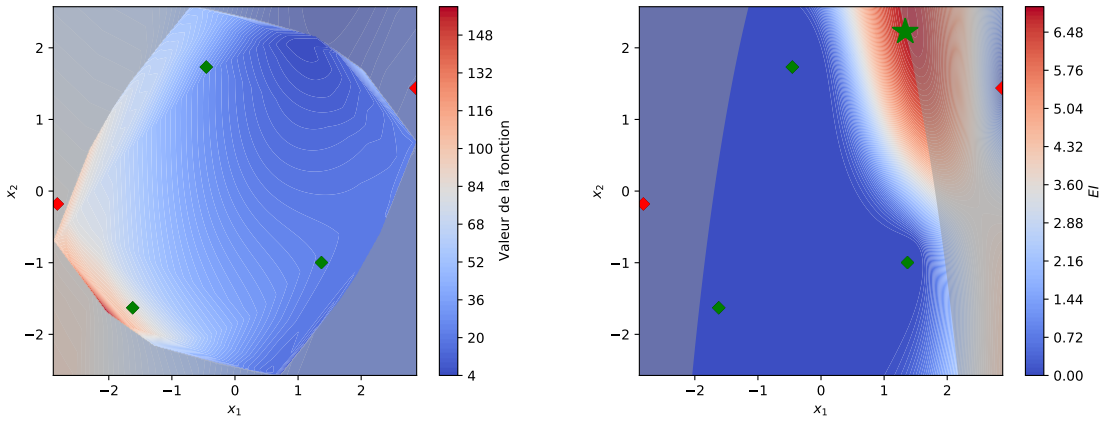
En somme, on cherche à résoudre un problème d'optimisation sous contrainte d'inégalité dans l'hypercube $\mathcal{B}_{\mathrm{B}}^{(t)}$ qui s'écrit :

$$\min_{\boldsymbol{u}\in\mathcal{B}_{\mathrm{B}}^{(t)}} \left\{ f^{(t)}(\boldsymbol{u}) \quad \text{s.c.} \quad g^{(t)}(\boldsymbol{u}) \geq 0 \right\}. \tag{3.14}$$

Pour résoudre ce problème, on peut finalement utiliser une méthode CBO classique (voir Section 2.3), comme SEGO, sans avoir à modifier la fonction d'acquisition, le critère de faisabilité ou la fonction de mérite contrairement à RREMBO.

La Figure 3.2a montre le problème d'optimisation (3.14) pour la fonction de 10 variables de conception utilisée dans la Figure 2.11a et projetée dans un sous-espace linéaire de 2 variables

de conception. On remarque que les zones grisées non-faisables correspondent aux zones grises



(a) Problème d'optimisation (3.14).

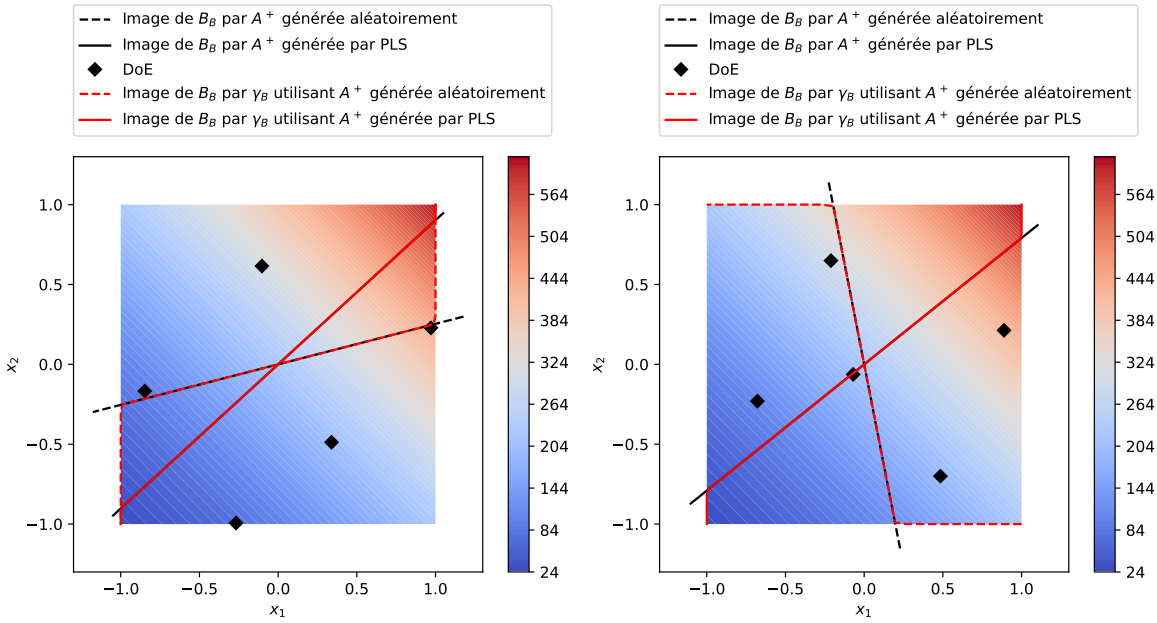
(b) Sous-problème d'optimisation de SEGO associé au problème (3.14).

FIGURE 3.2 – Représentation du problème d'optimisation (3.14) dans un sous-espace de dimension $d_e = 2$ et la représentation du sous-problème d'optimisation de SEGO associé. La zone grise représente les zones non-faisables, les carrés verts sont les points du DoE dans $\mathcal{A}^{(t)}$, les carrés rouges sont les points du DoE n'appartenant pas à $\mathcal{A}^{(t)}$, l'étoile verte est la solution du sous-problème d'optimisation.

de la Figure 2.11a. La Figure 3.2b montre le sous-problème d'optimisation de la méthode SEGO utilisant l'EI où les carrés verts et rouges sont les points du DoE initial. On remarque que les deux zones non-faisables prédites, représentées par les zones grisées (voir Figure 3.2b), contiennent les deux points non-faisables connus du problème (3.14). On souligne, qu'avec SEGO, seule la moyenne du GP modélisant la contrainte est utilisée pour définir les zones admissibles. Contrairement à RREMBO, aucune résolution d'un problème quadratique n'est nécessaire à la résolution de ce sous-problème. Au final, cette modification du problème d'optimisation permet d'éviter la résolution du problème quadratique de l'évaluation de la fonction d'acquisition et le reporte sur l'évaluation de la fonction objectif. Ainsi, on ne résout qu'un seul problème quadratique à chaque itération de SEGO.

Apprentissage du sous-espace linéaire : Dans la Section 2.2.3, on a montré que la sélection aléatoire d'un sous-espace linéaire peut donner un sous-espace qui ne contient pas le minimum de la fonction objectif (voir Figure 2.9). En effet, la zone rouge que l'on peut explorer est quasiment perpendiculaire à la direction de variation de la fonction objectif considérée.

Pour espérer trouver un sous-espace linéaire dans une direction proche de la meilleure direction de variation, on peut s'intéresser aux méthodes de réduction de dimension supervisées, c.-à-d. qui prennent en entrée un ensemble de points du domaine considéré et les valeurs de la fonction associée. En retour, ces méthodes donnent une matrice de passage $\mathbf{A}^{(t)} \in \mathbb{R}^{d \times d_e}$ de l'espace Ω à $\mathcal{A}^{(t)}$. Par exemple, on peut utiliser la méthode des PLS (voir Section 2.1.4.2) ou la méthode des MGP (voir Section 2.1.4.1) en considérant les points du DoE initial. La Figure 3.3 montre la prise en compte de deux DoE initiaux différents pour le calcul d'un sous-espace linéaire par PLS. On a également ajouté un sous-espace généré par une matrice aléatoire gaussienne. On constate que la méthode des PLS donne, pour les deux DoE, un sous-espace qui contient le minimum de la fonction. Bien que la méthode aléatoire produit un sous-espace contenant le minimum sur la



(a) DoE 1.

(b) DoE 2.

FIGURE 3.3 – Images de deux sous-espaces $\mathcal{B}_B$ par γ_B utilisant A^+ générée par PLS ou aléatoirement.

Figure 3.3a, ce n'est plus le cas sur la Figure 3.3b. L'utilisation d'une méthode de réduction de dimension supervisée est donc justifiée. Pour trouver le minimum de la fonction, on réalise une CBO dans le sous-espace $\mathcal{A}^{(t)}$ généré par la matrice $\mathbf{A}^{(t)}$ avec un maximum de max_nb_it_sub itérations dans le sous-espace $\mathcal{A}^{(t)}$. Cependant, on imagine que dans un espace de dimension plus important, l'approximation du sous-espace linéaire peut s'averer plus complexe.

Pour améliorer le sous-espace dans lequel on recherche le minimum de la fonction objectif, on peut itérer le processus précédent en utilisant l'ensemble des points évalués sur la fonction. On note qu'on peut également ajouter des points qui ne font pas partie du sous-espace linéaire $\mathcal{A}^{(t)}$ découvert afin d'apporter de l'information non-biaisée par la sélection du sous-espace. Ce type de méthodes est notamment mises en place par GAUDRIE et al. [42]. Dans ce sens, on peut ajouter les points issus d'une optimisation de type RREMBO, on fait appel à une CBO dans un sous-espace linéaire dont la matrice de passage est issue d'une distribution gaussienne aléatoire. Ainsi, on conserve les propriétés de convergence des méthodes REMBO [134] et RREMBO [15] précédemment introduites. Pour finir, on peut imaginer une généralisation du processus à plusieurs méthodes de réduction de dimension non-supervisées (comme les matrices aléatoires gaussiennes ou les tables de hachage) ou supervisées (comme les PLS ou les MGP) pour profiter de l'ensemble de leurs avantages et ainsi améliorer l'apport des méthodes de réduction de dimension supervisées.

Bilan : Pour résumer, on a construit un algorithme d'optimisation Bayésienne en grande dimension utilisant une combinaison de sous-espaces linéaires supervisés et non-supervisés, qu'on appelle EGORSE. On a défini un groupe $\mathcal{R}$ de $T \in \mathbb{N}^+$ méthodes de réduction de dimension supervisées et/ou non-supervisées pour lesquelles on réalise une CBO à chaque itération $i \in \{1, \ldots, \text{max\_nb\_it}\}$ de l'algorithme. Ces méthodes de réduction de dimension prennent en compte l'ensemble des points évalués lors des précédentes itérations ce qui permet d'adapter le sous-

espace de recherche au cours de l'optimisation. De plus, on a limité le nombre de résolutions de problèmes quadratiques par rapport à la méthode RREMBO en modifiant le problème d'optimisation résolu dans chaque sous-espace. En effet, on a ajouté la contrainte d'appartenance des points de $\mathcal{B}_{\mathrm{B}}^{(t)}$ à $\mathcal{A}^{(t)}$ au problème à résoudre dans chaque sous-espace (voir (3.14)). En utilisant une méthode CBO pour résoudre (3.14), on modélise la fonction objectif et la contrainte d'appartenance par deux GP. Ainsi, avec la méthode EGORSE, l'optimisation du sous-problème d'optimisation dans le processus BO ne nécessite plus de résolution de problèmes quadratiques contrairement à RREMBO. Au final, la méthode EGORSE comporte trois apports par rapport à RREMBO :

1. la limitation du nombre de résolutions de problèmes quadratiques grâce à une reformulation du problème d'optimisation dans les sous-espaces considérés.
2. l'utilisation de méthodes de réduction de dimension supervisées qui favorisent la recherche dans des directions de l'espace Ω pour lesquelles la fonction objectif varie fortement.
3. l'apprentissage adaptatif des sous-espaces linéaires en utilisant l'ensemble des points du DoE.

Ce processus d'optimisation est donné dans l'Algorithme 3.2 et est représenté dans le schéma de la Figure 3.4.

Algorithme 3.2 Le processus EGORSE.

entrée : une fonction objectif f, un DoE initial $\mathcal{D}_f^{(0)}$, un nombre maximum d'itérations max_nb_it, un nombre maximum d'itérations par sous-espace max_nb_it_sub, un nombre de composantes d_e, une liste $\mathcal{R} = \{\mathcal{R}^{(1)}, \ldots, \mathcal{R}^{(\mathrm{T})}\}$ de $\mathrm{T} \in \mathbb{N}^+$ méthodes de réduction de dimension supervisées et non-supervisées

1: **pour** $i = 0$ **à** max_nb_it - 1 **faire**
2: **pour** $t = 1$ **à** T **faire**
3: Construire $\mathbf{A}^{(t)} \in \mathbb{R}^{d_e \times d}$ avec $\mathcal{R}^{(t)}$ nécessitant ou non $\mathcal{D}_f^{(i)}$
4: Établir $\mathcal{B}_{\mathrm{B}}^{(t)}$ (voir (3.3))
5: Construire $f^{(t)}$ (voir (3.13))
6: Construire $g^{(t)}$ (voir (3.11))
7: Résoudre le problème d'optimisation

$$\min_{\boldsymbol{u} \in \mathcal{B}_{\mathrm{B}}^{(t)}} \left\{ f^{(t)}(\boldsymbol{u}) \quad \text{s.c.} \quad g^{(t)}(\boldsymbol{u}) \geq 0 \right\}$$

avec une méthode CBO utilisant au maximum max_nb_it_sub itérations.
8: **fin pour**
9: $\mathcal{D}_f^{(i+1)} = \mathcal{D}_f^{(i)} \cup \{\text{Les points déjà évalués par } f \text{ à l'itération } i\}$
10: **fin pour**

sortie : Le meilleur point en terme de valeur de f dans $\mathcal{D}_f^{(\text{max\_nb\_it})}$

3.1.3 Conclusion

Dans cette section, on a mis en place un algorithme d'optimisation Bayésienne pour les problèmes de grande dimension $d \gg 100$.

Pour cela, on a d'abord évalué les performances des méthodes HDBO de la littérature. On a constaté que les méthodes TuRBO et EGO-KPLS sont difficilement capables de prendre en compte des problèmes avec un grand nombre de variables de conception puisqu'elles travaillent dans l'espace initial de grande dimension. Elles doivent donc construire un GP en grande dimension, ce

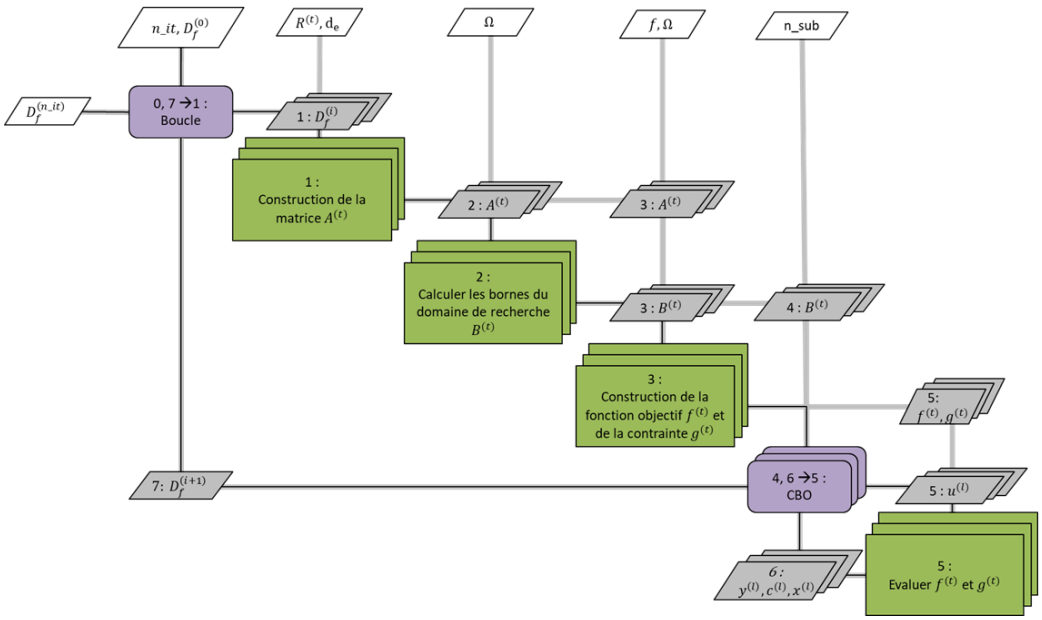


FIGURE 3.4 – Représentation par diagramme XDSM (eXtended Design Structure Matrix) [64] de la méthode EGORSE. Les lignes noires représentent le chemin, correspondant aux nombres, suivis par le processus, les boîtes blanches sont les entrées/sorties du processus, les boîtes grises sont les données utilisées pendant le processus, les boîtes vertes sont les actions exécutées, les boîtes violettes représentent les boucles et optimisations.

qui peut être coûteux, et réaliser l'optimisation de la fonction d'acquisition en grande dimension, ce qui peut conduire à une forte exploration du domaine de conception. Les méthodes REMBO, HeSBO et RREMBO sont plus appropriées car elles travaillent dans un sous-espace de faible dimension. Ainsi, la construction du GP et la résolution du sous-problème d'optimisation sont réalisées dans l'espace de faible dimension ce qui en réduit fortement le coût de calcul. Cependant, ces méthodes reposent sur la génération de sous-espaces linéaires aléatoires ce qui ne garantit pas que le minimum de la fonction objectif soit contenu dans ces sous-espaces. De plus pour RREMBO, le calcul de la fonction d'acquisition nécessite la résolution d'un problème d'optimisation quadratique rendant son optimisation coûteuse.

On a choisi de lever les verrous de la méthode RREMBO en :

— reformulant le problème d'optimisation dans le sous-espace en un problème d'optimisation sous-contrainte. Ceci supprime la résolution du problème quadratique à chaque évaluation de la fonction d'acquisition.

— utilisant des méthodes de réduction de dimension supervisées ce qui permet de maximiser les chances que le minimum soit dans le sous-espace considéré.

— introduisant un processus itératif d'apprentissage du sous-espace grâce à une optimisation dans un sous-espace généré aléatoirement.

Ce nouveau processus HDBO est nommé EGORSE.

Toutefois, on constate que l'algorithme EGORSE proposé comporte deux paramètres majeurs que l'on doit régler, liés à la taille du DoE initial et aux méthodes de réduction de dimension utilisées. Afin de proposer le meilleur processus d'optimisation possible, on étudie l'impact de ces paramètres dans la section suivante.

3.2 Impact des paramètres d'EGORSE

Dans cette section, on étudie l'impact des paramètres d'EGORSE sur la rapidité et la robustesse de convergence. Pour cela, on introduit d'abord l'implémentation choisie pour EGORSE. Ensuite, on liste les différents paramètres pris en compte dans l'étude. Dans un premier temps, on évalue l'impact de ces paramètres sur un ensemble de quatre problèmes de taille moyenne ($d \leq 100$) à l'aide de courbes de convergence. Dans un second temps, on étudie l'influence de ces mêmes paramètres sur un problème comportant un très grand nombre de variables de conception ($d = 1\,000$).

3.2.1 Implémentation de EGORSE

Pour l'implémentation d'EGORSE, on fait le choix de travailler avec le langage Python 3,7. On fait également les choix suivants. On rappelle que dans le processus EGORSE, une CBO est réalisée à chaque itération. Cette CBO est effectuée avec l'algorithme SEGO [108] de la toolbox SEGOMOE [9] développée par l'ONERA et l'ISAE-SUPAERO. Plus spécifiquement, cette toolbox utilise SMT [19] pour construire les GP nécessaires à l'optimisation. SMT repose sur des GP dont la moyenne est un polynôme de degré choisi par l'utilisateur et dont les coefficients de régression sont des hyper-paramètres des GP. Pour ces expérimentations numériques, on utilise une régression constante et un noyau de covariance Gaussien. Tous les hyper-paramètres des GP sont gérés grâce aux paramètres par défaut de SMT.

On choisi l'EI comme fonction d'acquisition à optimiser. Pour son optimisation, on procède en deux étapes. D'une part, on trouve un point de départ en résolvant le problème (2.72) avec improved stochastic ranking evolution strategy (ISRES) [107] de la toolbox Python Nlopt [55]. ISRES est un algorithme d'optimisation évolutionnaire capable de résoudre des problèmes d'optimisation multi-modaux comprenant des contraintes d'inégalité et d'égalité. On peut ainsi explorer le domaine à la recherche de la zone qui maximise la fonction d'acquisition tout en respectant le critère de faisabilité. Cependant, ce type d'algorithme nécessite un grand nombre d'appels à la fonction objectif pour converger. C'est pourquoi on limite le nombre d'appels et on raffine la solution donnée par ISRES avec un algorithme d'optimisation utilisant les dérivées. Ceci est possible car on dispose des dérivées analytiques de la fonction d'acquisition et du critère de faisabilité entrant en jeu dans le problème d'optimisation. On termine donc la résolution du problème (2.72) avec l'algorithme sparse nonlinear optimizer (SNOPT) [45] de la toolbox Python PyOptSparse [82] dont le point initial est la solution fournie par ISRES.

Pour la résolution du problème quadratique (3.4) nécessaire à l'évaluation de la fonction objective, on utilise la toolbox Python CVXOPT [1] avec les paramètres par défaut. Pour l'évaluation de la fonction objectif, on considère qu'un point $\boldsymbol{u} \in \mathcal{B}_{\mathrm{B}}^{(t)}$ appartient à $\mathcal{A}^{(t)}$ si le statut du résultat de l'optimisation par CVXOPT est '`optimal`' signifiant que le problème a bien été résolu.

3.2.2 Choix des paramètres de EGORSE

Les paramètres pris en compte dans cette étude sont les suivants.

— La quantité de points dans le DoE initial peut fortement influencer les méthodes de réduction de dimension supervisées. On teste donc l'ensemble des versions d'EGORSE pour trois

tailles de DoE initial : 5 points, d points et $2d$ points, si d est le nombre de variables de conception du problème.

— La méthode de réduction de dimension supervisée peut aussi modifier le comportement de l'algorithme en ne favorisant pas les mêmes directions de recherche. On considère donc les méthodes des PLS [50] et MGP [40]. La méthode des PLS est issue de la toolbox Python scikit-learn [80]. Pour la méthode MGP, on se repose sur notre implémentation faite dans SMT [19].

De plus, la méthode de réduction de dimension non-supervisée choisie est la génération de matrices de passage aléatoires gaussiennes (voir Section 2.2.3). On appelle donc les deux versions d'EGORSE utilisant des méthodes de réduction de dimension supervisées : EGORSE (PLS + Gaussian) pour la méthode utilisant les PLS et EGORSE (MGP + Gaussian) pour la méthode utilisant les MGP.

Pour être plus complet, on ajoute deux autres versions pour EGORSE n'utilisant que des méthodes de réduction de dimension non-supervisées. La première utilise des matrices de passage aléatoires gaussiennes comme dans la méthode RREMBO [15], on la nomme EGORSE (Gaussian). La seconde construit des matrices de passage aléatoires à partir de tables de hachage comme la méthode HeSBO [74], on la nomme EGORSE (Hachage).

3.2.3 Étude sur quatre problèmes de taille moyenne

Dans cette section, on évalue l'impact des différents paramètres d'EGORSE que l'on fait varier sur la rapidité et la robustesse de convergence. Pour cela, on introduit d'abord l'ensemble des quatre problèmes de taille moyenne sur lesquels on réalise cette étude. Ensuite, on introduit la méthode de comparaison que l'on utilise. Pour finir, les résultats des tests sont analysés.

3.2.3.1 Les quatre problèmes de taille moyenne

La première classe de problèmes est une modification de MB [79] pour lequel on augmente artificiellement le nombre de variables de conception. Ce type de problèmes est largement utilisé dans la littérature [15, 74, 134]. Ce problème se définit comme suit :

$$\min_{\boldsymbol{u}\in\Omega_1} f_1(\boldsymbol{u}), \tag{3.15}$$

où $\Omega_1 = [-5,10]\times[0,15]$ et

$$f_1(\boldsymbol{u}) = \left[\left(u_2 - \frac{5.1u_1^2}{4\pi^2} + \frac{5u_1}{\pi} - 6\right)^2 + \left(10 - \frac{10}{8\pi}\right)\cos(u_1) + 1\right] + \frac{5u_1+25}{15}. \tag{3.16}$$

On choisit la version modifiée du problème de Branin [79] car elle comporte 3 minima locaux dont un global. La valeur de celui-ci est d'environ MB_d_{min} = 1.1. De plus, on normalise le problème pour que $\boldsymbol{u}\in[-1,1]^2$. Pour augmenter artificiellement le nombre de variables de conception, on génère aléatoirement une matrice $\mathbf{A}_d \in \mathbb{R}^{d\times 2}$ telle que pour tout $\boldsymbol{x}\in[-1,1]^d$, $\mathbf{A}_d\boldsymbol{x} = \boldsymbol{u}$ appartient au domaine $[-1,1]^2$. On peut donc définir une fonction objective MB_d, où d est le nombre de variables de conception, telle que MB_$d(\boldsymbol{x}) = f_1(\mathbf{A}_d\boldsymbol{x})$. Finalement, on s'intéresse à la résolution

du problème d'optimisation suivant :

$$\min_{x\in[-1,1]^d} \text{MB\_}d(\boldsymbol{x}). \tag{3.17}$$

On va donc travailler avec trois fonctions de ce type (MB_10, MB_20 et MB_100) afin d'étudier les dimensions 10, 20 et 100.

Le dernier problème, nommé Rover_60 [136], consiste à router un robot dans une forêt d'un point de départ x_{start} à un point d'arriver x_{goal} en évitant les arbres sur sa route. La trajectoire du robot est une spline définie par 30 points de contrôle. Ces trente points, comprenant les points de départ et d'arrivée, sont les variables de conception de ce problème d'optimisation. Au final, on a 60 variables de conception, c.-à-d. deux variables par point de contrôle. La fonction objectif est minimale lorsque le robot parcourt la trajectoire la plus courte sans rencontrer d'arbres dans la forêt. Le minimum vaut $f_{min} = -5$. La Figure 3.5 donne un exemple de trajectoire relative à ce problème.

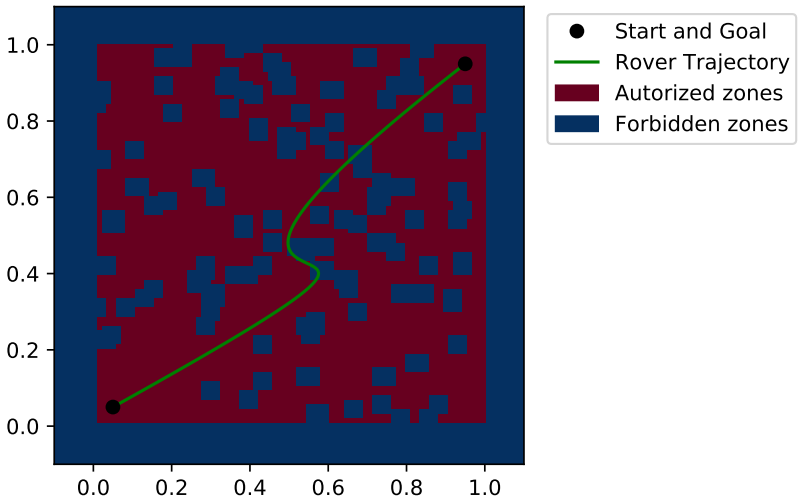


FIGURE 3.5 – Exemple de trajectoire d'un robot dans une forêt.

3.2.3.2 Construction des courbes de convergence

Pour étudier l'impact des paramètres d'EGORSE, on utilise des courbes de convergence sur chacun des quatre problèmes. Pour cela, on réalise 100 optimisations indépendantes pour chacune des variantes d'EGORSE en utilisant 100 DoE initiaux préalablement générés. De ce fait, l'ensemble des variantes est initialisé avec les mêmes DoE initiaux, et ainsi permet de réduire le biais causé par l'utilisation d'un DoE initial. On impose un nombre maximum de 10 itérations, un nombre maximum d'évaluations par sous-espace de $20 * d_e$ et une dimension efficace $d_e = 2$. Pour les versions d'EGORSE utilisant uniquement des méthodes de réduction non-supervisées (de type RREMBO et HeSBO), on double le nombre d'itérations afin de garder un nombre fixe d'évaluations de la fonction objectif. Ceci correspond à un budget total de $n_{evals} = 800 + n_{start}$ évaluations de la fonction objectif par optimisation avec n_{start} le nombre de points du DoE initial.

Toutes ces variantes sont ensuite comparées en affichant l'évolution de la moyenne et de l'écart type sur les 100 optimisations de la valeur évaluée la plus faible de la fonction objectif en

fonction du nombre d'évaluations. Pour des raisons de lisibilité, on affiche l'écart type réduit d'un facteur 4.

3.2.3.3 Analyse des résultats

Dans cette section, on analyse les résultats obtenus au cours des tests présentés dans la Section 3.2.3.2. Pour chacun des quatre problèmes de taille moyenne (MB_10, MB_20, Rover_60 et MB_100), on trace quatre figures. Les trois premières représentent l'ensemble des configurations d'EGORSE testées pour une taille de DoE initial fixe (5, d ou $2d$ points). Ensuite, on sélectionne la meilleure version pour chacune des tailles de DoE initial, c.-à-d. la variante qui converge le plus rapidement vers une valeur faible de la fonction objectif avec un faible écart type. Dans la dernière figure, on trace ces trois versions pour déterminer la taille de DoE initial la plus adaptée. De plus, on reporte le temps moyen d'optimisation pour chacune des variantes considérées. Les problèmes sont étudiés par ordre croissant de dimension afin de mieux appréhender l'impact du nombre de variables de conception sur les performances. On termine cette étude en essayant de dégager la meilleure version d'EGORSE à utiliser.

La Figure 3.6 montre les courbes de convergence de quatre variantes d'EGORSE pour trois tailles de DoE initial pour le problème MB_10. L'ensemble de ces figures montre qu'aucune des variantes d'EGORSE n'est sensiblement plus performante que les autres. Elles convergent toutes vers une valeur proche de 5 bien qu'EGORSE (PLS + Gaussian) semble converger plus rapidement vers une valeur plus faible. De plus, la taille du DoE initial ne semble pas avoir d'influence sur la rapidité et la robustesse de convergence. Enfin, on remarque qu'aucune de ces méthodes n'est capable de converger vers le minimum global de la fonction ($f_{min} \approx 1.1$) avec le budget alloué. Grâce à la Figure 3.6d, on valide que la taille du DoE initial n'impacte pas la rapidité et la robustesse de convergence d'EGORSE (PLS + Gaussian) pour MB_10. En résumé, la modification des paramètres d'EGORSE n'a pas d'impact sur la résolution du problème MB_10.

Grâce à la Figure 3.7, on constate qu'EGORSE (PLS + Gaussian) et EGORSE (MGP + Gaussian) trouvent les valeurs les plus faibles de MB_20 quelque soit la taille du DoE initial. Pour une taille de DoE initial de 5 points, l'utilisation des MGP semble la plus appropriée. Au contraire, pour les tailles de DoE initial de d et $2d$, l'emploi des PLS donne de meilleures performances. Dans tous les cas, les variantes d'EGORSE utilisant une méthode de réduction de dimension supervisée convergent vers les valeurs de fonction objectif les plus faibles. On constate également, avec la Figure 3.7d, que la taille du DoE n'a pas d'influence sur les propriétés de convergence d'EGORSE. En effet, toutes les versions d'EGORSE convergent en moyenne vers la même valeur de fonction objectif et avec la même vitesse. Sur cet exemple, l'utilisation d'une méthode de réduction de dimension est bénéfique pour la résolution de MB_20 et la taille du DoE initial n'influe pas sur la résolution du problème.

La Figure 3.8 montre les courbes de convergence des quatre versions d'EGORSE pour trois tailles de DoE initial pour le problème Rover_60. On constate ici que la variante EGORSE (PLS + Gaussian) converge vers des valeurs de fonction objectif plus faibles que les trois autres méthodes. De plus, lorsque la taille du DoE initial augmente, les premières itérations de l'algorithme montrent une forte amélioration de la moyenne de la meilleure valeur de fonction objectif découverte. Ce phénomène s'explique par la capacité des PLS à détecter un sous-espace de recherche pertinent grâce aux points du DoE initial. On remarque également que la version EGORSE (MGP +

Gaussian) ne montre pas de meilleures propriétés de convergence que les variantes n'utilisant pas de réduction de dimension supervisée. Ceci est certainement dû à un mauvais apprentissage des hyper-paramètres du MGP qui servent de base à la réduction de dimension. La Figure 3.8d compare la variante EGORSE (PLS + Gaussian) pour plusieurs tailles de DoE initial. On remarque ici que le DoE initial apportant les performances les plus importantes contient d points. En effet, la variante utilisant d points commence à exploiter les données du DoE plus tôt, en terme de nombre d'évaluations, que la variante utilisant $2d$ points. Au final, la méthode des PLS est la méthode de réduction la plus efficace pour résoudre le problème Rover_60 avec EGORSE. De plus, une taille de DoE initial de d points permet une exploitation des données du DoE tout en permettant une recherche précoce dans le domaine de conception et produit une convergence plus rapide qu'avec un DoE initial de 5 ou $2d$ points.

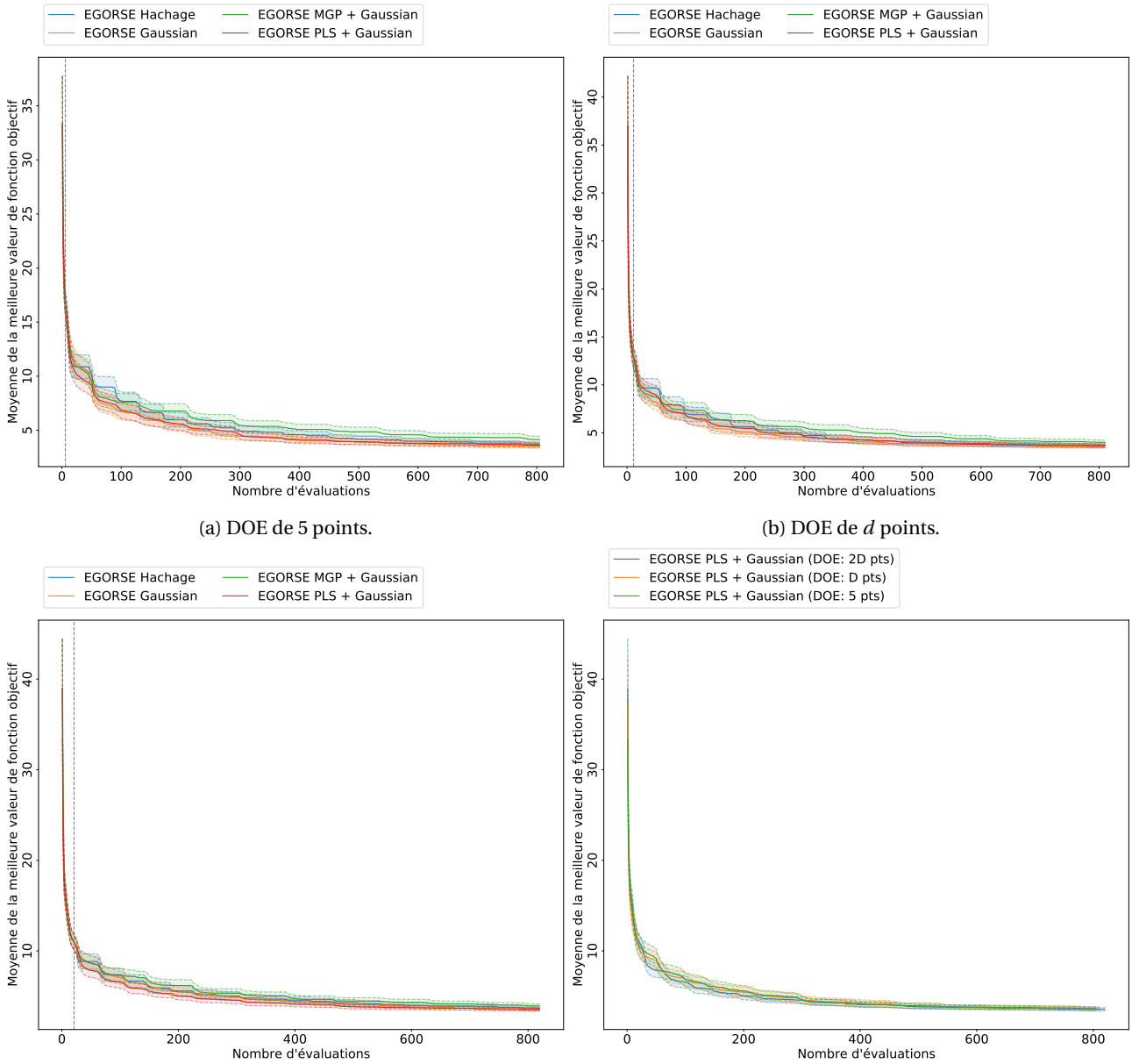
En ce qui concerne la résolution du problème MB_100 avec les différentes stratégies EGORSE, la Figure 3.9 montre clairement que la variante EGORSE (PLS + Gaussian) trouve les valeurs les plus faibles de la fonction objectif. De plus, on remarque qu'EGORSE (MGP + Gaussian) trouve également des valeurs plus faibles qu'EGORSE (Gaussian) et EGORSE (Hachage). L'utilisation d'une méthode de réduction de dimension supervisée facilite donc la convergence de l'algorithme. En outre, on remarque que plus le DoE initial est grand, c.-à-d. plus il contient de points, plus le gain dans les premières itérations de l'algorithme est important. Ainsi, l'utilisation du DoE initial permet donc de guider le processus d'optimisation vers des zones de l'espace ayant des valeurs faibles. La Figure 3.9d compare les meilleures versions d'EGORSE pour chaque taille de DoE initial pour le problème MB_100. Comme pour le problème Rover_60, on constate que le DoE optimal contient d points. Ceci correspond en réalité à un compromis entre l'exploitation des données du DoE initial et la recherche du minimum dans une direction favorable.

Dans le Tableau 3.1, on rassemble la moyenne des temps de calcul unité centrale de traitement, ou central processing unit (CPU) sur les 100 optimisations réalisées par chacune des variantes d'EGORSE pour chacun des quatre problèmes avec un DoE initial de d points. On constate que

TABLEAU 3.1 – Moyenne du temps CPU (en secondes) pour 100 optimisations pour les 4 variantes d'EGORSE sur 4 problèmes pour un budget de 800 + d évaluations.

Problème \ Méthode	(PLS + Gaussian)	(MGP + Gaussian)	(Gaussian)	(Hachage)
MB_10	569,52	1 175,81	597,12	480,63
MB_20	591,02	1 780,74	621,06	476,34
MB_100	594,30	22 370,18	634,94	473,69
Rover_60	553,69	7 312,96	562,52	523,28

les variantes EGORSE (PLS + Gaussian), EGORSE (Gaussian) et EGORSE (Hachage) réalisent les optimisations en un temps quasiment fixe quelque soit le problème considéré. Au contraire, la variante EGORSE (MGP + Gaussian) prend 19 fois plus de temps à réaliser l'optimisation de MB_100 qu'à résoudre MB_10. Ceci est certainement dû à la phase d'apprentissage du MGP par maximisation de la fonction de vraisemblance. En effet, comme indiqué dans la Section 2.2.3, cette optimisation peut s'avérer coûteuse en temps de calcul malgré l'utilisation des dérivées de la fonction de vraisemblance. Ainsi, la version EGORSE (MGP + Gaussian) n'est pas adaptée à la résolution de problèmes de grande taille.



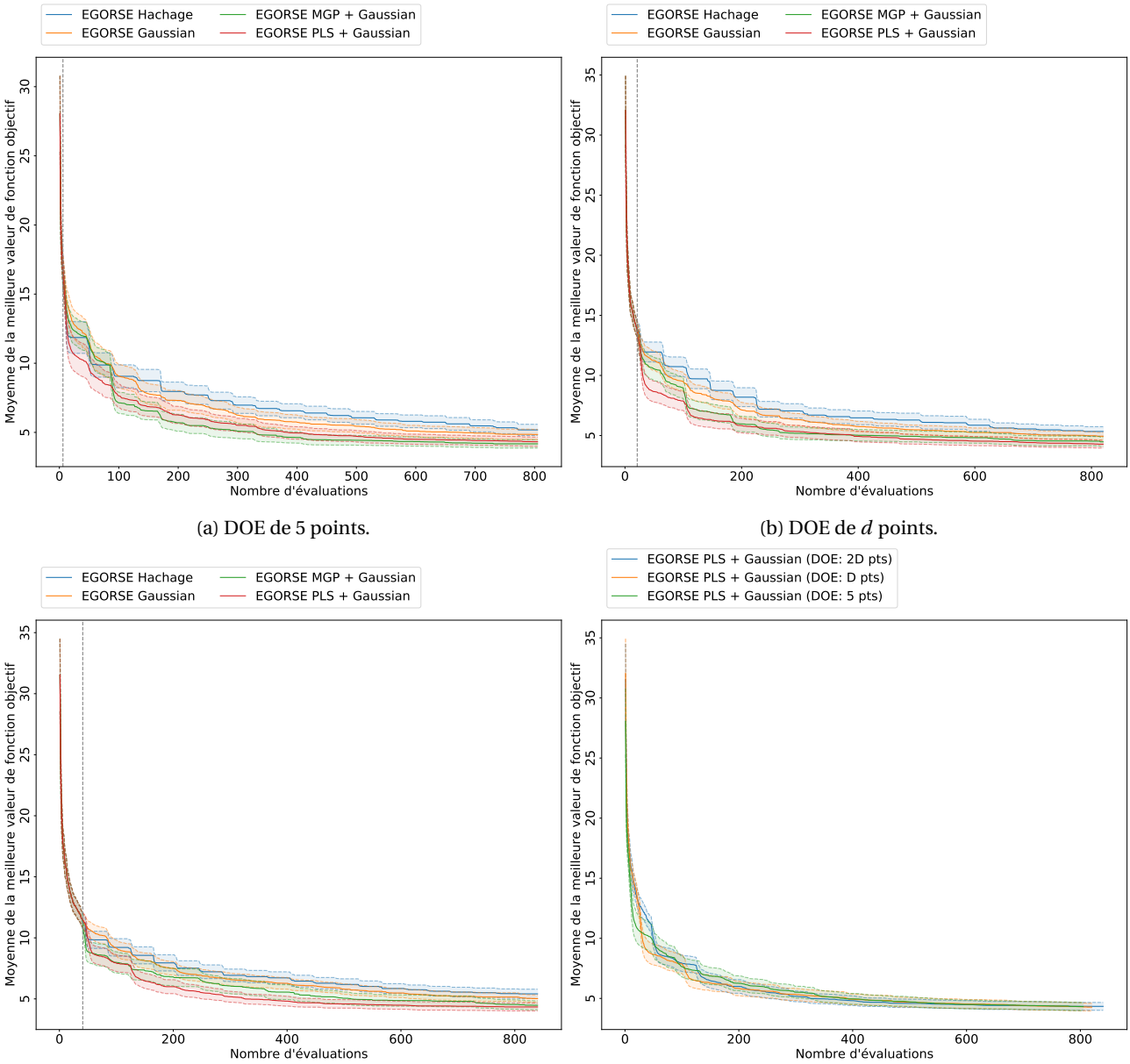
(a) DOE de 5 points.

(b) DOE de d points.

(c) DOE de $2d$ points.

(d) Meilleures variantes pour chaque taille de DOE.

FIGURE 3.6 – Courbes de convergence de l'optimisation du problème MB_10 par EGORSE pour quatre méthodes de réduction de dimension et trois tailles de DOE initial.



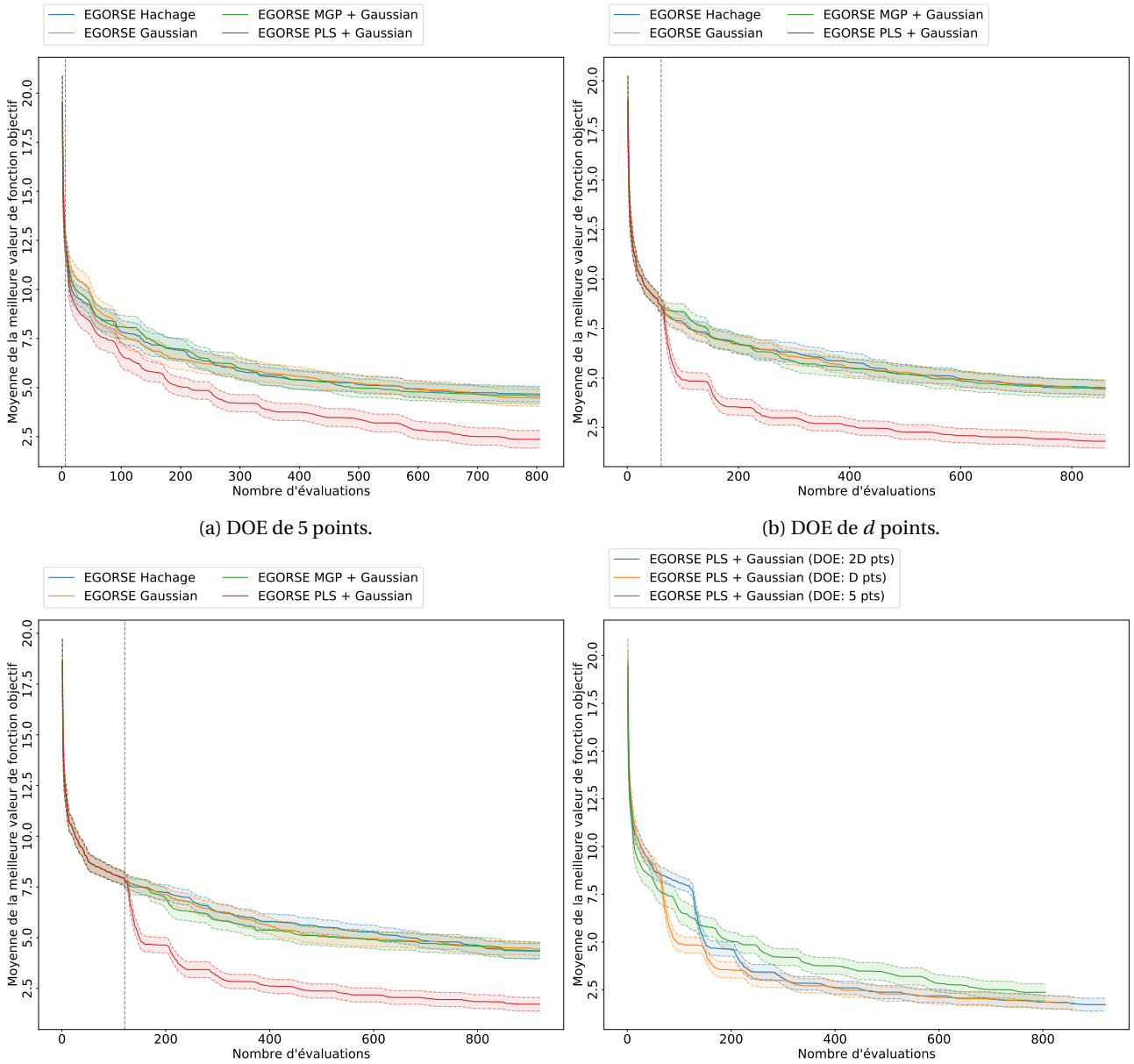
(a) DOE de 5 points.

(b) DOE de d points.

(c) DOE de $2d$ points.

(d) Meilleures variantes pour chaque taille de DOE.

FIGURE 3.7 – Courbes de convergence de l'optimisation du problème MB_20 par EGORSE pour quatre méthodes de réduction de dimension et trois tailles de DOE initial.



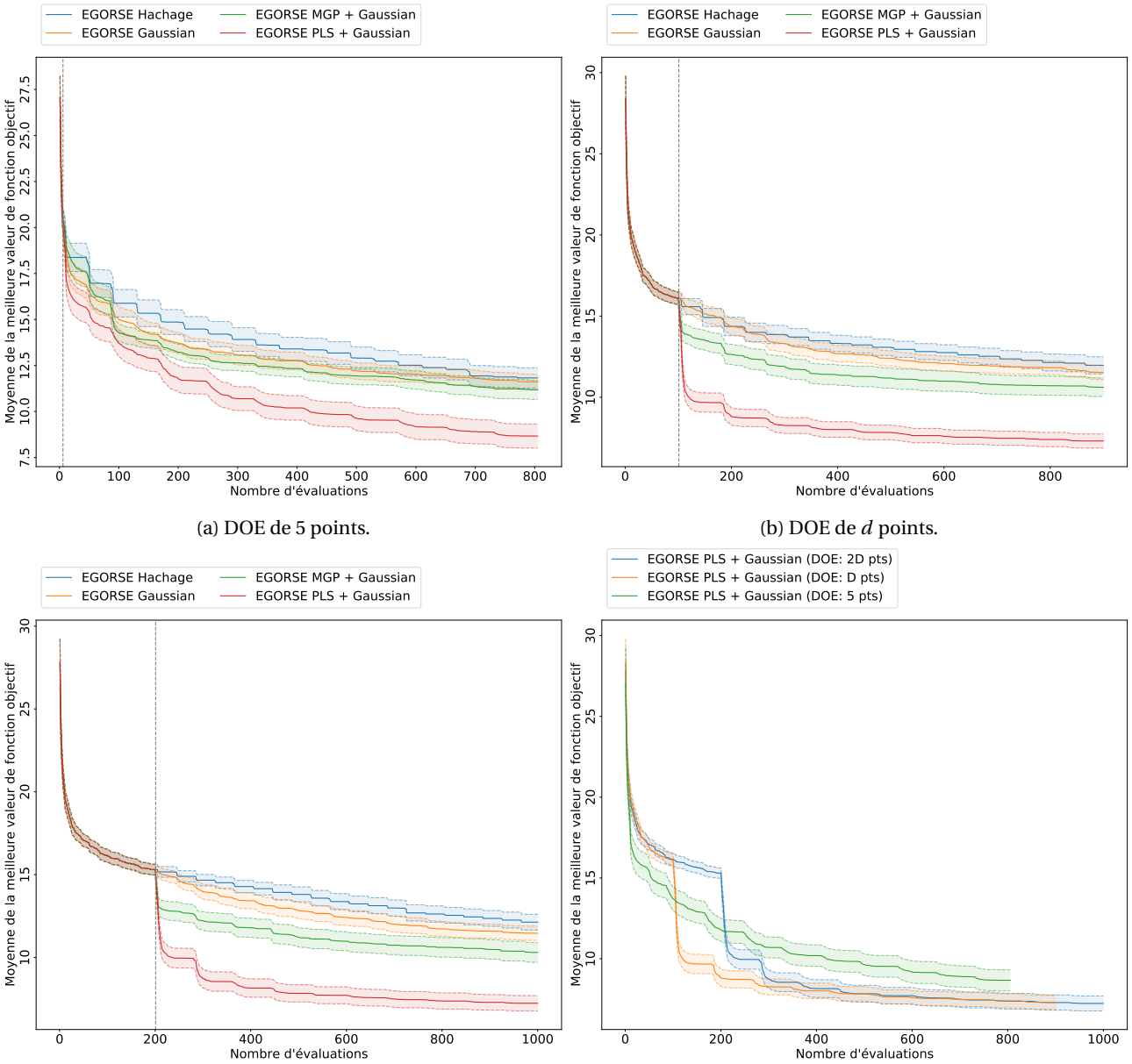
(a) DOE de 5 points.

(b) DOE de d points.

(c) DOE de $2d$ points.

(d) Meilleures variantes pour chaque taille de DOE.

FIGURE 3.8 – Courbes de convergence de l'optimisation du problème Rover_60 par EGORSE pour quatre méthodes de réduction de dimension et trois tailles de DOE initial.



(a) DOE de 5 points.

(b) DOE de d points.

(c) DOE de $2d$ points.

(d) Meilleures variantes pour chaque taille de DOE.

FIGURE 3.9 – Courbes de convergence de l'optimisation du problème MB_100 par EGORSE pour quatre méthodes de réduction de dimension et trois tailles de DOE initial.

Pour conclure sur l'étude de l'impact de la taille du DoE initial et l'impact des méthodes de réduction de dimension, on constate qu'un changement de méthode de réduction de dimension ne montre pas d'intérêt lorsque le nombre de variables de conception est trop faible (MB_10, MB_20). Au contraire, la méthode des PLS montre les meilleures performances en terme de valeurs obtenues lorsque le nombre de variables de conception augmente (MB_100, MB_60). De plus, on remarque que la taille du DoE initial influe beaucoup sur les performances de l'algorithme lorsque le nombre de variables de conception est important (MB_100, MB_60). En effet

— un DoE de 5 points offre une convergence lente vers une valeur forte de la fonction objectif,

— un DoE de $2d$ points offre une convergence vers une valeur faible de la fonction objectif,

— un DoE de d points offre une convergence rapide vers une valeur faible de la fonction objectif.

En outre, la version EGORSE (MGP + Gaussian) souffre de la malédiction de la dimension ce qui ne permet pas son utilisation dans un contexte HDBO. Au final, la variante d'EGORSE qui apporte les meilleurs résultats en terme de rapidité et de valeur d'optimum obtenue est EGORSE (PLS + Gaussian). Toutefois, l'ensemble de ces tests n'est réalisé que sur des problèmes comportant moins de 100 variables de conception. Dans la section suivante, on compare ces différentes versions sur un problème de dimension $d = 1000$.

3.2.4 Étude sur un problème de grande taille

On étudie ici l'impact des paramètres d'EGORSE sur un problème de taille $d = 1\,000$. Pour cette étude, on commence par introduire le problème considéré. Ensuite, on présente les détails relatifs aux tests. Enfin, on analyse les résultats de ces tests.

3.2.4.1 Problème de grande taille et détails relatif aux tests

Le problème de grande taille est du type MB_d, comme introduit dans la Section 3.2.3.1, avec $d = 1\,000$. Ainsi, on crée un problème non-linéaire dépendant linéairement de $d_e = 2$ variables efficaces et considérant $d = 1\,000$ variables de conception.

Pour réaliser la comparaison des différents paramètres sur ce problème de grande taille, on modifie légèrement la planification des tests par rapport à la Section 3.2.3. En réalité, on garde la même planification sauf que l'on ne réalise plus que 10 optimisations par version d'EGORSE. De plus, on augmente le nombre d'itérations à 100 pour réussir à explorer suffisamment le domaine de conception.

Pour cette étude avec le problème de grande taille, on ne considère pas les variantes utilisant les MGP comme méthode de réduction de dimension supervisée. En effet, le temps nécessaire à la réalisation d'une optimisation avec EGORSE (MGP + Gaussian) est long pour les problèmes de taille moyenne (voir Tableau 3.1). On peut donc supposer que ce phénomène est accentué pour un problème de grande taille à cause de la nécessité de construire un MGP en dimension $d = 1000$.

3.2.4.2 Analyse des résultats

Dans cette section, on analyse les résultats obtenus sur MB_1000 avec les 3 variantes d'EGORSE testées. Pour cela, on utilise des courbes de convergence comme introduites dans la Section 3.2.3.2.

La Figure 3.10 montre les résultats de 3 variantes d'EGORSE initialisées avec 3 tailles différentes de DoE initial. On constate que la version EGORSE (PLS + Gaussian) donne les meilleurs résultats

en terme de vitesse et de valeurs de fonction objectif découvertes. De plus, on remarque, encore une fois, un fort gain en début d'optimisation pour les variantes EGORSE (PLS + Gaussian) utilisant des DoE initiaux de d ou $2d$ points.

Les versions EGORSE (PLS + Gaussian) utilisant différentes tailles de DoE initial sont comparées sur la Figure 3.10d. La variante utilisant un DoE de $2d$ points converge vers des valeurs de fonction objectif plus faibles que les deux autres. De plus, on remarque que la version utilisant un DoE initial de d points stagne après 2000 évaluations de la fonction.

Pour résumer, les tests réalisés sur le problème MB_1000 permettent de valider les résultats obtenus dans la Section 3.2.3.3 sur des problèmes de taille plus faible. Ainsi, la variante EGORSE (PLS + Gaussian) est la plus adaptée pour résoudre des problèmes de grande taille. De plus, un DoE initial de $2d$ points permet la découverte de valeurs de fonction objectif plus faibles qu'avec d ou 5 points. Toutefois, le gain entre un DoE initial de d point et $2d$ points est minime.

3.2.5 Conclusion

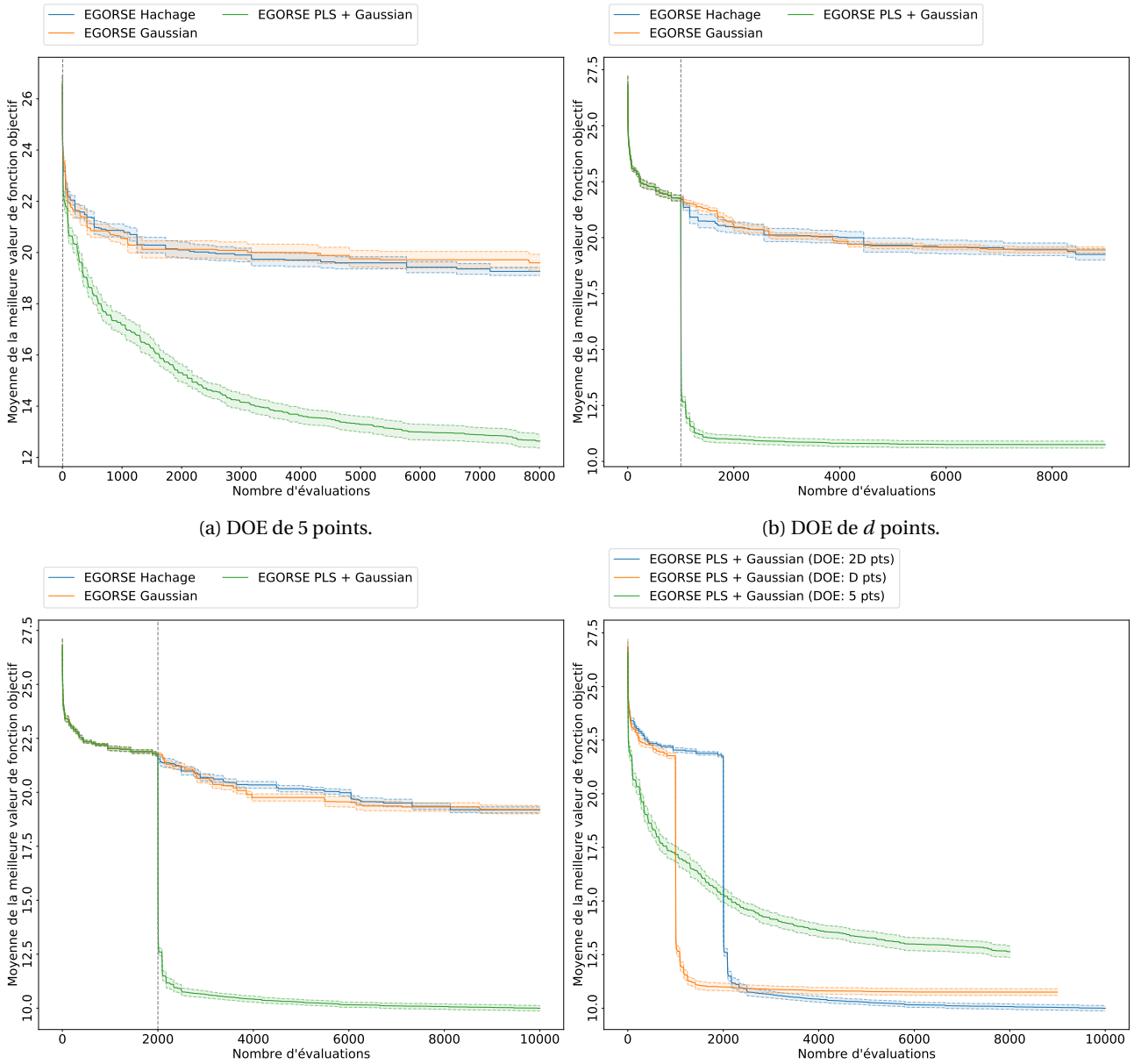
Dans cette section, on a étudié l'impact des paramètres d'EGORSE sur la rapidité et la robustesse de convergence.

Pour cela, on a d'abord présenté l'implémentation choisie pour l'algorithme EGORSE ainsi que les différents paramètres pris en compte dans cette étude (la taille du DoE initial et la méthode de réduction de dimension).

Puis on a étudié l'impact de ces paramètres sur un ensemble de quatre problèmes de taille moyenne (MB_10, MB_20, Rover_60 et MB_100). D'une part, on a constaté que la version EGORSE (PLS + Gaussian) initialisée avec un DoE de d points converge le plus rapidement vers la solution la plus faible sur les quatre problèmes de taille moyenne. Cette version utilise notamment un apprentissage supervisé du sous-espace linéaire de recherche. D'autre part, on a remarqué que les méthodes n'utilisant pas les MGP réalisent l'optimisation de ces problèmes avec un temps quasiment fixe.

Ensuite, on a validé ces conclusions sur un problème de très grande taille ($d = 1000$). On a constaté que la version EGORSE (PLS + Gaussian) initialisée avec un DoE de $2d$ points converge le mieux. Encore une fois, c'est une version qui tente d'apprendre le sous-espace linéaire qui offre les meilleures performances. Au contraire, les versions qui ne cherchent pas à apprendre ce sous-espace ne parviennent pas à trouver de valeurs aussi faibles.

Au final, on conclut que la version EGORSE (PLS + Gaussian) initialisée avec un DoE de d points est la plus appropriée pour réussir à découvrir des valeurs faibles de fonction objectif de grande dimension. Toutefois, aucune comparaison d'EGORSE n'a encore été réalisée avec les algorithmes de la littérature. Dans la section suivante, on s'intéresse donc à évaluer les performances d'EGORSE par rapport aux algorithmes de l'état de l'art.



(a) DOE de 5 points.

(b) DOE de d points.

(c) DOE de $2d$ points.

(d) Meilleures variantes pour chaque taille de DOE.

FIGURE 3.10 – Courbes de convergence de l'optimisation du problème MB_1000 par EGORSE pour trois méthodes de réduction de dimension et trois tailles de DOE initial.

3.3 Comparaison d'EGORSE avec les méthodes d'optimisation Bayésienne de la littérature

Dans cette section, on compare EGORSE avec les algorithmes BO de la littérature présentés dans la Section 2.2.3. On expose d'abord les méthodes qui sont prises en compte dans cette étude. Ensuite, on étudie les performances de ces algorithmes sur les quatre problèmes de taille moyenne de la Section 3.2.3.1. Pour finir, on valide ces résultats sur le problème de grande taille de la Section 3.2.4.1.

3.3.1 Algorithmes d'optimisation pris en compte dans la comparaison

Les algorithmes de la littérature pris en compte dans cette comparaison sont les suivants :

— TuRBO [33] : un algorithme HDBO qui utilise des régions de confiance pour favoriser l'exploitation des données du DoE. On réalise les tests avec la toolbox Python TuRBO [1] [33].

— EGO-KPLS [18] : une méthode HDBO reposant sur la réduction du nombre d'hyper-paramètres des GP. Ceci permet l'accélération de la construction des GP. On utilise en particulier la toolbox Python SEGOMOE [9]. Tous les paramètres de cet algorithme sont ceux par défaut de la toolbox. On considère deux composantes principales dans la construction des modèles KPLS.

— RREMBO [15] : une méthode HDBO utilisant la génération d'une matrice aléatoire gaussienne pour réduire le nombre de dimensions du problème. L'implémentation de la toolbox R RREMBO [2] est utilisée pour cette méthode. Les paramètres sont également choisis par défaut.

— HeSBO [74] : un algorithme HDBO qui emploie des tables de hachage pour générer une matrice de réduction. On utilise l'implémentation de la toolbox Python HeSBO. Les paramètres sont également choisis par défaut.

Pour EGORSE, on choisit la variante qui montre les meilleures performances en terme de rapidité et de robustesse de convergence dans la Section 3.2, c.-à-d. EGORSE (PLS + Gaussian) avec un DoE initial de d points.

3.3.2 Comparaison sur les quatre problèmes de taille moyenne

Dans cette section, on compare EGORSE avec les algorithmes BO de l'état de l'art sur les quatre problèmes de taille moyenne introduits dans la Section 3.2.3.1.

3.3.2.1 Détails relatifs aux tests

On introduit ici les détails nécessaires à la réalisation des tests avec EGORSE, TuRBO, EGO-KPLS, RREMBO et HeSBO.

En ce qui concerne EGORSE, on garde la même planification des tests que dans la Section 3.2.3.2.

On utilise 5 régions de confiance et un maximum de $800 + d$ évaluations de la fonction objectif pour TuRBO, avec d le nombre de variables de conception dans le problème considéré. Comme

1. https://github.com/uber-research/TuRBO
2. https://github.com/mbinois/RRembo

on ne peut pas fournir les mêmes DoE initiaux qu'EGORSE à la méthode TuRBO, on impose que le nombre total de points générés en début d'optimisation soit de d points. La fonction d'acquisition utilisée est l'EI.

Pour EGO-KPLS, on réalise les optimisations avec un budget maximal de $800 + d$ évaluations, les DoE initiaux utilisés par EGORSE et l'EI comme fonction d'acquisition.

Enfin et pour être cohérent avec l'implémentation d'EGORSE, on réalise 20 optimisations de $20d_e$ évaluations pour chacun des DoE initiaux pour RREMBO et HeSBO. Ces 20 optimisations sont ensuite mises bout à bout et considérées comme une unique optimisation. De plus, on impose le nombre de directions efficaces à $d_e = 2$ et l'utilisation de l'EI comme fonction d'acquisition. En réalité, on constate que les méthodes RREMBO et HeSBO sont respectivement équivalentes aux méthodes n'utilisant pas de réduction de dimension supervisée.

En résumé, on réalise 100 optimisations par problème pour chacune des méthodes de la littérature prises en compte.

3.3.2.2 Analyse des résultats

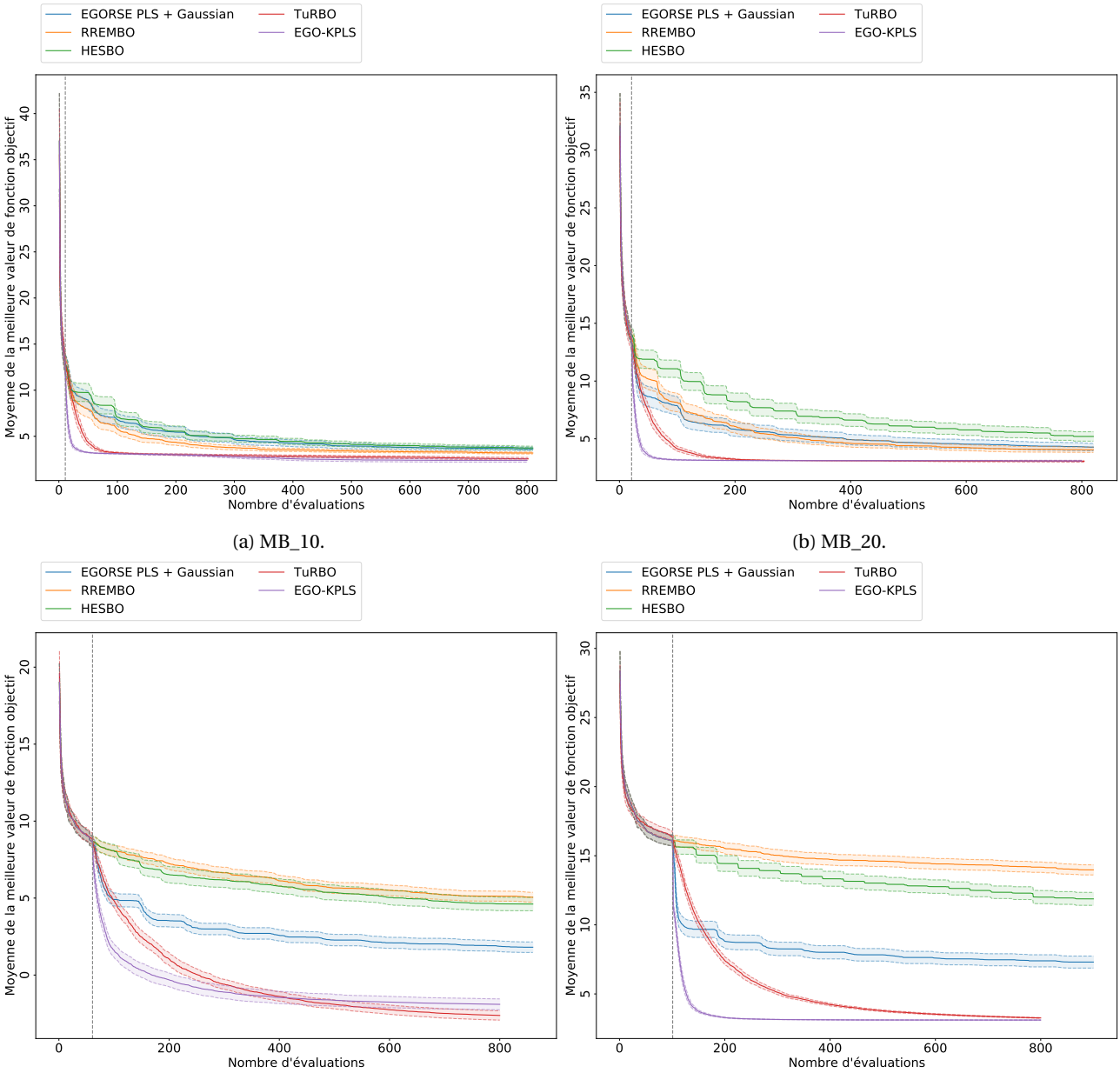
Dans cette section, on étudie la rapidité et la robustesse de convergence des quatre algorithmes BO testés sur l'ensemble de quatre problèmes de taille moyenne (MB_10, MB_20, Rover_-60 et MB_100) grâce à la méthode des courbes de convergence. La construction de ces courbes est la même que celle utilisée dans la Section 3.2.3.2.

Les courbes de convergence pour les 5 algorithmes considérés et réalisées sur les quatre problèmes de taille moyenne sont représentées par la Figure 3.11.

Les Figures 3.11a 3.11b, montrent les courbes de convergence pour les problèmes MB_10 et MB_20. On constate que TuRBO et EGO-KPLS convergent le plus rapidement et avec un écart type très faible vers les valeurs de fonction objectif les plus basses. En effet, ces deux algorithmes réalisent une recherche dans le domaine complet ce qui leur permet de l'explorer et de trouver rapidement des valeurs de fonction objectif faibles. Toutefois, ils ne sont pas capables de trouver la valeur optimale de la fonction ($f_{min} \approx 1.1$). De plus, on remarque qu'EGORSE, RREMBO et HeSBO trouvent des valeurs de fonction objectif similaires. En effet, ces méthodes restreignent leur recherche à un sous-espace de dimension plus faible ce qui limite leur capacité d'exploration.

Pour les problèmes Rover_60 et MB_100, les Figures 3.11c 3.11d soulignent le même comportement des algorithmes. On constate que TuRBO et EGO-KPLS découvrent toujours les meilleures valeurs de fonction objectif avec un écart type faible. En effet, ils réalisent toujours une optimisation dans le domaine de conception initial ce qui leur permet de découvrir les zones de valeurs les plus faibles. On remarque également qu'EGORSE semble plus performant que RREMBO et HeSBO. Ce phénomène est certainement dû à la méthode de réduction de dimension supervisée utilisée par EGORSE qui lui permet de se concentrer sur le sous-espace de plus forte variation de la fonction.

Afin d'évaluer la rapidité d'exécution de ces algorithmes en terme de temps de calcul CPU, on rassemble dans le Tableau 3.2 la moyenne des temps de calcul nécessaires à la réalisation d'une optimisation. On trace également, sur la Figure 3.12, l'évolution de la moyenne des meilleures valeurs de fonction objective découvertes en fonction du temps de calcul CPU. On constate que les méthodes travaillant dans le domaine de conception initial (TuRBO et EGO-KPLS) utilisent beaucoup plus de temps de calcul CPU pour réaliser l'optimisation que les méthodes utilisant des



(a) MB_10.

(b) MB_20.

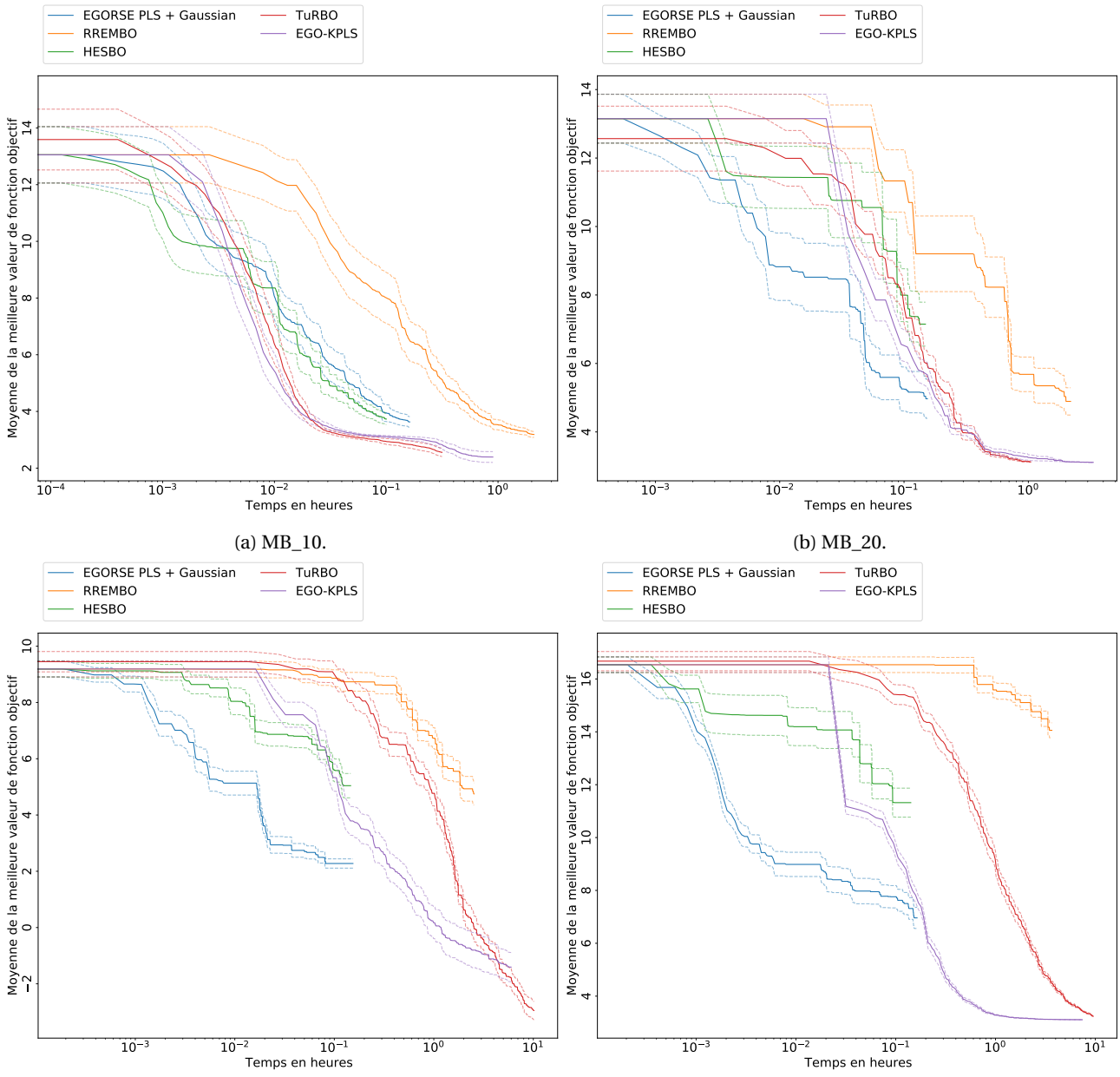
(c) Rover_60.

(d) MB_100.

FIGURE 3.11 – Courbes de convergence de HESBO, RREMBO, TuRBO, EGO-KPLS et EGORSE pour les quatre problèmes avec un DOE initial de taille d.

TABLEAU 3.2 – Moyenne du temps CPU (en secondes) pour 100 optimisations pour EGORSE, TuRBO, EGO-KPLS, RREMBO et HeSBO sur 4 problèmes pour un budget d'environ $800 + d$ évaluations.

Problème \ Méthode	EGORSE	TuRBO	EGO-KPLS	RREMBO	HeSBO
MB_10	569,52	1 102,52	4 938,36	8 156,97	466,97
MB_20	591,02	3 854,34	16 039,25	8 155,35	534,02
MB_100	594,30	33 920,14	27 111,06	15 059,45	472,01
Rover_60	553,69	37 246,25	30 423,94	523,28	532,09



(a) MB_10.

(b) MB_20.

(c) Rover_60.

(d) MB_100.

FIGURE 3.12 – Courbes de convergence en temps de calcul CPU de HESBO, RREMBO, TuRBO, EGO-KPLS et EGORSE pour les quatre problèmes avec un DOE initial de taille d.

sous-espaces de recherche (EGORSE, RREMBO, HeSBO). Ainsi, ces méthodes sont utilisables pour résoudre des problèmes de taille moyenne ($d \leq 100$) mais sont difficilement utilisables sur des problèmes de taille plus importante. De plus, RREMBO montre également des temps d'optimisation important. Ce phénomène se produit à cause de la résolution d'un problème quadratique à chaque évaluation de la fonction d'acquisition dans le sous-problème d'optimisation. Ainsi, RREMBO est difficilement utilisable pour résoudre des problèmes de très grande taille ($d \gg 100$).

Dans cette section, on a mis en évidence la supériorité de TuRBO et EGO-KPLS pour la résolution de problèmes de taille moyenne ($d \leq 100$) par rapport à EGORSE, RREMBO et HeSBO. Toutefois, les temps de calcul CPU nécessaires à TuRBO et EGO-KPLS pour réaliser l'optimisation sont prohibitifs et ne permettent pas d'utiliser ces algorithmes sur des problèmes de grande taille ($d \gg 100$). De même, le temps de calcul CPU utilisé par RREMBO rend son utilisation impossible en plus grande dimension. Enfin, EGORSE donne les meilleurs résultats si on se restreint aux méthodes utilisant des sous-espaces linéaires de recherche. Cependant, on n'a pas encore étudié les performances de ces algorithmes sur un problème de grande taille. Dans la section suivante, on évalue la rapidité et la robustesse de convergence des méthodes de la littérature sur le problème MB_1000.

3.3.3 Comparaison sur le problème de grande taille

On compare EGORSE aux algorithmes de la littérature sur le problème de taille MB_1000. Pour cela, on présente d'abord les détails nécessaires à la réalisation des tests. Ensuite, on commente les résultats de ces tests.

3.3.3.1 Détails relatif aux tests et limites

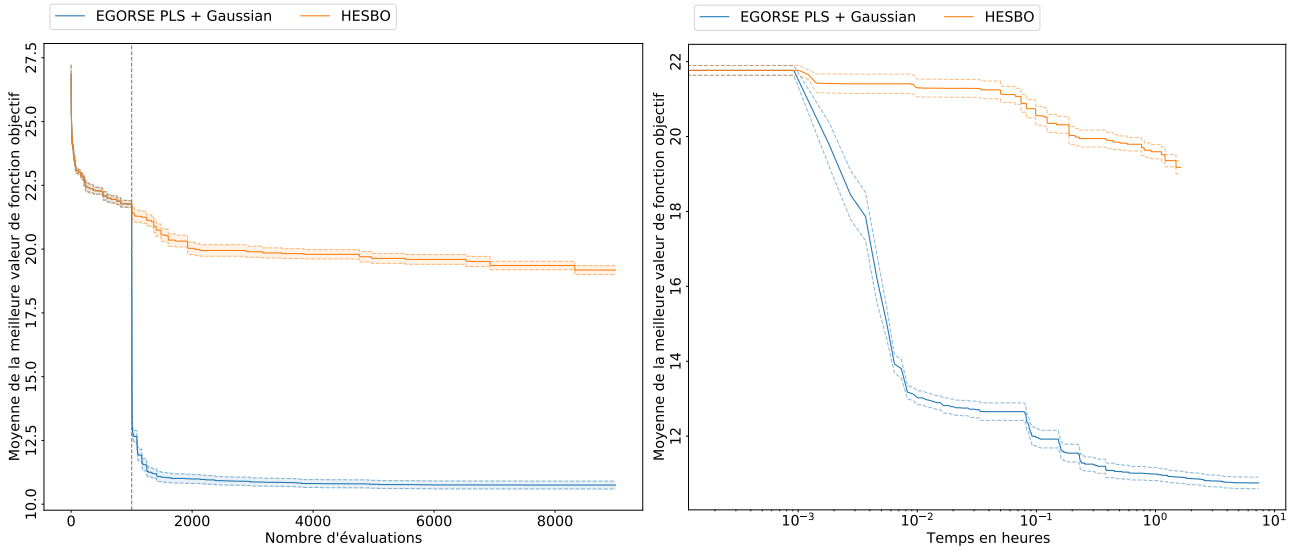
Pour réaliser la comparaison des différents algorithmes sur ce problème de grande taille, on modifie légèrement la planification des tests par rapport à la Section 3.3.2.1. En réalité, on garde la même planification sauf que l'on ne réalise plus que 10 optimisations par algorithme. De plus, on augmente le nombre d'itérations à 100 pour réussir à explorer suffisamment le domaine de conception.

Pour cette étude avec le problème de grande taille, on ne considère pas les méthodes RREMBO, TuRBO et EGO-KPLS qui seraient trop gourmandes en temps de calcul CPU. En effet, le temps nécessaire à la réalisation d'une optimisation avec ces méthodes est long pour les problèmes de taille moyenne (voir Tableau 3.2). On peut donc supposer que ce phénomène est accentué pour un problème de dimension $d = 1000$.

On souligne que RREMBO est une extension de l'algorithme REMBO. REMBO ne nécessite pas la résolution d'un problème quadratique à chaque évaluation de la fonction d'acquisition et pourrait être intégré dans cette étude. Cependant les tests préliminaires réalisés avec REMBO sur MB_1000 n'ont pas permis de découvrir des points de plus faible valeur que ceux contenus dans le DoE initial. C'est pourquoi, ni RREMBO ni REMBO ne sont inclus dans cette étude.

3.3.3.2 Analyse des résultats

Les courbes de convergence de la Figure 3.13a montrent qu'EGORSE donne les valeurs de fonction objectif les plus faibles. On remarque une gain important en début d'optimisation et en-



(a) Courbe de convergence en nombre d'évaluations.

(b) Courbe de convergence en temps de calcul CPU.

FIGURE 3.13 – Courbes de convergence en temps de calcul CPU et en nombre d'évaluation de HESBO et EGORSE pour le problème MB_1000 avec un DOE initial de taille d.

suite l'algorithme stagne autour d'une moyenne $f = 11$ alors que la valeur optimale est $f_{min} \approx 1.1$. La méthode EGORSE n'est donc pas capable de résoudre le problème MB_1000 bien qu'elle offre les meilleures performances. Ceci suggère que la méthode de réduction de dimension n'est pas capable de découvrir un sous-espace linéaire contenant le minimum de la fonction. On peut imaginer que cela est dû soit au nombre de directions efficaces ($d_e = 2$) considéré par les méthodes de réduction de dimension supervisées qui peut être trop faible, soit au nombre de directions efficaces ($d_e = 2$) considéré par les méthodes de réduction de dimension non-supervisées qui peut également être trop faible. En utilisant un plus grand nombre de directions efficaces, on agrandirait l'espace de recherche ce qui pourrait permettre de trouver de meilleures valeurs.

En terme de temps de calcul CPU, la Figure 3.13b montre qu'EGORSE prend en moyenne 28 185,10 secondes et HeSBO prend en moyenne 5 901,55 secondes pour réaliser une optimisation du problème MB_1000 avec un DoE initial de d points. On constate que HeSBO réalise l'optimisation 7 fois plus vite qu'EGORSE. Ceci est certainement dû à la méthode de retour du sous-espace $\mathcal{B}$ vers Ω à calculer à chaque évaluation. En effet, EGORSE nécessite la résolution d'un problème quadratique à chaque évaluation de la fonction objectif ce qui n'est pas le cas de HeSBO. Pour réduire le temps de calcul CPU d'EGORSE, on peut penser à utiliser la même méthode de réduction de dimension non-supervisée que HeSBO. On peut ainsi espérer diviser par 2 le temps de calcul CPU d'EGORSE puisque la méthode de réduction de dimension non-supervisée est utilisée pour la moitié des évaluations.

3.3.4 Conclusion

Dans cette section, on a comparé EGORSE (PLS + Gaussian) aux autres méthodes HDBO de la littérature.

Pour cela, on a d'abord introduit les différentes méthodes HDBO prises en compte qui sont HeSBO, RREMBO, TuRBO et EGO-KPLS.

Ensuite, on a étudié la convergence de ces algorithmes sur cinq problèmes de la dimension

10 à 1000 (MB_10, MB_20, Rover_60 et MB_1000). On a testé les différentes méthodes avec un DoE initial de d points. Ceci a permis de déduire que EGO-KPLS et TuRBO sont les algorithmes les plus performants pour résoudre des problèmes de taille moyenne. De plus, EGORSE est l'algorithme utilisant une méthode de réduction de dimension qui trouve des valeurs de fonction objectif les plus faibles. En ce qui concerne les temps moyens de calcul CPU, on remarque que TuRBO et EGO-KPLS sont beaucoup plus coûteux. Ceci ne permet pas leur utilisation sur un problème comprenant un très grand nombre de variables de conception ($d \gg 100$). De la même manière, RREMBO réalise les optimisations en un temps beaucoup plus long qu'EGORSE et HeSBO notamment sur les problèmes comportant le plus de variables de conception. Ceci est dû à la résolution d'un problème quadratique à chaque évaluation de la fonction d'acquisition ce qui n'est pas le cas d'EGORSE et de HeSBO. De plus, HeSBO est aussi plus rapide qu'EGORSE car il ne résout aucun problème quadratique alors qu'EGORSE en résout un à chaque évaluation de la fonction objectif. Au final EGORSE apparaît comme la meilleure méthode pour résoudre des problèmes comprenant un grand nombre de variables de conception. Les courbes de convergence avec des DoE initiaux de 5 et $2d$ points sont disponible en Annexe A.2.

On valide les hypothèses précédentes sur un problème de grande taille (MB_1000). Toutefois, il n'est pas possible de réaliser les tests avec les méthodes EGO-KPLS, TuRBO et RREMBO à cause de leur temps de calcul CPU prohibitif. On ne compare donc que HeSBO et EGORSE. On remarque qu'EGORSE trouve des valeurs de fonction objectif bien plus faibles que HeSBO. En effet, EGORSE détecte les zones de l'espace intéressantes en utilisant des méthodes de réduction supervisées. C'est pourquoi, on observe un fort gain en début d'optimisation lorsque le DoE initial est suffisamment grand. Les courbes de convergence avec des DoE initiaux de 5 et $2d$ points sont disponible en Annexe A.2.

Pour résumer, lorsque le nombre de variables de conception n'est pas trop grand ($d \leq 100$), les méthodes TuRBO et EGO-KPLS doivent être privilégiées puisqu'elles convergent vers les valeurs les plus faibles de fonction objectif. Pour les problèmes avec plus de variables de conception, EGORSE (PLS + Gaussian) est plus performant. De plus, EGORSE offre un fort gain en début d'optimisation grâce à la recherche dans le sous-espace des directions principales découvertes par PLS.

3.4 Synthèse du chapitre

Dans ce chapitre, l'objectif était de mettre en place une méthode capable de résoudre des problèmes sans contrainte comprenant un grand nombre de variables de conception ($d \gg 100$) ayant un temps de calcul CPU raisonnable.

Pour cela, on a d'abord étudié les méthodes de la littérature pour identifier celle qui permet de résoudre de plus efficacement ce type de problèmes. La méthode RREMBO est apparue comme la plus appropriée pour prendre en compte des problèmes de grande taille bien qu'elle repose sur un nombre de résolutions de problèmes quadratiques important (une résolution par évaluation de fonction d'acquisition) et une sélection aléatoire du sous-espace de recherche. Cette sélection aléatoire ne garantit donc pas que le minimum de la fonction est inclus dans le sous-espace. On a tenté de lever ces deux verrous en développant une nouvelle méthode HDBO nommée EGORSE. Elle réalise une optimisation sous contrainte dans le sous-espace de recherche considéré afin de

ne résoudre qu'un unique problème quadratique lors des évaluations de la fonction objectif. On réduit donc drastiquement le nombre de problèmes quadratiques à résoudre lors d'un processus complet d'optimisation comparé à la méthode RREMBO. De plus, on emploie des méthodes de réduction de dimension supervisées pour sélectionner les directions efficaces de recherche. Celles-ci utilisent les données du DoE pour découvrir ces directions efficaces. Ainsi, la recherche du minimum est effectuée dans les directions de plus fortes variations. Enfin, on ajoute un processus d'apprentissage adaptatif de ces sous-espaces de recherche.

EGORSE dépend donc de deux paramètres majeurs : le nombre de points dans le DoE initial, qui est utilisé pour découvrir les premières directions efficaces, et les méthodes de réduction de dimension. C'est pourquoi on a étudié l'impact du nombre de points dans le DoE initial, à travers 3 valeurs, et le type de réduction de dimension, avec les méthodes PLS, MGP, matrices aléatoires gaussiennes et matrices aléatoires à base de tables de hachage. On a donc testé quatre versions d'EGORSE (EGORSE (PLS + Gaussian), EGORSE (MGP + Gaussian), EGORSE (Gaussian) et EGORSE (Hachage)), sur un ensemble de quatre problèmes de taille moyenne (MB_10, MB_-20, Rover_60 et MB_100) pour trois tailles de DoE initial. On a constaté que l'utilisation d'un DoE initial de grande taille ne facilite pas la convergence pour les problèmes de petite taille (MB_10, MB_20) mais devient intéressant lorsque le nombre de variables de conception augmente (MB_-100, Rover_60). De la même manière, l'emploi de méthodes de réduction de dimension supervisées est utile lorsque le nombre de variables de conception est important. En réalité, EGORSE (PLS + Gaussian) initialisé avec un DoE initial de d points découvre les valeurs de fonction objectif les plus faibles sur l'ensemble des problèmes. Pour valider ces conclusions sur un problème de grande taille, on a comparé ces versions sur le problème MB_1000 comportant $d = 1000$ variables de conception. Les même conclusions peuvent être apportées. On remarque toutefois un fort gain un début d'optimisation pour EGORSE (PLS + Gaussian). Ce gain est dû à la bonne sélection du sous-espace de recherche qui permet de se concentrer sur le sous-espace de plus fortes variations de la fonction objectif.

Ensuite, on a comparé la méthode EGORSE (PLS + Gaussian) avec les algorithmes de la littérature, c.-à-d. TuRBO, EGO-KPLS, HeSBO et RREMBO, sur MB_10, MB_20, Rover_60, MB_100 et MB_1000. En ce qui concerne les résultats sur les problèmes de taille moyenne ($d \leq 100$), ils montrent que EGO-KPLS et TuRBO découvrent des valeurs de fonction objectif plus faible qu'EGORSE, HeSBO et RREMBO. Ces performances s'expliquent par leur capacité à travailler dans l'espace Ω initial. Toutefois, le temps de calcul CPU nécessaire est beaucoup plus important que pour EGORSE et HeSBO. Ils ne sont donc pas utilisables sur des problèmes de grande taille ($d \gg 100$). Les bonnes performances d'EGORSE et de HeSBO en temps de calcul s'expliquent par la réalisation du processus BO dans un sous-espace de dimension $d_e = 2$. RREMBO réalise aussi le processus BO dans un sous-espace de dimension $d_e = 2$ mais le nombre important de problèmes quadratiques à résoudre le rende beaucoup plus lent. En réalité, il n'est pas possible d'utiliser RREMBO sur des problèmes de grande taille. Au final, seuls EGORSE et HeSBO sont capables de résoudre ce type de problèmes. On compare donc EGORSE et HeSBO sur le problème MB_1000. On constate qu'EGORSE fournit des valeurs de fonction objectif bien plus faibles que HeSBO. Toutefois, HeSBO prend 7 fois moins de temps qu'EGORSE pour réaliser l'optimisation complète. On suspecte que cela est dû à la résolution de problèmes quadratiques à chaque évaluation de la fonction objectif par EGORSE ce qui n'est pas nécessaire pour HeSBO.

En résumé, EGORSE (PLS + Gaussian) est la méthode la plus appropriée pour résoudre des problèmes de très grande taille bien que le temps de calcul CPU nécessaire à la réalisation est important comparé à la méthode HeSBO. Il est toutefois plus raisonnable d'utiliser les méthodes TuRBO et EGO-KPLS sur les problèmes de taille plus restreinte ($d \leq 100$).

On souligne que lors de la rédaction du manuscrit, un article décrivant une méthode du même type, nommée sequential-subspace-search bayesian functional optimisation (S^3-BFO), a été publiée par SHILTON et al. [118]. Elle repose sur la construction d'un GP pour détecter les directions de recherche et centre le sous-espace de recherche au meilleur point du DoE.

Récapitulatif du chapitre

On a répondu aux objectifs introduits en début de chapitre grâce aux points suivants :

- Reformulation du problème d'optimisation pour réduire le nombre de résolutions de problèmes quadratiques
- Utilisation de méthodes de réduction de dimension supervisées (PLS et MGP)
- Implémentation d'un processus d'apprentissage adaptatif des directions de recherche à favoriser.
- Étude de l'impact de la taille du DoE initial sur 5 problèmes analytiques ($10 \leq d \leq 1000$).
- Étude de l'impact des méthodes de réduction (MGP, PLS, matrice aléatoire gaussienne, matrice à base de tables de hachages) sur 5 problèmes analytiques ($10 \leq d \leq 1000$).
- Comparaison de la méthode proposée avec les algorithmes BO de la littérature (TuRBO, HeSBO, EGO-KPLS, RREMBO.)

Chapitre 4

Développement et étude d'un critère de faisabilité pour l'optimisation Bayésienne sous contraintes multi-modales mixtes

« Nous nous fabriquons souvent nous-mêmes nos propres prisons. Mais on peut aussi créer sa propre liberté. »

Robin Hobb

Sommaire

Objectifs du chapitre

- Proposer une méthode d'optimisation Bayésienne pour les problèmes d'optimisation comprenant un grand nombre de contraintes multi-modales mixtes et un grand nombre de variables de conception.
- Évaluer l'impact des paramètres de la méthode proposée sur la rapidité et la robustesse de convergence.
- Comparer la méthode proposée aux algorithmes d'optimisation de la littérature.

Dans ce chapitre, on s'intéresse aux problèmes d'optimisation coûteux à évaluer en temps de calcul et comprenant des contraintes multi-modales d'égalité et/ou d'inégalité

$$\min_{\boldsymbol{x}\in\Omega} \left\{ f(\boldsymbol{x}) \text{ s.c. } \boldsymbol{g}(\boldsymbol{x}) \geq 0 \text{ et } \boldsymbol{h}(\boldsymbol{x}) = 0 \right\}, \tag{4.1}$$

où $f : \mathbb{R}^d \mapsto \mathbb{R}$ est la fonction objectif, $\boldsymbol{g} : \mathbb{R}^d \mapsto \mathbb{R}^m$ sont les contraintes d'inégalité et $\boldsymbol{h} : \mathbb{R}^d \mapsto \mathbb{R}^p$ correspondent aux contraintes d'égalité. On sous-entend par contraintes multi-modales, des contraintes qui définissent un domaine de faisabilité qui n'est pas convexe. De plus, ces contraintes peuvent être nombreuses ($m + p \approx 40$) et le nombre de variables peut être grand ($d \approx 20$). On rappelle que la fonction objectif peut être fortement multi-modale et que les contraintes sont QRSK (connues, issues de simulations, relâchables et quantifiables) suivant la taxonomie des contraintes introduite par DIGABEL et WILD [30]. En outre, aucune information sur la régularité ou sur les dérivées des fonctions n'est disponible. On est donc en présence d'un problème d'optimisation sous contraintes dit boîte noire.

Comme on l'a déjà précisé dans le Chapitre 2, on se place dans le contexte de la CBO. Cependant, les algorithmes CBO introduits dans la Section 2.3 ont des difficultés à résoudre le problème (4.1) à cause du grand nombre de variables de conception, ou de la présence de contraintes d'égalité, ou du caractère multi-modale des contraintes.

Pour lever ces verrous, on choisit d'étendre l'une des méthodes présentées dans la Section 2.3 : SEGO-UTB. Tout d'abord, on argumente la préférence de SEGO-UTB par rapport aux autres méthodes CBO. Puis, on introduit un critère de faisabilité spécifique aux contraintes d'égalité. Ensuite, on évalue l'impact des paramètres de SEGO-UTB sur la convergence du processus d'optimisation. Ceci permet notamment de mieux choisir les paramètres pour rendre SEGO-UTB le plus performant possible. Pour finir, on compare SEGO-UTB avec les algorithmes CBO de la littérature ainsi qu'avec deux algorithmes classiques d'optimisation sans dérivées (COBYLA et le nonlinear optimization with the MADS algorithm (NOMAD)).

4.1 Mise en place d'une méthode d'optimisation Bayésienne sous contraintes multi-modales mixtes

Dans cette section, on développe un algorithme CBO pour résoudre des problèmes d'optimisation comprenant une quarantaine de contraintes d'égalité et/ou d'inégalité et une vingtaine de variables (≈ 20). On identifie d'abord la méthode de la littérature qui répond le mieux à ces exigences. Ensuite, on étend cette méthode pour qu'elle puisse résoudre le problème (4.1).

4.1.1 Choix de la méthode d'optimisation

On cherche la méthode CBO de la littérature qui correspond le plus à nos spécifications, c.-à-d. qui est capable de résoudre un problème d'optimisation comportant un grand nombre de contraintes multi-modales mixtes et un grand nombre de variables de conception.

En pratique, d'après la Section 2.3.1 la plupart des méthodes reposant sur une fonction de mérite ne peuvent pas être utilisées avec plus d'une dizaine de variables à cause des phases d'estimation nécessaires à leur calcul. Les méthodes PESC, ALBO, SUR [51, 83, 84] sont donc écartées. Au contraire, l'EFI [112] est calculable en grande dimension mais un nombre important de contraintes rend cette fonction de mérite non-informative (voir Section 2.3.1.1). En effet, EFI est nulle presque partout lorsque le nombre de contraintes devient grand.

En ce qui concerne les méthodes CBO utilisant des critères de faisabilité (voir Section 2.3.2), la méthode SEGO-EV [2] semble inadaptée à la résolution de problèmes à contraintes multi-modales. En effet, on a précisé dans la Section 2.3.2.2 que SEGO-EV restreint l'exploration du domaine aux zones où les GP modélisant les contraintes ont un faible écart type et une moyenne faisable. De plus, SEGO-EV n'autorise pas l'utilisation de contraintes d'égalité. Par contre, la méthode SEGO [9, 108] est utilisable avec des contraintes d'égalité. Cependant, si les GP ont un grand écart type et une moyenne non faisable dans une zone du domaine, alors le processus d'optimisation ne visitera pas cette zone. Comme expliqué dans la Section 2.3.2.1, ceci restreint l'exploration du domaine et peut faire converger SEGO vers un optimum local. Pour finir, la méthode SEGO-UTB [63] encourage l'exploration du domaine en utilisant l'écart type des GP modélisant les contraintes. Des zones faisables peuvent ainsi être découvertes dans des domaines où l'écart type des GP est grand et la moyenne non faisable. Néanmoins, cette méthode ne prend en compte que des contraintes d'inégalité. On pourrait utiliser directement SEGO-UTB en transformant les contraintes d'égalité $h(\boldsymbol{x}) = 0$ en deux contraintes d'inégalité $h(\boldsymbol{x}) \geq \epsilon_c$ et $h(\boldsymbol{x}) \leq \epsilon_c$ mais cette transformation n'est pas souhaitable car elle augmente fortement le nombre de contraintes à prendre en compte et crée des contraintes antagonistes.

En résumé, aucune des méthodes de la littérature ne semble par permettre une résolution efficace des problèmes d'optimisation comprenant un grand nombre de contraintes multi-modales d'égalité et d'inégalité et un grand nombre de variables de conception. On souligne que seule la méthode SEGO-UTB est capable d'explorer le domaine de conception avec un coût raisonnable en temps de calcul en grande dimension. On va donc étendre cette méthode aux contraintes d'égalité afin de pouvoir résoudre le problème d'optimisation (4.1) plus efficacement.

4.1.2 Extension de la borne supérieure de confiance aux contraintes d'égalité

Dans la section précédente, on a choisi d'étendre le critère de faisabilité UTB aux contraintes d'égalité. Pour cela, on va d'abord rappeler le cadre CBO reposant sur le critère de faisabilité UTB. Ensuite, on introduira l'extension aux contraintes d'égalité.

4.1.2.1 Rappels sur la borne supérieure de confiance

En choisissant de travailler avec UTB, on se place dans le cadre plus général de la CBO liée à un critère de faisabilité et qui repose sur le sous-problème d'optimisation suivant :

$$\boldsymbol{x}^{(l+1)} \in \arg\max_{\boldsymbol{x}\in\Omega} \left\{ \alpha_f^{(l)}(\boldsymbol{x}) \quad \text{s.c.} \quad \boldsymbol{x} \in \Omega_{\boldsymbol{h}}^{(l)} \cap \Omega_{\boldsymbol{g}}^{(l)} \right\} \tag{4.2}$$

où $\alpha_f^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}$ est l'une des fonctions d'acquisition présentées dans la Section 2.2, les domaines $\Omega_{\boldsymbol{g}}^{(l)}$ et $\Omega_{\boldsymbol{h}}^{(l)}$ sont respectivement les domaines faisables définis par les critères de faisabilité $\boldsymbol{\alpha}_{\boldsymbol{g}}^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}^m$ et $\boldsymbol{\alpha}_{\boldsymbol{h}}^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}^p$. Ici, on a plus spécifiquement

$$\alpha_{\mathrm{UTB},g_i}^{(l)}(\boldsymbol{x}) = \hat{\mu}_{g_i}^{(l)}(\boldsymbol{x}) + \tau_{g_i}^{(l)} \hat{\sigma}_{g_i}^{(l)}(\boldsymbol{x}), \ \forall i \in \{1,\dots,m\}, \tag{4.3}$$

$$\Omega_{\boldsymbol{g}}^{(l)} = \left\{ \boldsymbol{x} \in \Omega, \ \alpha_{\mathrm{UTB},g_i}^{(l)}(\boldsymbol{x}) \geq 0, \ \forall i \in \{1,\dots,m\} \right\}, \tag{4.4}$$

où $\hat{\mu}_{g_i}^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}$, $\hat{\sigma}_{g_i}^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}$ et $\tau_{g_i}^{(l)} \in \mathbb{R}^+$ sont respectivement la moyenne, l'écart type et le seuil de doute du GP modélisant la i$^{\text{e}}$ contrainte d'inégalité.

Le critère UTB cherche en réalité à faciliter l'exploration du domaine en favorisant les zones avec un fort écart type. Ceci autorise la visite de zones inconnues qui peuvent être mal représentées par la moyenne des GP. De plus, la capacité d'exploration du processus est contrôlée grâce au seuil de doute $\boldsymbol{\tau}^{(l)} \in \mathbb{R}^{+^m}$. En effet, plus ce paramètre est grand, plus les zones mal classées par la moyenne des GP peuvent être explorées. Le lecteur peut se reporter à la Section 2.3.2 pour un exemple de ce phénomène sur le problème de Branin modifié [79].

4.1.2.2 Prise en compte des contraintes d'égalité

Pour étendre le critère UTB aux contraintes d'égalité, on cherche à visiter les zones où la violation des contraintes par la moyenne des GP peut être questionnée au regard des valeurs de leur écart type. Par exemple, si la violation des contraintes par la moyenne des GP est faible et que l'écart type est fort, alors on autorise la visite de cette zone mal représentée qui offre une forte probabilité de faisabilité. De plus, comme pour les contraintes d'inégalité, on souhaite relier la confiance de l'utilisateur dans la qualité de la prédiction des GP au critère de faisabilité. Ainsi, si on a une grande confiance dans la prédiction des GP, le processus d'optimisation ne doit pas visiter les zones avec une forte violation des contraintes par la moyenne des GP. Au contraire, si la confiance est faible, on autorise la visite des zones où la violation des contraintes par la moyenne des GP est forte.

En s'inspirant du critère UTB pour les contraintes d'inégalité, on exprime le critère UTB pour les contraintes d'égalité comme

$$\alpha_{h_i}^{(l)}(\boldsymbol{x}) = \tau_{h_i}^{(l)} \sigma_{h_i}^{(l)}(\boldsymbol{x}) - \left| \mu_{h_i}^{(l)}(\boldsymbol{x}) \right|, \ \forall i \in \{1,\dots,p\}, \tag{4.5}$$

$$\Omega_{\boldsymbol{h}} = \left\{ \boldsymbol{x} \in \Omega, \ \alpha_{h_i}^{(l)}(\boldsymbol{x}) \geq 0, \ \forall i \in \{1,\dots,p\} \right\}, \tag{4.6}$$

où $\hat{\mu}_{h_i}^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}$, $\hat{\sigma}_{h_i}^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}$ et $\tau_{h_i}^{(l)} \in \mathbb{R}^+$ sont respectivement la moyenne, l'écart type et le seuil de doute du GP modélisant la i$^{\text{e}}$ contrainte d'égalité.

On peut définir la même zone de recherche grâce à un critère de faisabilité comprenant une

contrainte d'égalité. Ce critère s'écrit de la manière suivante :

$$\alpha_{h_i}^{(l)}(\boldsymbol{x}) = \max\left\{\left[\mu_{h_i}^{(l)}(\boldsymbol{x}) + \tau_{h_i}^{(l)}\sigma_{h_i}^{(l)}(\boldsymbol{x})\right]^{-},\ \mu_{h_i}^{(l)}(\boldsymbol{x}) - \tau_{h_i}^{(l)}\sigma_{h_i}^{(l)}(\boldsymbol{x})\right\}, \tag{4.7}$$

$$\Omega_{h_i} = \left\{\boldsymbol{x} \in \Omega,\ \alpha_{h_i}^{(l)}(\boldsymbol{x}) = 0,\ \forall i \in \{1,\dots,p\}\right\}, \tag{4.8}$$

où $[s]^{-} = \min(s,0)$. Ainsi il y a trois cas de figures :

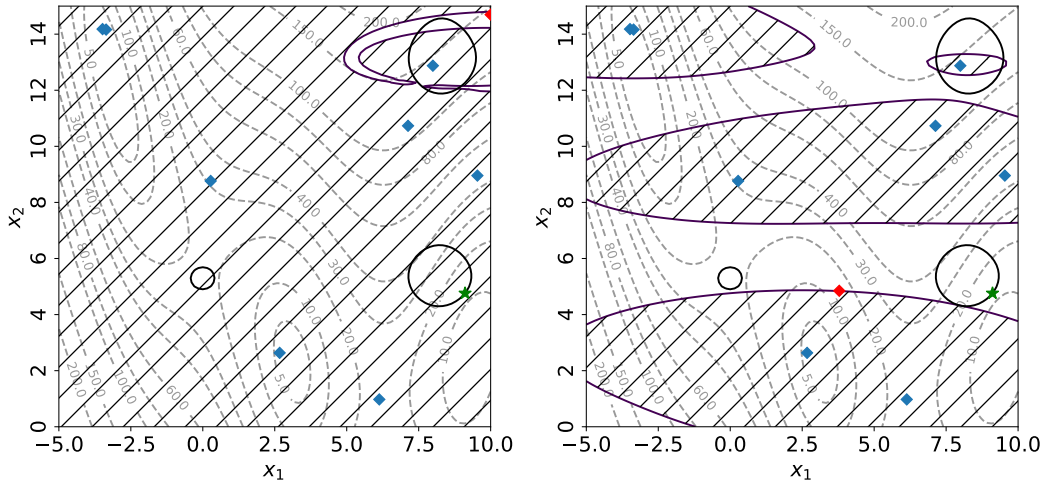
— si $\mu_{h_i}^{(l)}(\boldsymbol{x}) + \tau_{h_i}^{(l)}\sigma_{h_i}^{(l)}(\boldsymbol{x}) \leq 0$, alors $\alpha_{h_i}^{(l)}(\boldsymbol{x}) = \mu_{h_i}^{(l)}(\boldsymbol{x}) + \tau_{h_i}^{(l)}\sigma_{h_i}^{(l)}(\boldsymbol{x})$,

— si $\mu_{h_i}^{(l)}(\boldsymbol{x}) - \tau_{h_i}^{(l)}\sigma_{h_i}^{(l)}(\boldsymbol{x}) \geq 0$, alors $\alpha_{h_i}^{(l)}(\boldsymbol{x}) = \mu_{h_i}^{(l)}(\boldsymbol{x}) - \tau_{h_i}^{(l)}\sigma_{h_i}^{(l)}(\boldsymbol{x})$,

— sinon $\alpha_{h_i}^{(l)}(\boldsymbol{x}) = 0$.

En réalité, on impose $\alpha_{h_i}^{(l)}(\boldsymbol{x}) = 0$ lorsque $\tau_{h_i}^{(l)}\sigma_{h_i}^{(l)}(\boldsymbol{x}) - \left|\mu_{h_i}^{(l)}(\boldsymbol{x})\right| \geq 0$, ce qui est équivalent à (4.5). Cependant, cette définition de $\alpha_{h_i}^{(l)}$ introduit une zone de recherche avec un plateau à valeurs nulles. On ajoute également plus de points non différentiables que (4.5) à cause des opérations de minimisation et maximisation. C'est pourquoi, on utilise la version (4.5) du critère de faisabilité dans le reste du manuscrit.

Dans (4.5), on remarque que le seuil de doute $\boldsymbol{\tau}_{\boldsymbol{h}}^{(l)} \in \mathbb{R}^{+p}$ représente l'inverse d'un seuil de confiance. En effet, plus il est grand, plus les zones de forte violation des contraintes par la moyenne des GP sont explorées. Par exemple, la Figure 4.1 représente la zone de recherche autorisée par le critère de faisabilité UTB pour deux valeurs de seuil de doute pour le problème de Branin modifié [79] où la contrainte d'inégalité est remplacée par une contrainte d'égalité. On voit clairement



(a) $\tau_h = 0.3$ (b) $\tau_h = 3$

FIGURE 4.1 – Représentation du domaine faisable de SEGO-UTB pour le problème Branin modifié avec contrainte d'égalité. La zone hachurée est le domaine non faisable de SEGO-UTB, les lignes noires montrent le domaine faisable réel et les courbes en pointillé sont les courbes de niveau de la fonction objectif. Les carrés bleus représentent le DOE courant tandis que l'étoile verte indique l'optimum global. Le carré rouge est le nouveau point ajouté au DOE.

qu'avec un seuil de doute faible (voir Figure 4.1a), la zone faisable définit par le critère de faisabilité est beaucoup plus petite que celle définie par un seuil de doute plus fort (voir Figure 4.1b).

Pour finir, on peut utiliser le critère de faisabilité UTB dans un processus CBO complet (nommé SEGO-UTB) qui est rappelé dans l'Algorithme 4.1. Avec un seuil de doute élevé, SEGO-UTB favorise l'exploration du domaine de conception. Ceci devrait permettre de résoudre plus facilement les problèmes d'optimisation dont les contraintes sont multi-modales en ne restant pas bloqué

Algorithme 4.1 Le processus SEGO-UTB.

entrée : fonctions objectif et contraintes, DoE initiaux pour les contraintes et l'objectif, un nombre maximum d'itérations max_nb_it, un seuil de doute.

1: **pour** $l = 0$ **à** max_nb_it - 1 **faire**
2: Construire les GP des contraintes et de l'objectif.
3: Trouver le nouveau point à évaluer en résolvant

$$\boldsymbol{x}^{(l+1)} \in \arg\max_{\boldsymbol{x}\in\Omega} \left\{ \alpha_f^{(l)}(\boldsymbol{x}) \quad \text{s.c.} \quad \boldsymbol{x} \in \Omega_{\boldsymbol{h}}^{(l)} \cap \Omega_{\boldsymbol{g}}^{(l)} \right\},$$

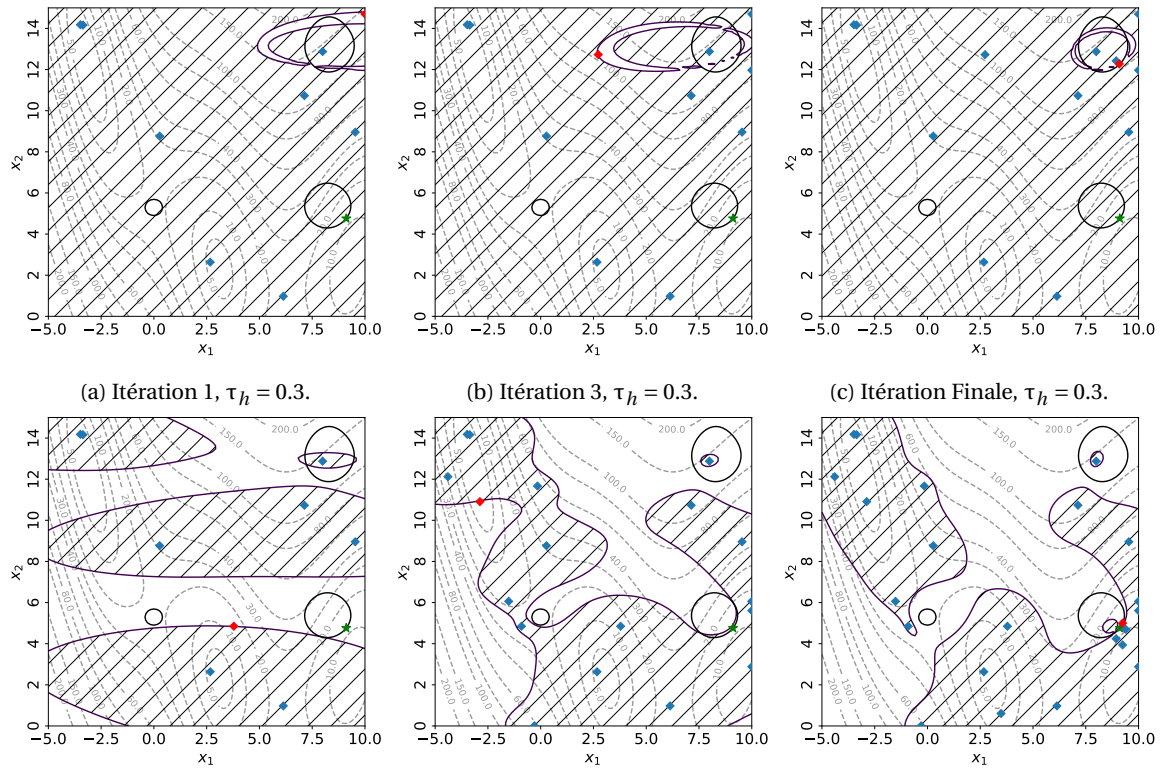
où $\alpha_f^{(l)}$ est une fonction d'acquisition, $\Omega_g^{(l)}$ et $\Omega_h^{(l)}$ sont donnés respectivement par (4.4) et (4.6).
4: Évaluer l'objectif et les contraintes en $\boldsymbol{x}^{(l+1)}$.
5: Mettre à jour les DoE.
6: **fin pour**

sortie : Le meilleur point des DoE

dans une zone faisable qui ne contient pas l'optimum global du problème. Toutefois, en favorisant l'exploration du domaine de conception, la convergence locale du processus d'optimisation est fortement ralentie. Par exemple, la Figure 4.2 représente la première itération, une itération intermédiaire et la dernière itération du processus CBO utilisant le critère de faisabilité UTB sur le problème de Branin modifié avec contrainte pour deux valeurs du seuil de doute ($\tau = 0.3$ et $\tau = 3$). On remarque qu'avec un seuil de doute faible, le processus d'optimisation n'explore pas suffisamment le domaine de conception et reste bloqué aux alentours d'un sous-domaine faisable du domaine de conception. Au contraire, si le seuil de doute est trop élevé, le processus d'optimisation explore beaucoup le domaine et converge vers l'optimum global du problème. Toutefois, le nombre d'appels au problème d'optimisation est beaucoup plus élevé. C'est pourquoi le choix du seuil de doute est capital pour que SEGO-UTB converge rapidement vers l'optimum global.

4.1.3 Conclusion

Dans cette section, on a réintroduit les propriétés désirées pour le processus CBO afin qu'il puisse résoudre des problèmes d'optimisation comprenant un grand nombre de contraintes multimodales d'égalité et d'inégalité et un grand nombre de variables. Ces exigences ont permis d'identifier SEGO-UTB comme la méthode de la littérature la plus prometteuse. Toutefois, le critère de faisabilité sur lequel repose SEGO-UTB ne prend pas en compte les contraintes d'égalité. C'est pourquoi on l'a étendu aux contraintes d'égalité par le biais d'un seuil de doute choisi par l'utilisateur. Ce seuil de doute permet notamment de contrôler le caractère exploratoire de SEGO-UTB. Il est donc capital de bien le maîtriser pour offrir à SEGO-UTB les meilleures performances de convergence. De la même manière, la qualité des GP est primordiale pour le processus CBO. Elle est majoritairement contrôlée par le type de noyau de covariance choisi. Dans la Section suivante, on réalise des tests de convergence en faisant varier ces deux paramètres afin de mieux les appréhender.



(a) Itération 1, $\tau_h = 0.3$. (b) Itération 3, $\tau_h = 0.3$. (c) Itération Finale, $\tau_h = 0.3$.

(d) Itération 1, $\tau_h = 3$. (e) Itération 10, $\tau_h = 3$. (f) Itération Finale, $\tau_h = 3$.

FIGURE 4.2 – Représentation du domaine faisable de SEGO-UTB avec un seuil de doute de 0.3 ou de 3 (pour trois itérations) pour le problème Branin modifié avec contrainte d'égalité. La zone hachurée est le domaine non faisable de SEGO-UTB, la zone grise montre le domaine non faisable réel et les courbes en pointillé sont les courbes de niveau de la fonction objectif. Les carrés bleus représentent le DOE courant tandis que l'étoile verte indique l'optimum global. Le carré rouge est le nouveau point ajouté au DOE.

4.2 Impact du seuil de doute et du noyau de covariance sur SEGO-UTB

Dans cette section, on évalue l'impact des différents paramètres de SEGO-UTB. Plus particulièrement, on cherche à identifier l'impact du seuil de doute et du noyau de covariance des GP modélisant les contraintes sur la rapidité et la robustesse de convergence de l'algorithme. Pour cela, on introduit d'abord les choix d'implémentation réalisés pour SEGO-UTB. Ensuite, on présente les différents choix de paramètres. Par la suite, on teste les différentes mises à jour du seuil de doute ainsi que deux noyaux de covariance sur un ensemble de 4 problèmes. Ces 4 problèmes sont représentatifs des problèmes d'optimisation comportant des contraintes multi-modales d'égalité et/ou d'inégalité. Cette première étude permet d'évaluer la capacité de SEGO-UTB à résoudre de tels problèmes. Pour finir, on évalue la robustesse de convergence des différents paramètres sur un ensemble de 29 problèmes comportant des contraintes d'égalité et/ou d'inégalité. Ceci permet de valider les résultats obtenus précédemment sur les 4 problèmes représentatifs.

4.2.1 Implémentation de SEGO-UTB

Concernant l'implémentation de SEGO-UTB, les choix suivants ont été adoptés. Pour la construction des GP, on travaille avec la toolbox open-source Python SMT [19]. Elle repose sur des GP dont la moyenne est un polynôme de degré choisi par l'utilisateur et dont les coefficients de régression sont des hyper-paramètres des GP. Plus communément, on appelle ces GP, des mo-

dèles de Krigeage universel [61]. De plus, ces hyper-paramètres additionnels peuvent être calculés analytiquement [61] à partir du noyau de covariance choisi. En particulier, on choisit une régression linéaire. Le choix de tous les hyper-paramètres des GP est géré automatiquement à travers les réglages par défaut de la toolbox. De plus, on utilise la fonction d'acquisition WB2S (2.35), introduite dans la Section 2.2, avec $\beta = 100$.

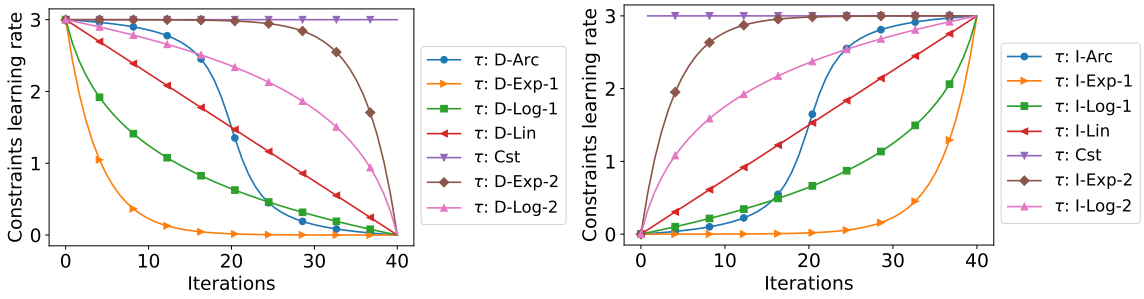
Pour finir, le sous-problème d'optimisation (4.2) est résolu en deux étapes. La première consiste à trouver un point de départ en résolvant (4.2) avec la méthode ISRES [107] de la toolbox Python Nlopt [55]. En effet, ISRES est un algorithme d'optimisation évolutionnaire capable de résoudre des problèmes d'optimisation multi-modaux sous contraintes d'égalité et/ou d'inégalité. Ceci permet l'exploration du domaine de conception sans la restriction à un minimum local comme pourrait le faire un algorithme d'optimisation utilisant l'information des dérivées. Cependant, cette méthode peut nécessiter un grand nombre d'appels à la fonction d'acquisition, et aux critères de faisabilité, ce qui s'avère prohibitif lorsqu'on travaille avec un grand nombre de variables de conception. C'est pourquoi on limite ce nombre d'appels et on utilise la solution obtenue avec ISRES comme point initial d'une optimisation résolue avec un algorithme plus local utilisant l'information des dérivées. Ainsi, la solution est obtenue en résolvant (4.2) avec SNOPT [45] de la toolbox Python PyOptSparse [82] où le point initial est fourni par ISRES. Ce choix offre une solution du problème (4.2) qui a plus de chance d'être global que si on utilisait un algorithme utilisant les dérivées sans l'initialisation adoptée.

4.2.2 Analyse des paramètres de SEGO-UTB

Tout d'abord, on introduit les différentes méthodes d'évolution du seuil de doute liées aux GP modélisant les contraintes. Une première évolution triviale consiste à choisir une valeur de $\tau_{c_i}^{(l)}$ constante tout au long de l'optimisation. Cette mise à jour est notée Cst.. En effet, $\tau_{c_i}^{(l)}$ est utilisé pour contre-balancer $\hat{\mu}_{c_i}^{(l)}$, la moyenne du GP, avec $\hat{\sigma}_{c_i}^{(l)}$, l'écart type du GP. Ce même écart type diminue quand la moyenne du GP devient plus précise. Par exemple, lorsque le nombre de points augmente dans le DoE, la moyenne du GP devient plus représentative de la fonction modélisée. Dans ce cas, un choix naturel pour $\tau_{c_i}^{(l)}$ est 3 pour tout $i = 1, \ldots, m+p$ à l'indice d'itération l. Ainsi, on utilise l'hypothèse que l'intervalle de confiance $[\hat{\mu}_{c_i}^{(l)}(\boldsymbol{x}) - \tau_{c_i}^{(l)}\hat{\sigma}_{c_i}^{(l)}(\boldsymbol{x}), \hat{\mu}_{c_i}^{(l)}(\boldsymbol{x}) + \tau_{c_i}^{(l)}\hat{\sigma}_{c_i}^{(l)}(\boldsymbol{x})]$ à 99% de chance de contenir la valeur de la fonction à modéliser. On rappelle toutefois que cette hypothèse n'est pas vérifiable en pratique (voir Section 2.1.1). En outre, en ne considérant que les contraintes d'inégalité et un seuil de doute fixe $\tau_{c_i}^{(l)} = 3$, on retrouve l'algorithme proposé par LAM et al. [63] (voir Section 2.3.2.3).

Une deuxième stratégie possible considère que la qualité du GP modélisant la contrainte repose fortement sur la taille du DoE. En effet, on s'attend à ce que plus le DoE contienne de points d'échantillonage, plus l'approximation par GP soit fiable. La confiance en la prédiction donnée par la moyenne du GP est donc renforcée lorsque le nombre de points dans le DoE augmente. Concrètement, ceci se traduit par une diminution de la valeur du seuil de doute $\tau_{c_i}^{(l)}$, c.-à-d. une augmentation du seuil de confiance, tant que le nombre de points dans le DoE augmente, c.-à-d. lorsque l devient plus grand. De plus, on considère que le nombre maximum d'itérations max_nb_it dans l'Algorithme 4.1 est suffisamment grand, et donc que la moyenne du GP donne une bonne approximation de la fonction à modéliser. Le seuil de doute peut donc être diminué systématiquement à partir d'une valeur initiale ($\tau_{c_i}^{(0)} = 3$) jusqu'à une valeur nulle à la fin du processus d'optimi-

sation ($\tau_{c_i}^{(\text{max\_nb\_it})} = 0$). Par exemple, la Figure 4.3a montre différentes stratégies décroissantes et monotones applicables à chaque composante du vecteur du seuil de doute $\tau^{(l)}$: arc-tangent (Arc), linéaire (Lin), deux profils logarithmes (Log), and deux profils exponentiels (Exp). Pour chacune de ces stratégies de mise à jour, le seuil de doute est systématiquement réduit quel que soit la fiabilité de la valeur fournie par la moyenne des GP au cours du processus d'optimisation. Comme



(a) Stratégie décroissante (b) Stratégies croissante

FIGURE 4.3 – Évolution du seuil de doute avec le nombre d'itérations (avec max_nb_it = 40).

introduit dans la Section 4.1.2.2, ces stratégies d'évolution correspondent également à une forte exploration du domaine de conception en début d'optimisation lorsque le seuil de doute est élevé et à une forte exploitation des zones faisables connues en fin d'optimisation lorsque le seuil de doute est faible.

Dans les travaux de SASENA et al. [108] et BARTOLI et al. [9], SEGO a montré de bons résultats de convergence. Il s'est notamment avéré à la fois rapide pour atteindre le point optimal et précis dans la valeur de l'optimum. C'est pourquoi on tente d'imiter SEGO au début du processus d'optimisation. On choisit donc de favoriser l'exploitation des données du DoE des contraintes au début de l'optimisation. Comme précisé dans la Section 2.3.1, ceci permet une convergence rapide et précise vers un optimum local du domaine de conception. On autorise ensuite l'exploration du domaine de conception par le processus d'optimisation. Pour cela, on incorpore graduellement l'écart type des GP modélisant les contraintes au cours du processus d'optimisation. Ceci peut se traduire par une mise à jour du seuil de doute $\boldsymbol{\tau}^{(l)}$ croissante avec le nombre d'itérations, c.-à-d. croissante avec le nombre de points dans le DoE. Par exemple, à la première itération, le seuil de doute peut être nul et ensuite systématiquement augmenté pour atteindre une valeur maximale. Classiquement et comme pour les stratégies décroissantes, la valeur maximale du seuil de doute au cours de l'optimisation est $\tau_{c_i}^{(\text{max\_nb\_it})} = 3$. La Figure 4.3b représente différentes stratégies croissantes qui peuvent être utilisées et qui sont testées dans la suite de ce manuscrit.

En ce qui concerne le noyau de covariance, on s'attarde sur deux d'entre eux qui sont particulièrement employés en pratique. D'une part, on utilise le noyau de covariance gaussien (voir Tableau 2.1) qui est le plus classique. D'autre part, on recourt au noyau de covariance Matérn 5/2. Ce dernier est le noyau de covariance par défaut dans la plupart des algorithmes BO et CBO (comme ALBO, PESC, EFI, etc.).

Pour conclure sur les différents paramètres pris en compte dans cette étude, on considère 3 types de mise à jour du seuil de doute : constant, croissant et décroissant. Plus particulièrement, 6 variantes de mise à jour croissantes et décroissantes sont prises en compte et représentées dans la Figure 4.3. Pour chacune de ces 13 variantes, on analyse deux noyaux de covariance possibles : gaussien et Matérn 5/2. On trouve leurs expressions dans le Tableau 2.1. Au final, on confronte 26

versions de l'algorithme SEGO-UTB sur différents cas tests.

4.2.3 Étude sur quatre problèmes d'optimisation représentatifs

On évalue tout d'abord l'impact des paramètres de SEGO-UTB sur un ensemble de 4 tests représentatifs des problèmes d'optimisation sous contraintes multi-modale mixtes, c.-à-d. qui comportent des contraintes d'égalité et/ou d'inégalité. Afin de mener à bien ces tests, on va d'abord présenter les 4 problèmes considérés et les raisons de leur représentativité. Ensuite, on introduit les détails d'implémentation relatifs aux tests et à SEGO-UTB. Pour finir, on présente les résultats de ces tests grâce à des courbes de convergence.

4.2.3.1 Quatre problèmes représentatifs

On cherche d'abord à tester les différentes versions de SEGO-UTB introduites dans la Section 4.2.2 sur des problèmes représentatifs. Ces problèmes doivent donc comporter des contraintes multi-modales d'égalité et/ou d'inégalité ainsi qu'une fonction objective qui peut être multi-modale. En effet, SEGO-UTB s'adresse principalement à la résolution de problèmes à contraintes multi-modales. On a donc sélectionné dans la littérature quatre problèmes qui sont particulièrement représentatifs. Parmi les quatre problèmes, trois proviennent du travail de PICHENY et al. [84] et le dernier est issu de PARR et al. [79].

— Linear-Hartman-Ackley problem (LAH) : ce problème comporte quatre variables de conception, une fonction objectif linéaire, une contrainte d'égalité et une contrainte d'inégalité respectivement données par la fonction de Hartmann et la fonction de Ackley. Avec ce problème, on s'intéresse plus particulièrement à la prise en compte des contraintes mixtes fortement multi-modales. Mathématiquement, ce problème s'écrit

$$\min_{\boldsymbol{x}\in\Omega_2} f_1(\boldsymbol{x}) \ \text{ s.c. } \ c_1(\boldsymbol{x}) \leq 0, \ c_2(\boldsymbol{x}) = 0, \tag{4.9}$$

où $\Omega_2 = [0,1]^4$ et

$$f_1(\boldsymbol{x}) = \sum_{i=1}^{4} x_i, \tag{4.10}$$

$$c_1(\boldsymbol{x}) = 20\exp\left(-0.2\sqrt{\frac{1}{4}\sum_{i=1}^{4}(3x_i-1)^2}\right) + \exp\left(\frac{1}{4}\sum_{i=1}^{4}\cos\left(2\pi(3x_i-1)\right)\right), \tag{4.11}$$

$$c_2(\boldsymbol{x}) = \frac{1}{0.8387}\left[-1.1 + \sum_{i=1}^{4} \mathrm{C}_i \exp\left(-\sum_{j=1}^{4} a_{ij}(x_j - p_{ij})^2\right)\right], \tag{4.12}$$

avec

$$\boldsymbol{a} = \begin{bmatrix} 10.00 & 0.05 & 3.00 & 17.00 \\ 3.00 & 10.00 & 3.50 & 8.00 \\ 17.00 & 17.00 & 1.70 & 0.05 \\ 3.50 & 0.10 & 10.00 & 10.00 \end{bmatrix}, \ \boldsymbol{p} = \begin{bmatrix} 0.131 & 0.232 & 0.234 & 0.404 \\ 0.169 & 0.413 & 0.145 & 0.882 \\ 0.556 & 0.830 & 0.352 & 0.873 \\ 0.012 & 0.373 & 0.288 & 0.574 \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} 1.0 & 1.2 & 3.0 & 3.2 \end{bmatrix}.$$

— Goldstein-Price-Sinusoidal-Branin-Parr problem (GSBP) : ce problème comprend deux variables de conception. La fonction objective est une version modifiée de la fonction de Goldstein-Price. Le problème comporte une contrainte d'inégalité donnée par une fonction sinusoïdale et deux contraintes d'égalité qui s'expriment grâce la fonction de Branin et une fonction provenant du travail de PARR et al. [79]. De plus, il se compose d'une fonction objectif fortement multi-modale. La Figure 4.4b fournit une représentation de ce problème qui dispose de zones faisables disjointes pour chaque contrainte d'égalité. De plus, seuls deux points du domaine de conception sont faisables. Pour ces deux raisons, le problème GSBP est particulièrement difficile à résoudre. Concrètement, le problème GSBP s'écrit comme suit

$$\min_{\boldsymbol{x}\in\Omega_1} f_2(\boldsymbol{x}) \;\text{ s.c. }\; c_3(\boldsymbol{x}) \geq 0,\; c_4(\boldsymbol{x}) = 0,\; c_5(\boldsymbol{x}) = 0, \tag{4.13}$$

où $\Omega_1 = [0,1]^2$ et

$$f_2(\boldsymbol{x}) = \frac{\log\left[\left(1+a(4x_1+4x_2-3)^2\right)\left(30+b(8x_1-12x_2+2)^2\right)\right]-8.69}{2.43}, \tag{4.14}$$

$$c_3(\boldsymbol{x}) = 0.5\sin\left(2\pi\left(x_1^2-2x_2\right)\right)+x_1+2x_2-1.5, \tag{4.15}$$

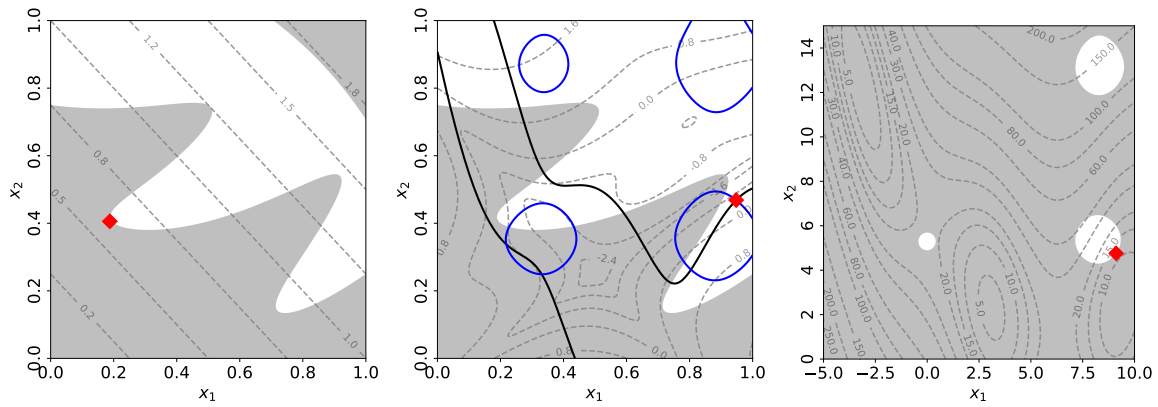
$$c_4(\boldsymbol{x}) = -x_1^2-x_2^2+1.5, \tag{4.16}$$

$$c_5(\boldsymbol{x}) = 15-\left(15x_2-\frac{5}{4\pi^2}(15x_1-5)^2+\frac{5}{\pi}(15x_1-5)-6\right)^2-10\left(1-\frac{1}{8\pi}\right)\cos(15x_1-5), \tag{4.17}$$

avec

$$a = 75-56(x_1+x_2)+3(4x_1-2)^2+6(4x_1-2)(4x_2-2)+3(4x_2-2)^2, \tag{4.18}$$

$$b = -14-128x_1+12(4x_1-2)^2+192x_2-36(4x_1-2)(4x_2-2)+27(4x_2-2)^2. \tag{4.19}$$



(a) LSQ. (b) GSBP. (c) MB.

FIGURE 4.4 – Représentation des problèmes GSBP, LSQ et MB. Les lignes en pointillé sont les courbes de niveaux de la fonction objectif, la zone grise est le domaine non faisable pour les contraintes d'inégalité, les lignes pleines de même couleur représentent le domaine faisable d'une contrainte d'égalité, le carré rouge est l'optimum global du problème d'optimisation.

— Linear-Sinusoidal-Quadratic problem (LSQ) : ce problème mêle deux variables de conception, une fonction objective linéaire et deux contraintes d'inégalité. La première contrainte d'inégalité est sinusoïdale et la seconde est quadratique. La Figure 4.4a représen-

tant le problème LSQ montre que les contraintes sont non-linéaires et produisent un domaine faisable non-convexe. Il existe ici trois minima locaux dont un global. Ceci rend le problème pertinent à utiliser dans notre étude. Pour finir, on définit le problème LSQ avec les expressions suivantes

$$\min_{\boldsymbol{x}\in\Omega_1} f_1(\boldsymbol{x}) \text{ s.c. } c_3(\boldsymbol{x}) \geq 0,\ c_4(\boldsymbol{x}) \geq 0, \tag{4.20}$$

où $\Omega_1 = [0,1]^2$, f_1, c_3 et c_4 sont respectivement donnés par (4.10), (4.15) et (4.16).

— MB : ce problème [79], que l'on ajoute aux trois problèmes précédemment tirés de PICHENY et al. [84], compte deux variables de conception, une fonction objectif multi-modale et une contrainte multi-modale. La Figure 4.4c représente ce problème qui comporte trois zones faisables distinctes. Pour finir, on écrit le problème MB grâce aux expressions suivantes

$$\min_{\boldsymbol{x}\in\Omega_3} f_3(\boldsymbol{x}) \text{ s.c. } c_7(\boldsymbol{x}) \leq 0, \tag{4.21}$$

où $\Omega_3 = [-5,10] \times [0,15]$ et

$$f_3(\boldsymbol{x}) = \left[\left(x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6\right)^2 + \left(10 - \frac{10}{8\pi}\right)\cos(x_1) + 1\right] + \frac{5x_1+25}{15} \tag{4.22}$$

$$c_7(\boldsymbol{x}) = 6 - \left(4 - 2.1\overline{x}_1^2 + \frac{\overline{x}_1^4}{3}\right)\overline{x}_1^2 - \overline{x}_1\overline{x}_2 - \left(4\overline{x}_2^2 - 4\right)\overline{x}_2^2 - 3\sin\left(6(1-\overline{x}_1)\right) - 3\sin\left(6(1-\overline{x}_2)\right) \tag{4.23}$$

avec $\overline{x}_1 = {}^{(x_1-2.5)}/_{7.5}$ et $\overline{x}_2 = {}^{(x_2-7.5)}/_{7.5}$.

Pour résumer, on a présenté les quatre problèmes représentatifs qui sont utilisés dans le manuscrit pour évaluer les performances de SEGO-UTB. Le Tableau 4.1 rappelle les principales caractéristiques de ces problèmes. On a également expliqué en quoi ces problèmes sont représentatifs

TABLEAU 4.1 – Propriétés des 4 problèmes représentatifs pris en compte dans cette étude.

Nom du Problème	Nb. de Variables	Nb. de Eq. Cst.	Nb. de Ieq. Cst.	Valeur de Ref.
LAH	4	1	1	$5{,}176 \cdot 10^{-2}$
LSQ	2	0	2	0,600
GSBP	2	2	1	$-0{,}525\,2$
MB	2	0	1	12,00

des problèmes à contraintes multi-modales mixtes.

4.2.3.2 Construction des courbes de convergence

Dans cette section, on introduit la méthode de comparaison des différentes variantes de SEGO-UTB. Pour cela, on construit des courbes de convergence pour chacun des quatre problèmes. Elles sont construites de la manière suivante. Tout d'abord, on réalise 100 optimisations indépendantes pour chaque variante en utilisant 100 DoE initiaux générés par LHS. Pour chacune des optimisations, toutes les versions sont initialisées avec les mêmes DoE initiaux. Ceci permet d'avoir une base commune et de limiter l'influence du choix du DoE initial dans le processus d'optimisation. De plus, les DoE initiaux sont créés avec $n_{start} = \max(d+1,5)$ points d'échantillonage où d est le nombre de variables de conception du problème concerné. Le nombre maximum d'itérations est

imposé à $max\_nb\_it = 40d - n_{start}$ ce qui correspond à un budget total de $40d$ évaluations. Pour les contraintes, on autorise une violation absolue de chacune d'entre elles de $\epsilon_c = 10^{-4}$.

Toutes les variantes sont ensuite comparées en affichant la moyenne et l'écart type de la meilleure valeur sur les 100 optimisations pour un nombre d'évaluations croissant. La meilleure valeur à une itération donnée l du processus d'optimisation se définit comme la meilleure valeur faisable dans le DoE, s'il contient au moins un point faisable. Sinon une valeur de pénalisation remplace cette valeur permettant d'obtenir des courbes de convergence strictement décroissantes. Ces pénalisations sont données dans le Tableau 4.2. Pour des raisons de lisibilité des

TABLEAU 4.2 – Pénalisations et facteurs de mise à l'échelle de l'écart type pour les problèmes LAH, LSQ, MB et GSBP.

Problème	MB	LSQ	GSBP	LAH
Pénalisation	150	2	3	3
Facteur	4	2	4	4

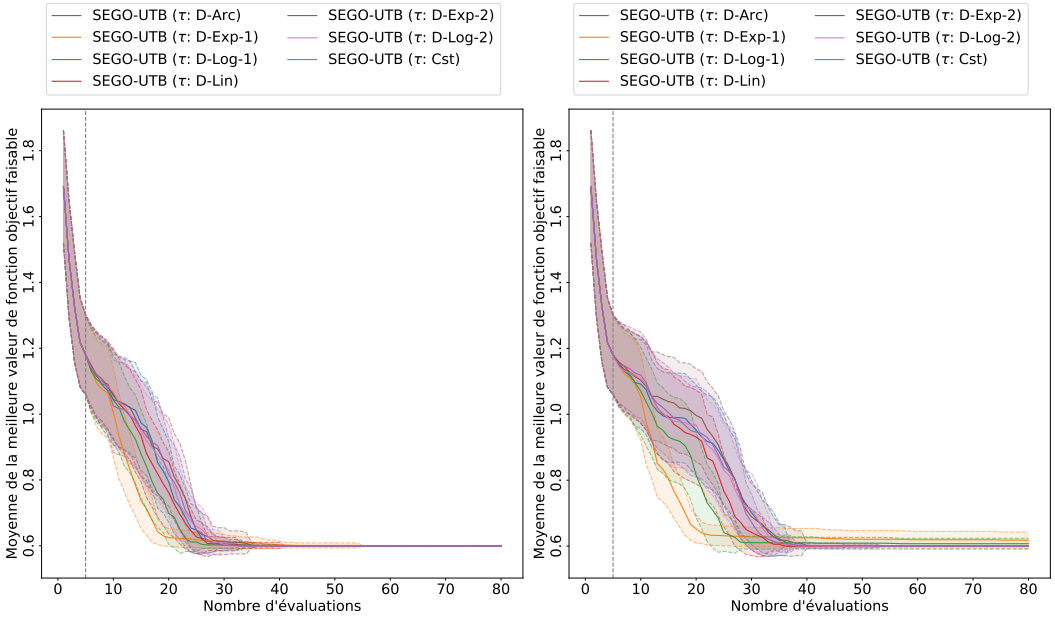
courbes produites, on affiche l'écart type réduit par un facteur d'échelle différent pour chaque problème. Ces facteurs sont donnés par le Tableau 4.2.

4.2.3.3 Analyse des résultats

Dans cette section, on analyse les résultats obtenus aux cours des tests présentés dans la Section 4.2.3.2. Pour chacun des quatre problèmes représentatifs, on présente cinq figures comprenant différentes variantes de SEGO-UTB. On crée d'abord une figure pour chaque noyau de covariance pris en compte avec les seuils de doute croissants auxquels on ajoute le seuil de doute constant. Les mêmes figures sont produites pour les seuils de doute décroissants. Ensuite, on sélectionne la meilleure version de SEGO-UTB sur chacune de ces figures, c.-à-d. qui converge le plus rapidement possible et vers la valeur de fonction objectif la plus faible. Pour finir, on compare les quatre versions choisies entre elles pour désigner la plus prometteuse. A la fin de ce processus, on choisit la meilleure version de SEGO-UTB à utiliser au vu des résultats obtenus sur les quatre problèmes.

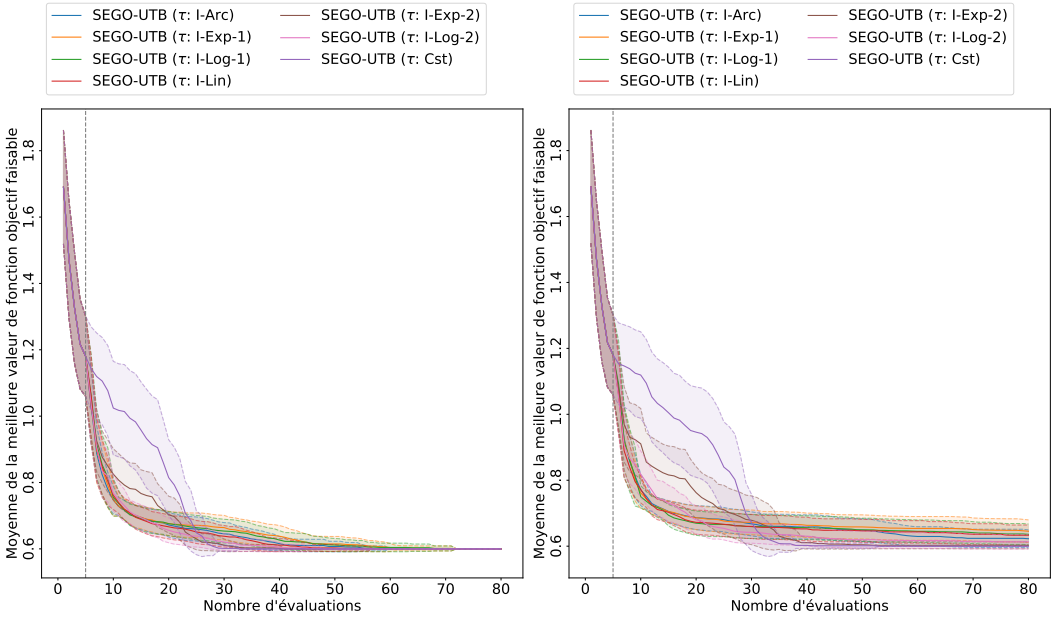
La Figure 4.5 présente les résultats des différentes versions de SEGO-UTB pour les noyaux de covariance gaussien et Matérn 5/2. On remarque tout d'abord que l'ensemble des versions de SEGO-UTB converge vers la solution du problème LSQ excepté pour les seuils de doute croissants avec noyau Matérn 5/2. De plus, on remarque que SEGO-UTB (τ : D-Log-1) à noyau gaussien converge le plus rapidement vers la solution du problème LSQ parmi les variantes à seuil de doute décroissant et à noyau gaussien. De la même façon, SEGO-UTB (τ : D-Lin) Matérn 5/2 converge le plus rapidement vers la solution du problème LSQ parmi les variantes à seuil de doute décroissant et à noyau Matérn 5/2. En ce qui concerne les seuils de doute croissants, on voit que SEGO-UTB (τ : Cst) surpasse clairement les autres versions quel que soit le noyau choisi. Pour finir avec le problème LSQ, la Figure 4.6 compare les quatre versions identifiées précédemment. On remarque que SEGO-UTB (τ : D-Log-1) à noyau gaussien converge le plus rapidement vers la solution optimale du problème LSQ avec un écart type nul. On note ici que le noyau gaussien et une mise à jour du seuil de doute décroissante semblent les plus adaptés à ce problème.

Les résultats du problème MB pour les différentes versions de SEGO-UTB sont présentés par la



(a) Seuil de doute décroissants et noyau gaussien.

(b) Seuil de doute décroissants et noyau Matérn 5/2.



(c) Seuil de doute croissants et noyau de covariance gaussien.

(d) Seuil de doute croissants et noyau Matérn 5/2.

FIGURE 4.5 – Courbes de convergence de l'optimisation du problème LSQ par SEGO-UTB pour deux noyaux de covariance et différentes évolutions du seuil de doute avec une violation des contraintes de $\epsilon_c = 10^{-4}$.

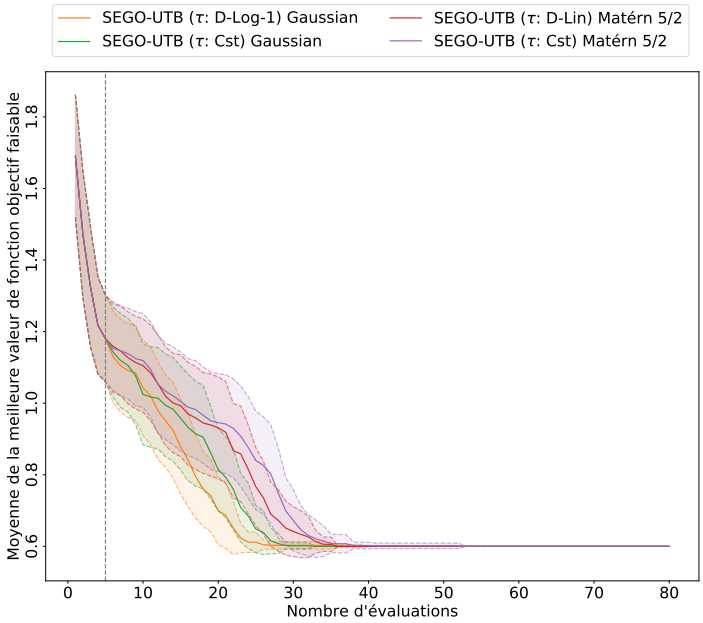
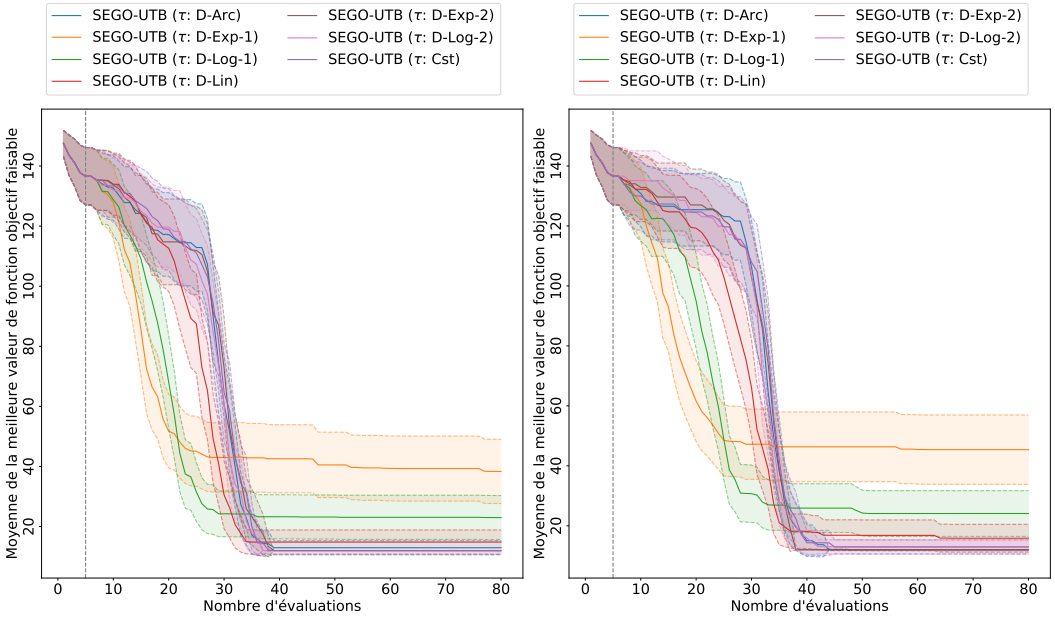


FIGURE 4.6 – Courbes de convergence de l'optimisation du problème LSQ par SEGO-UTB pour les meilleures évolutions du seuil de doute et pour deux noyaux de covariance avec une violation des contraintes de $\epsilon_c = 10^{-4}$.

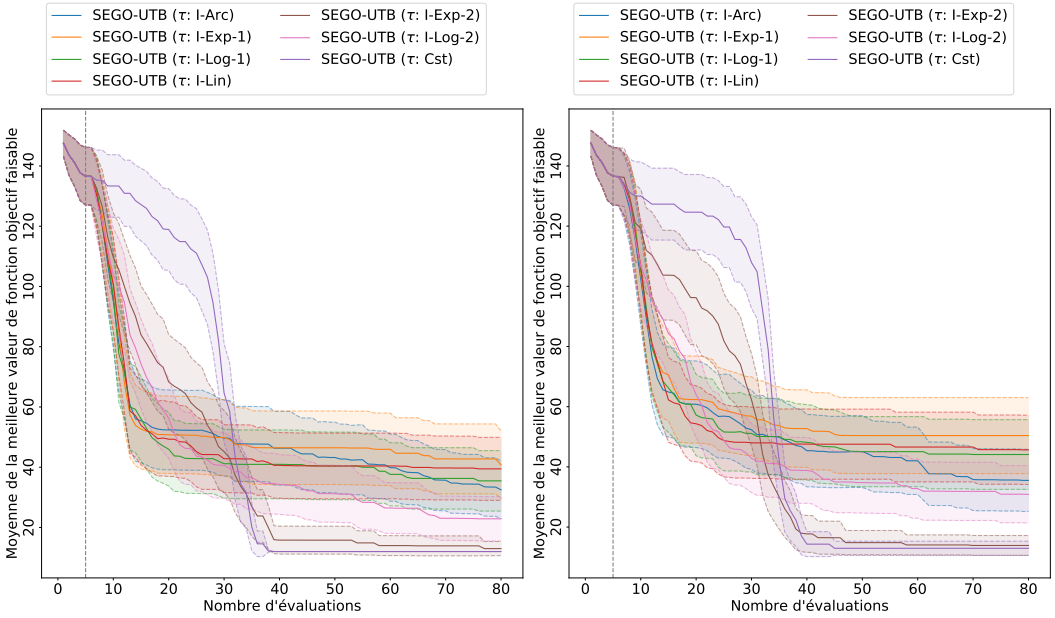
Figure 4.7. Les performances sont ici plus mitigées que pour le problème LSQ. D'une part, on remarque que l'ensemble des versions de SEGO-UTB ne converge pas vers l'optimum du problème MB. Pour les mises à jour du seuil de doute décroissantes, les versions SEGO-UTB (τ : D-Exp-1) et (τ : D-Log-1) offrent les moins bonnes performances quel que soit le noyau de covariance choisi. Ces deux versions sont celles qui autorisent le moins l'exploration du domaine de conception. On suppose donc qu'une exploration forte du domaine de conception est nécessaire à la résolution du problème MB. Les variantes SEGO-UTB (τ : D-Log-2) à noyau gaussien et (τ : D-Exp-2) à noyau Matérn 5/2 donnent la convergence la plus rapide vers l'optimum global. D'autre part, la majorité des versions comportant une mise à jour du seuil de doute croissant ne convergent pas vers l'optimum global. Seuls SEGO-UTB (τ : Cst) à noyau gaussien et à noyau Matérn 5/2 semblent offrir une convergence systématique vers l'optimum global avec un écart type quasi nul après 50 évaluations. Ceci conforte la nécessité d'une exploration importante du domaine de conception dès le début du processus d'optimisation du problème MB pour une bonne convergence. On compare ensuite les quatre variantes sélectionnées sur la Figure 4.8. D'une part, on remarque que SEGO-UTB (τ : Cst) Matérn 5/2 est la seule variante à ne pas converger systématiquement vers l'optimum global. En effet, l'écart type est non nul après 80 évaluations du problème MB. D'autre part, les trois autres méthodes montrent un comportement similaire et convergent toutes vers l'optimum global en moins de 40 évaluations. On remarque néanmoins que SEGO-UTB (τ : D-Log-2) avec un noyau gaussien converge légèrement plus rapidement que les trois autres variantes. Pour conclure sur le problème MB, on a remarqué qu'il nécessite une mise à jour du seuil de doute qui permet une exploration dès le début du processus d'optimisation afin de converger systématiquement vers l'optimum global. Néanmoins, SEGO-UTB (τ : Cst) à noyau Matérn 5/2 ne converge pas systématiquement ce qui indique qu'une phase d'exploitation des données est nécessaire à la résolution du problème.

Les résultats du problème GSBP sont commentés dans la suite à partir de la Figure 4.9. D'une



(a) Seuil de doute décroissants et noyau gaussien.

(b) Seuil de doute décroissants et noyau Matérn 5/2.



(c) Seuil de doute croissants et noyau de covariance gaussien.

(d) Seuil de doute croissants et noyau Matérn 5/2.

FIGURE 4.7 – Courbes de convergence de l'optimisation du problème MB par SEGO-UTB pour deux noyaux de covariance et différentes évolutions du seuil de doute avec une violation des contraintes de $\epsilon_c = 10^{-4}$.

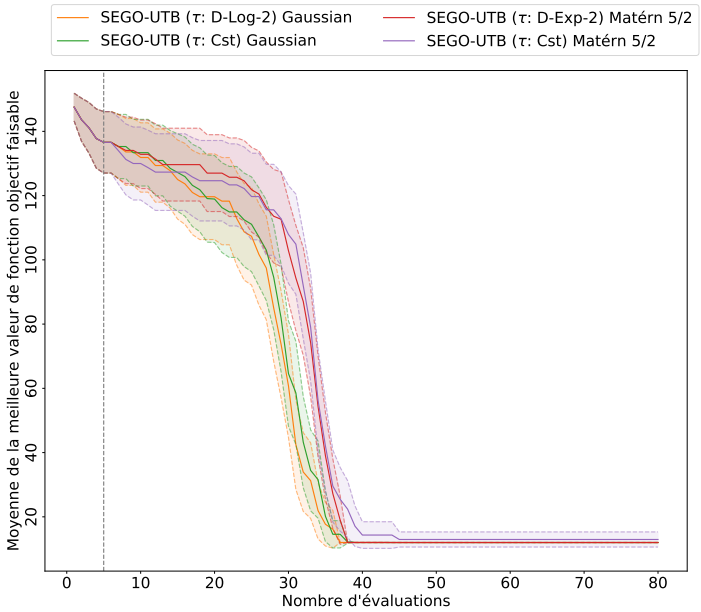
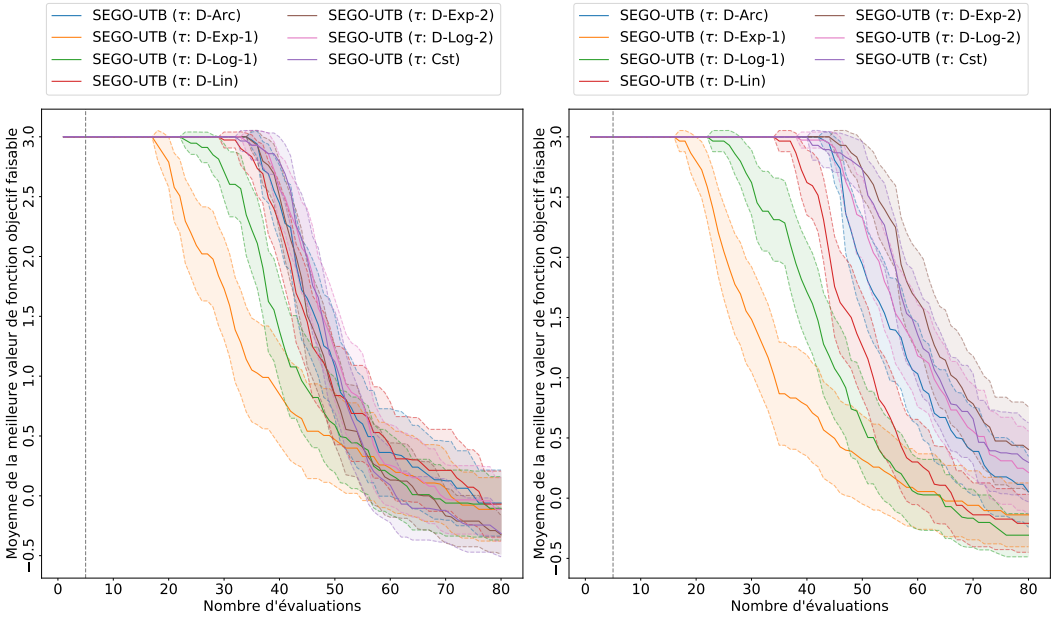


FIGURE 4.8 – Courbes de convergence de l'optimisation du problème MB par SEGO-UTB pour les meilleures évolutions du seuil de doute et pour deux noyaux de covariance avec une violation des contraintes de $\epsilon_c = 10^{-4}$.

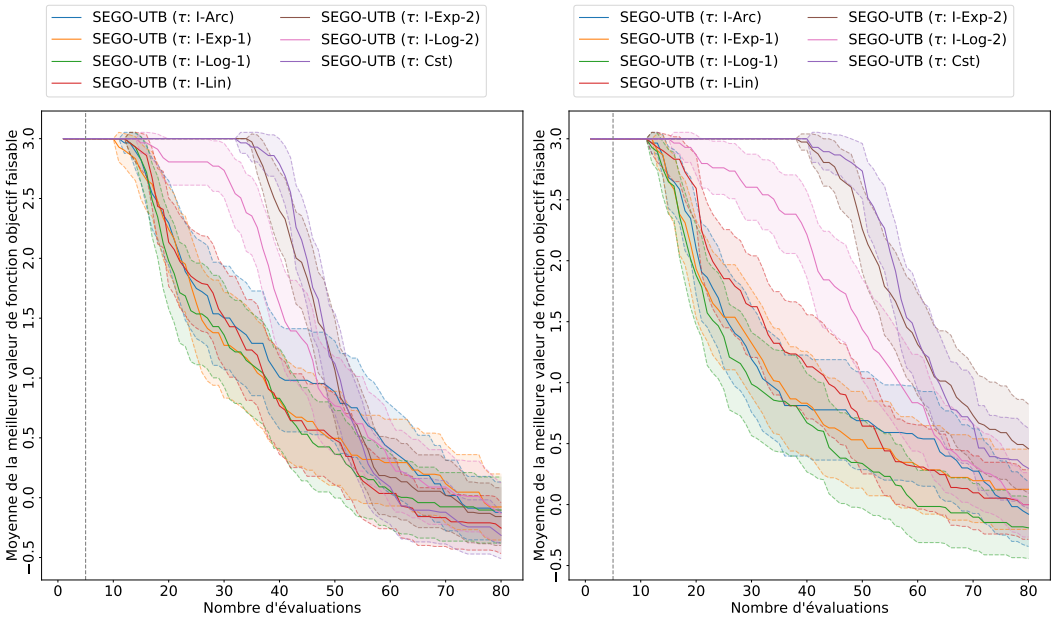
part, on constate qu'aucune des variantes de SEGO-UTB n'est capable de converger systématiquement vers l'optimum global car l'écart type n'est jamais nul. Néanmoins, on peut sélectionner les versions qui convergent vers la valeur de fonction objectif la plus faible. Pour les versions à noyau gaussien, la mise à jour décroissante qui converge le mieux est SEGO-UTB (τ : Cst) bien que (τ : D-Exp-2) soit également compétitif. Pour les mises à jour croissantes, SEGO-UTB (τ : I-Lin) découvre les valeurs du problème GSBP faisables les plus faibles. SEGO-UTB (τ : Cst) converge également vers la même valeur de fonction objectif mais nécessite un plus grand nombre d'évaluations pour trouver des valeurs faisables du problème. En effet, le plateau à la valeur de pénalisation indique qu'aucun point faisable n'a été découvert pour les 100 optimisations réalisées. Les versions à noyau Matèrn 5/2 convergent, quant à elles, vers des valeurs de fonctions objectif différentes. On peut notamment remarquer que plus la phase d'exploration est importante, plus la meilleure valeur obtenue est forte. Ce phénomène se produit lorsque certaines optimisations ne trouvent pas de point faisable. La meilleure valeur est donc fixée à la valeur de pénalisation, donnée dans le Tableau 4.2, ce qui augmente la moyenne. On peut toutefois identifier les deux méthodes les plus prometteuses. Ici, SEGO-UTB (τ : I-Log-1) et (τ : D-Log-1) convergent le plus rapidement vers la meilleure valeur de fonction objectif. Ensuite, on compare les quatre versions de SEGO-UTB qui offrent les meilleurs résultats à partir de la Figure 4.10. Les quatre versions de SEGO-UTB convergent vers la même valeur de fonction objectif avec le même écart type. Toutefois, on constate que SEGO-UTB (τ : I-lin) converge vers la valeur la plus faible et le plus rapidement. Au vu des résultats obtenus, on peut favoriser le noyau gaussien ainsi que les mises à jour croissantes pour résoudre le problème GSBP.

On finit par étudier les résultats des tests réalisés sur le problème LAH et présentés dans la Figure 4.11. Pour des raisons de lisibilité, on ne montre que les 40 premières évaluations réalisées. On remarque ici que l'ensemble des versions de SEGO-UTB converge vers le minimum global du problème LAH. On peut toutefois sélectionner les meilleures versions pour chaque groupe de



(a) Seuil de doute décroissants et noyau gaussien.

(b) Seuil de doute décroissants et noyau Matérn 5/2.



(c) Seuil de doute croissants et noyau de covariance gaussien.

(d) Seuil de doute croissants et noyau Matérn 5/2.

FIGURE 4.9 – Courbes de convergence de l'optimisation du problème GBSP par SEGO-UTB pour deux noyaux de covariance et différentes évolutions du seuil de doute avec une violation des contraintes de $\epsilon_c = 10^{-4}$.

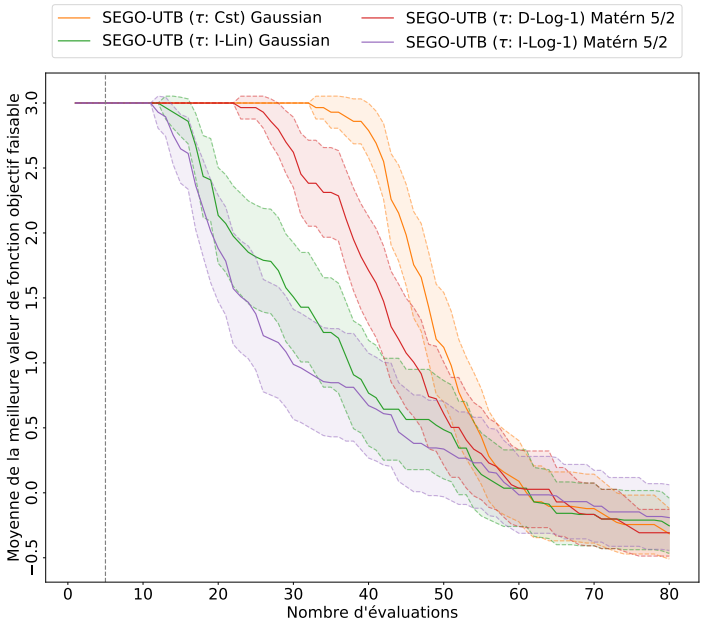


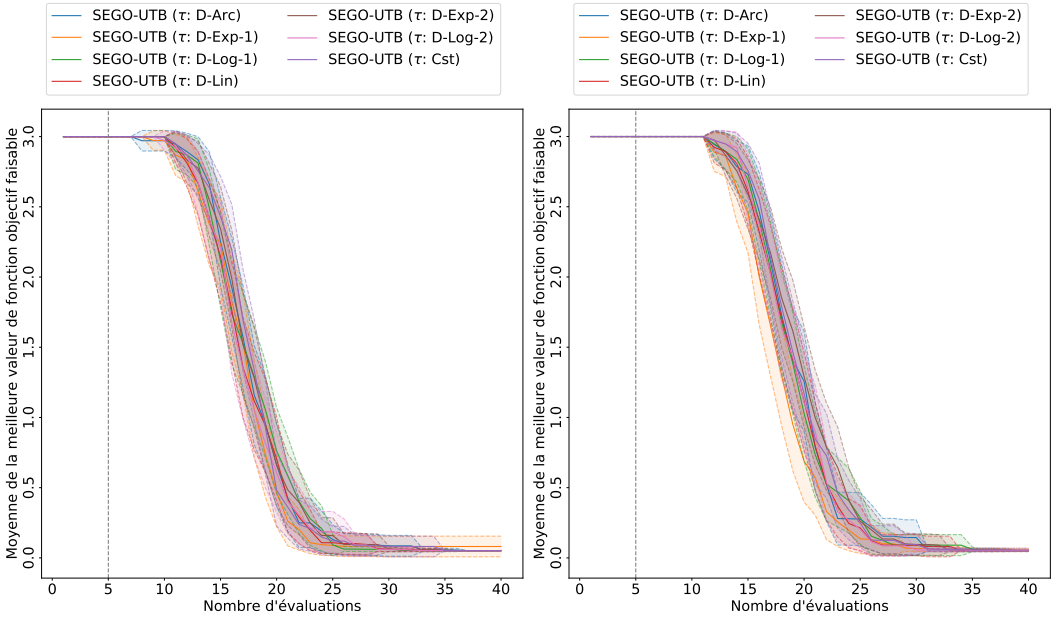
FIGURE 4.10 – Courbes de convergence de l'optimisation du problème GBSP par SEGO-UTB pour les meilleures évolutions du seuil de doute et pour deux noyaux de covariance avec une violation des contraintes de $\epsilon_c = 10^{-4}$.

courbes de convergence de la Figure 4.11. Pour les mises à jours décroissantes à noyau gaussien, la méthode SEGO-UTB (τ : D-Log-1) est la première à converger systématiquement vers l'optimum global, c.-à-d. que l'écart type s'annule le plus rapidement. De la même façon, SEGO-UTB (τ : D-Exp-1) à noyau Matérn 5/2, (τ : I-Log-1) à noyau gaussien et (τ : I-Lin) à noyau Matérn 5/2 convergent systématiquement le plus rapidement. En comparant ces quatre variantes avec la Figure 4.12, on constate que SEGO-UTB (τ : D-Log-1) converge légèrement plus rapidement que les trois autres. Pour conclure sur les tests réalisés sur le problème LAH, on remarque que l'influence du noyau de covariance et de la mise à jour du seuil de doute est presque nulle. Ainsi la présence d'une phase d'exploration dans le processus d'optimisation ne semble pas être préjudiciable à la bonne résolution du problème LAH.

Pour conclure sur l'évaluation de l'impact du seuil de doute et du noyau de covariance sur les performances de convergence de SEGO-UTB, on remarque que le noyau de covariance gaussien donne de meilleurs résultats que le noyau Matérn 5/2. Ces versions convergent notamment plus rapidement et plus systématiquement vers un optimum global. L'impact de la mise à jour du seuil de doute est, quant à lui, plus difficile à quantifier. En effet, cette mise à jour peut n'avoir aucun effet comme avec les problèmes GSBP et LAH comme beaucoup d'effets avec les problèmes MB et LSQ. Plus particulièrement, aucune méthode croissante n'est capable de résoudre le problème MB systématiquement. Néanmoins, on remarque qu'une mise à jour du type décroissante obtient les meilleures performances de résolution des problèmes GSBP, LSQ et MB. Afin de valider et étayer ces conclusions, on évaluera l'impact du seuil de doute et du noyau de covariance sur un ensemble de 29 problèmes comportant des contraintes d'égalité et/ou d'inégalité dans la section suivante.

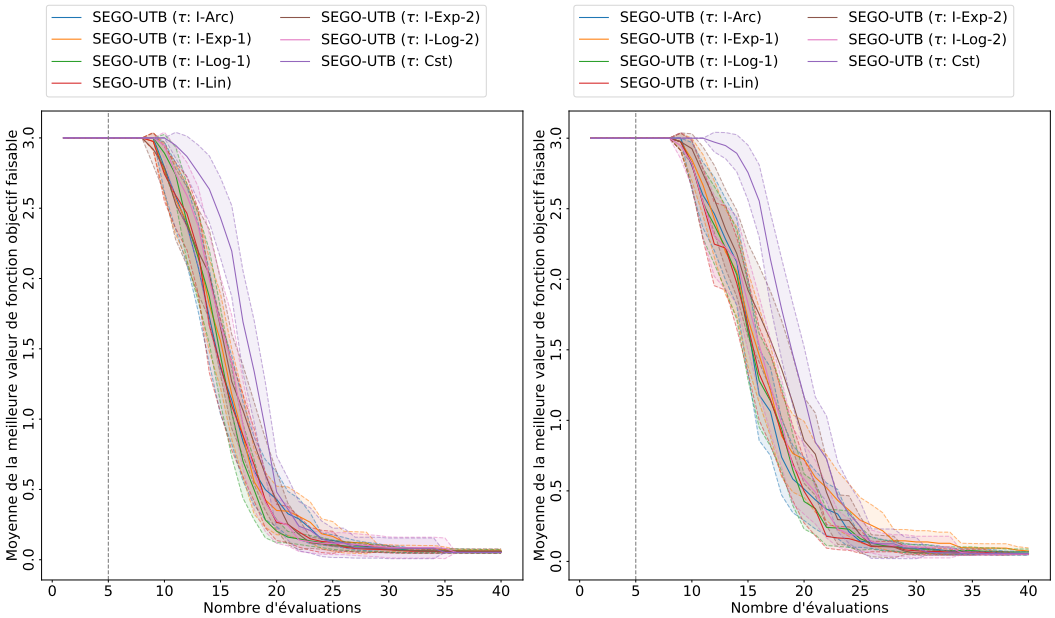
4.2.4 Étude sur un ensemble de 29 problèmes

Dans cette section, on cherche à vérifier les résultats obtenus dans la Section 4.2.3 ainsi que la robustesse de SEGO-UTB. Pour cela, on teste les différentes versions de SEGO-UTB sur un en-



(a) Seuil de doute décroissants et noyau gaussien.

(b) Seuil de doute décroissants et noyau Matérn 5/2.



(c) Seuil de doute croissants et noyau de covariance gaussien.

(d) Seuil de doute croissants et noyau Matérn 5/2.

FIGURE 4.11 – Courbes de convergence de l'optimisation du problème LAH par SEGO-UTB pour deux noyaux de covariance et différentes évolutions du seuil de doute avec une violation des contraintes de $\epsilon_c = 10^{-4}$.

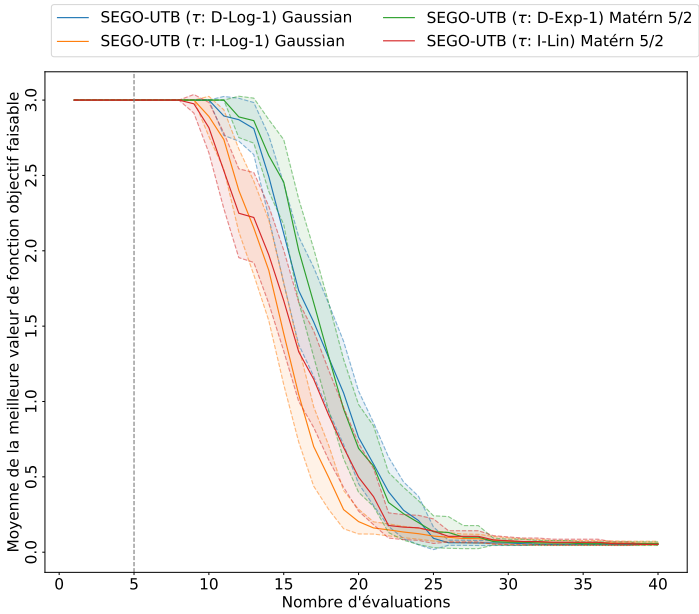


FIGURE 4.12 – Courbes de convergence de l'optimisation du problème LAH par SEGO-UTB pour les meilleures évolutions du seuil de doute et pour deux noyaux de covariance avec une violation des contraintes de $\epsilon_c = 10^{-4}$.

semble de 29 problèmes comportant des contraintes d'égalité et d'inégalité. On présente d'abord les caractéristiques de cet ensemble de problèmes. Ensuite, on introduit la méthode de comparaison reposant sur l'utilisation de profils de données, ou data profiles (DP). Enfin, on étudie les DP produits pour découvrir la version de SEGO-UTB la plus robuste.

4.2.4.1 L'ensemble de problèmes de références

L'ensemble des problèmes de référence sur lequel est évalué l'impact du seuil de doute et du noyau de convergence se compose de 29 problèmes d'optimisation provenant de [69, 79, 84, 100]. Ce sont des problèmes d'optimisation sous contraintes mixtes (jusqu'à 38 contraintes) dépendant de 2 à 10 variables de conception. Le nombre de contraintes, leur type, le nombre de variables de conception ainsi que la valeur de l'optimum considérée dans cette étude sont donnés dans le Tableau 4.3. Pour le détail des expressions des problèmes, on renvoie le lecteur à l'Annexe A.3.

Afin de prendre en compte le caractère stochastique des méthodes d'optimisation Bayésienne, on crée plusieurs instances pour chaque problème. En réalité, les résultats obtenus pour un problème donné dépendent fortement du choix du DoE initial. Ainsi, on lie chaque instance de problèmes au choix du DoE initial. On génère donc 10 DoE initiaux de $n_{start} = \max(d+1, 5)$ points d'échantillonnage avec la méthode LHS pour chaque problème de l'ensemble introduit dans la Section 4.2.4.1. On obtient donc 290 instances de problèmes pour chaque version de SEGO-UTB testée.

4.2.4.2 Construction des data profiles

Dans cette section, on introduit l'outil qui permet de comparer les différentes versions de SEGO-UTB entre elles. En effet, une méthode spécifique est nécessaire pour comparer ces versions sur un ensemble de problèmes de propriétés variées. Pour cela, on utilise les DP [72] qui sont originalement introduits dans le contexte de l'optimisation sans contrainte et sans dérivées.

TABLEAU 4.3 – Propriétés des 29 problèmes pris en compte dans cette étude. Sont spécifiés : le nombre de variables, le nombre de contraintes d'égalité, le nombre de contraintes d'inégalité et la valeur de référence utilisée.

Nom du Problème	**Nb. de Variables**	**Nb. de Eq. Cst.**	**Nb. de Ieq. Cst.**	**Valeur de Ref.**
G03	10	1	0	$-1{,}000$
G04	5	0	6	$-3{,}067 \cdot 10^{4}$
G05	4	3	2	5 126
G06	2	0	2	$-6\,962$
G07	10	0	8	24,23
G08	2	0	2	$-9{,}583 \cdot 10^{-2}$
G09	7	0	4	680,6
G10	8	0	6	7 049
G11	2	1	0	0,750
G12	3	0	1	$-1{,}000$
G13	5	3	0	$2{,}201 \cdot 10^{-3}$
G14	10	3	0	$-47{,}71$
G15	3	2	0	961,7
G16	5	0	38	$-1{,}918$
G17	6	4	0	8 864
G18	9	0	13	$-0{,}866\,1$
G21	7	5	1	193,8
G23	9	4	2	$-400{,}1$
G24	2	0	2	$-6{,}031$
WB4	4	0	6	0,473 4
GBSP	2	2	1	$-0{,}525\,2$
GTCD	4	0	1	$2{,}965 \cdot 10^{6}$
Hesse	6	0	6	$-310{,}0$
LAH	4	1	1	$5{,}176 \cdot 10^{-2}$
LSQ	2	0	2	0,600
SR7	7	0	11	2 994
MB	2	0	1	12,00
MBE	2	1	0	12,00
PVD4	4	0	3	5 809

Ils montrent les performances d'un algorithme d'optimisation étant donné un budget d'évaluations et une réduction de valeur de la fonction objectif par rapport au point initial choisi. Ainsi le test de convergence suivant est défini :

$$f\left(\boldsymbol{x}^{(0)}\right) - \tilde{f}\left(\boldsymbol{x}^{(l)}\right) \geq (1-\epsilon)\left(f\left(\boldsymbol{x}^{(0)}\right) - f_{opt}\right), \tag{4.24}$$

où $\epsilon \in [0,1]$ est la quantité de réduction demandée, f_{opt} représente la meilleure valeur de la fonction objectif découverte par l'ensemble des algorithmes d'optimisation testés et $f(\boldsymbol{x}^{(0)})$ est la valeur de la fonction objectif au point initial choisi. $\tilde{f}\left(\boldsymbol{x}^{(l)}\right)$ est la meilleure valeur de la fonction objectif obtenue jusqu'à l'évaluation l. Toutefois, la présence de contraintes dans l'ensemble des problèmes pris en compte ne permet pas l'utilisation de ce test de convergence. En effet, la valeur de $f\left(\boldsymbol{x}^{(0)}\right)$ peut ne pas être faisable et être plus faible que la solution du problème f_{opt}. On utilise donc le test proposée par DIOUANE et al. [31] qui n'utilise pas la valeur de $f(\boldsymbol{x}^{(0)})$ et qui s'exprime comme suit :

$$\tilde{f}\left(\boldsymbol{x}^{(l)}\right) - f_{opt} \leq \epsilon(|f_{opt}| + 1), \tag{4.25}$$

où $\tilde{f}\left(\boldsymbol{x}^{(l)}\right)$ est la meilleure valeur faisable de la fonction objectif obtenue jusqu'à l'évaluation l. Si aucun point faisable n'est découvert parmi les l premières évaluations, alors $\tilde{f}\left(\boldsymbol{x}^{(l)}\right) = +\infty$. Pour les problèmes utilisés dans cette étude, les meilleures valeurs faisables sont référencées dans le Tableau 4.3.

En réalité, on trace le pourcentage d'instances de problèmes résolus par chaque méthode d'optimisation pour un budget d'évaluations croissant. Ceci permet d'évaluer la rapidité de convergence de chaque méthode d'optimisation en réduisant le biais introduit par le choix des problèmes considérés. Soit $\mathcal{S}$ l'ensemble des méthodes testées et $\mathcal{P}$ l'ensemble des instances de problèmes. Un DP est, pour chaque méthode $s \in \mathcal{S}$, le pourcentage d'instances de problèmes résolus avec un budget de κ évaluations :

$$\frac{1}{|\mathcal{P}|}\,\text{size}\left\{p \in \mathcal{P} \;:\; \frac{t_{p,s}}{d_p} \leq \kappa\right\}, \tag{4.26}$$

où d_p est le nombre de variables de l'instance de problèmes $p \in \mathcal{P}$, $t_{p,s}$ est le nombre d'évaluations pour lequel la méthode d'optimisation $s \in \mathcal{S}$ satisfait le test de convergence (4.25) pour une tolérance ϵ donnée. Si le test de convergence n'est pas satisfait dans un budget maximum d'évaluations, alors $t_{p,s} = +\infty$. Pour finir, l'unité de budget est exprimée avec d_p pour permettre la combinaison d'instances de problèmes de dimension différente dans le même DP.

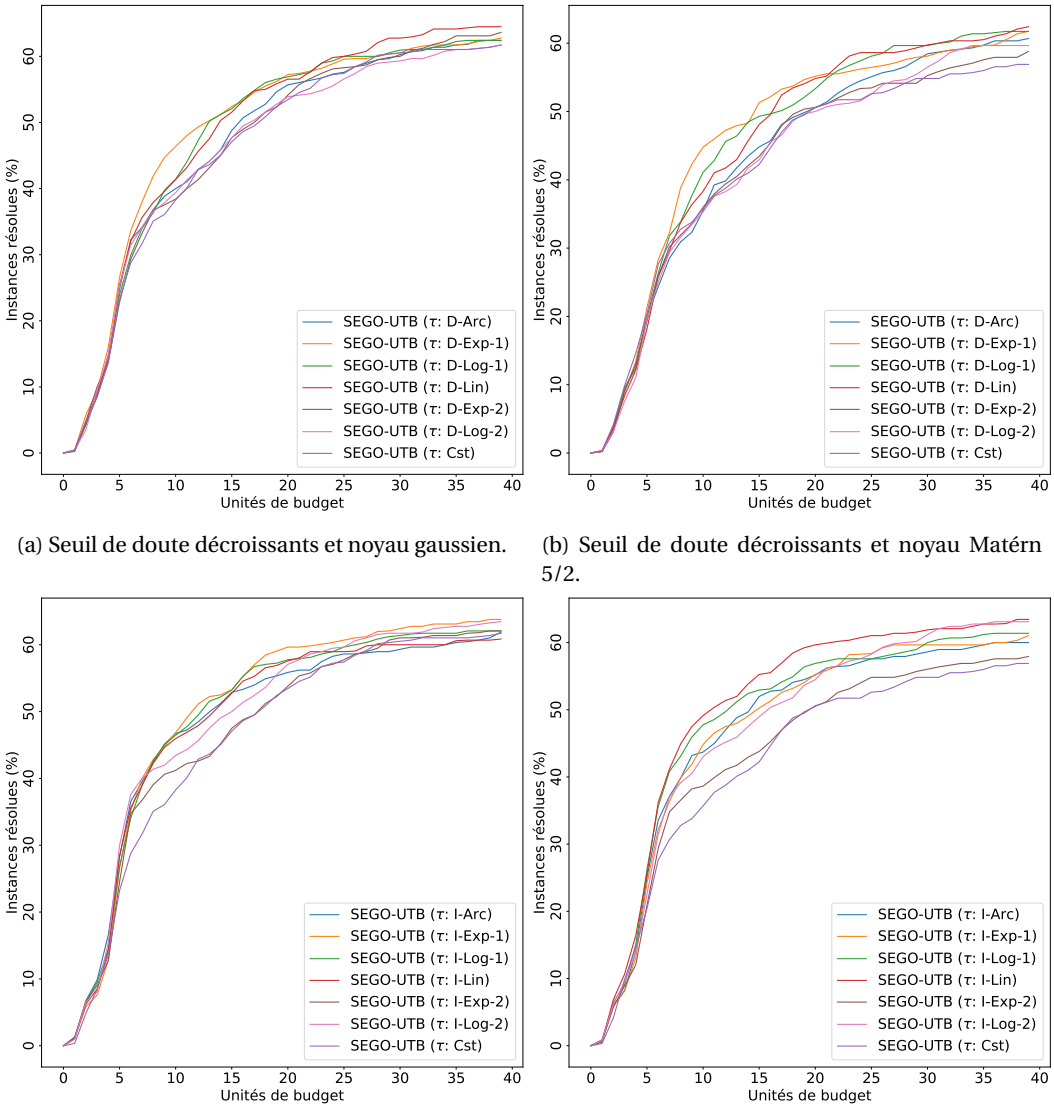
Dans cette étude, on se limite à un budget maximum de $40d_p$ évaluations puisqu'on s'intéresse à des problèmes dont l'évaluation est coûteuse en temps de calcul. De plus, on utilise $\epsilon = 10^{-3}$ comme précision dans le test de convergence (4.25) et on autorise une violation de chaque contrainte $\epsilon_c = 10^{-4}$.

4.2.4.3 Analyse des résultats

Dans cette section, on analyse les résultats des tests réalisés sur l'ensemble des 29 problèmes (voir Section 4.2.4.1). Pour cela, on utilise les DP introduits dans la Section 4.2.4.2. D'une part, on compare les mises à jour croissantes et décroissantes du seuil de doute pour les noyaux de covariance gaussien et Matérn 5/2. On sélectionne ensuite la méthode la plus performante sur

chacun des quatre DP produits. Pour finir, on confronte ces quatre méthodes grâce à un DP final.

La Figure 4.13 représente les quatre DP construits pour les différentes mises à jour du seuil de doute ainsi que pour les différents noyaux de covariance. Tout d'abord, on constate que l'ensemble des versions testées donnent des résultats du même type. Il est donc difficile de sélectionner les quatre meilleures. La Figure 4.13a montre les résultats pour les mises à jour décroissantes à noyau



(a) Seuil de doute décroissants et noyau gaussien.

(b) Seuil de doute décroissants et noyau Matérn 5/2.

(c) Seuil de doute croissants et noyau de covariance gaussien.

(d) Seuil de doute croissants et noyau Matérn 5/2.

FIGURE 4.13 – Data profile pour 29 problèmes pour deux noyaux de covariance et différentes évolutions du seuil de doute avec une violation des contraintes de $\epsilon_c = 10^{-4}$.

gaussien. On remarque que SEGO-UTB (τ : D-Exp-1) a les meilleurs performances pour un budget inférieur à 15. Plus particulièrement pour un budget de 15, SEGO-UTB (τ : D-Exp-1), (τ : D-Lin) et (τ : D-Log-1) résolvent environ 50% des problèmes, soit 5% de plus que les autres versions. A la fin de l'optimisation, c.-à-d. lorsque le budget de $40d_p$ évaluations est atteint, SEGO-UTB (τ : D-Lin) résout le plus de problèmes avec 65% des instances résolues. On considère donc que SEGO-UTB (τ : D-Lin) apporte le meilleur compromis entre rapidité et robustesse de convergence. Les résultats des versions à seuil de doute décroissant et noyau Matérn 5/2 sont rassemblés dans la Figure 4.13b. Comme pour les versions à noyau gaussien, SEGO-UTB (τ : D-Exp-1) résout le plus

de problèmes pour un budget inférieur à 20. A la fin de l'optimisation, cette version, résolvant plus de 60% des problèmes, reste compétitive malgré les bonnes performances de SEGO-UTB (τ : D-Lin) et (τ : D-Log-1). C'est pourquoi, on sélectionne SEGO-UTB (τ : D-Exp-1) comme meilleure version à seuil de doute décroissant et à noyau Matérn 5/2. Pour le noyau gaussien et les seuils de doute croissants, la Figure 4.13c montre que les versions SEGO-UTB (τ : I-Exp-2) et (τ : Cst) résolvent le moins de problèmes pour un budget inférieur à 25. Les autres méthodes sont équivalentes avec plus de 60% d'instances résolues à la fin du processus d'optimisation. Toutefois, SEGO-UTB (τ : I-Exp-1) résout plus d'instances que toutes les autres versions quel que soit le budget pris en compte. On sélectionne donc SEGO-UTB (τ : I-Exp-1) sans hésitation. Pour finir, la Figure 4.13d montre clairement que SEGO-UTB (τ : I-Lin) surpasse en robustesse et en rapidité les autres versions de SEGO-UTB à noyau Matérn 5/2 et à mise à jour croissante. En effet, elle résout le plus d'instances jusqu'à un budget de 30 et reste compétitive ensuite.

Pour finir, on compare les quatre versions sélectionnées sur la Figure 4.14. On remarque que

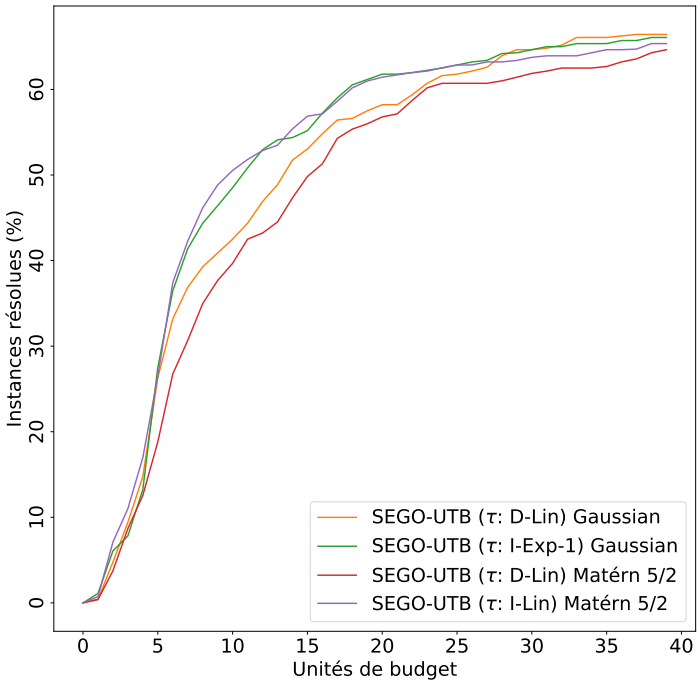


FIGURE 4.14 – Data profile pour 29 problèmes pour les meilleures évolutions du seuil de doute et pour deux noyaux de covariance avec une violation des contraintes de $\epsilon_c = 10^{-4}$.

les versions à seuil de doute croissant résolvent le plus d'instances de problèmes jusqu'à un budget de 25. Plus particulièrement, pour un budget de 10, 55% des problèmes sont résolus pour ces versions contre 45% pour les deux autres versions. A la fin de l'optimisation, l'écart se réduit avec un léger avantage pour SEGO-UTB (τ : D-Lin) à noyau gaussien. On voit aussi que les deux méthodes à noyau Matérn 5/2 résolvent moins d'instances que les méthodes à noyau gaussien. Ainsi la version offrant le meilleur compromis est SEGO-UTB (τ : I-Exp-1) à noyau gaussien.

Pour conclure sur l'étude de l'impact du seuil de doute et du noyau de covariance sur un ensemble de 29 problèmes, on a constaté que les méthodes à seuil de doute croissant sont plus efficaces en début d'optimisation bien qu'elles soient équivalentes aux méthodes à seuil de doute décroissant en fin d'optimisation. De plus, lorsque le budget est important, on a remarqué que les méthodes à noyau Matérn 5/2 montrent de moins bonnes performances que les méthodes à

noyau gaussien. C'est pourquoi, on préférera choisir SEGO-UTB (τ : I-Exp-1) à noyau gaussien lorsque l'on n'a pas d'information sur les propriétés du problème d'optimisation.

4.2.5 Conclusion

Dans cette section, on a évalué l'impact du seuil de doute et du noyau de covariance sur les performances de convergence de SEGO-UTB. Pour cela, on a d'abord introduit les choix d'implémentation dans la mise en œuvre de l'algorithme. Ensuite, on a présenté les différentes mises à jour du seuil de doute ainsi que les noyaux de covariance à prendre en compte dans l'étude. Deux grandes catégories de mises à jour ont été utilisées. D'une part, les mises à jour croissantes permettent l'exploitation des données du DoE au début de l'optimisation et explorent ensuite le domaine de conception à la recherche de zones faisables comportant des valeurs de fonction objectif plus faibles. D'autre part, les mises à jour décroissantes explorent d'abord le domaine de conception pour trouver la zone faisable optimale puis exploitent les données du DoE pour converger vers l'optimum. Pour les noyaux de covariance, on a choisi les noyaux gaussien et Matérn 5/2.

On a d'abord testé ces différentes versions sur quatre problèmes comprenant des contraintes multi-modales d'égalité et/ou d'inégalité. Sur ces quatre problèmes, le noyau gaussien et les seuils de doute décroissants ont montré les meilleures performances en terme de rapidité et de précision de convergence vers l'optimum. Ceci suggère que l'exploration du domaine de conception est nécessaire en début d'optimisation pour trouver la zone faisable optimale lorsque les contraintes sont multi-modales. Cependant, on ne parvient pas à identifier un unique seuil de doute décroissant qui convient aux quatre problèmes. Le choix du seuil de doute est donc fortement problème dépendant.

Afin de valider les résultats obtenus sur les quatre problèmes, on a réalisé des tests sur un ensemble de 29 problèmes comportant des contraintes d'égalité et/ou d'inégalité. De la même façon que précédemment, les tests ont montré que le noyau gaussien offre une convergence plus rapide et précise que le noyau Matérn 5/2. De plus, on a également constaté que les seuils de doute croissants permettent de résoudre plus de problèmes avec un faible budget d'évaluations comparé aux seuils de doute décroissants. En effet, l'exploitation des données du DoE en début d'optimisation permet une convergence rapide si l'exploration du domaine de conception est inutile (par exemple lorsque le domaine faisable est convexe). Finalement, SEGO-UTB (τ : I-Exp-1) à noyau gaussien surpasse les autres méthodes en rapidité de convergence et en nombre de problèmes résolus.

4.3 Comparaison de SEGO-UTB avec la littérature

Dans cette section, on évalue les performances de SEGO-UTB comparées aux algorithmes de la littérature présentés dans la Section 2.3 et aux algorithmes utilisés classiquement en optimisation sans dérivées. On va d'abord exposer les algorithmes pris en compte dans cette étude. Ensuite, on étudie les performances de ces algorithmes sur les quatre problèmes représentatifs déjà introduits dans la Section 4.2.3.1. Pour finir, on valide les résultats obtenus sur les quatre problèmes représentatifs à partir des tests sur l'ensemble de 29 problèmes déjà présentés dans la Section 4.2.4.1.

4.3.1 Algorithmes d'optimisation pris en compte dans la comparaison

Dans cette étude, SEGO-UTB est comparé à quelques algorithmes CBO qui ont été introduits dans la Section 2.3 :

— ALBO : un algorithme CBO utilisant le AL [84] (voir Section 2.3.1.4),

— SUR : un algorithme CBO reposant sur la fonction de mérite SUR [83] (voir Section 2.3.1.2),

— PESC : un algorithme CBO ayant recourt à la fonction de mérite PESC [51] (voir Section 2.3.1.3),

— EFI : un algorithme CBO exploitant à la fonction de mérite EFI [112] (voir Section 2.3.1.1),

— SEGO : un algorithme CBO utilisant la moyenne des GP modélisant les contraintes comme critère de faisabilité [9, 108] (voir Section 2.3.2.1).

Les algorithmes ALBO, EFI et SUR proviennent du package R DiceOption [105] alors que PESC est fourni par la toolbox Python Spearmint [1]. Pour finir, on acquiert SEGO à partir de la toolbox SEGOMOE [9] (voir Section 2.3.2.1) pour laquelle on impose un unique cluster pour chaque GP. Pour ALBO, EFI, SUR et PESC, les paramètres par défaut des toolboxes sont utilisés.

On cherche aussi à évaluer les performances de SEGO-UTB comparées aux algorithmes classiques d'optimisation sans dérivées. Pour cela, on inclut :

— Mesh adaptive direct search algorithm (MADS) [3] : un algorithme utilisant des maillages successifs du domaine de conception. On utilise plus particulièrement le logiciel NOMAD [65] avec les paramètres par défaut. Pour les contraintes, on choisit l'approche barrière progressive et extrême, ou progressive and extreme barrier (PEB) et on teste la version de NOMAD sans recherche en voisinage, ou variable neighborhood search (VNS), que l'on note NOMAD, et avec VNS, que l'on note NOMAD-VNS (NOMAD-VNS).

— COBYLA : un algorithme à régions de confiance reposant sur des approximations linéaires [90]. La fonction objectif et les contraintes sont approchées sont approchées linéairement et permettent de résoudre itérative ment le problème d'optimisation par programmation linéaire. On travaille avec l'implémentation de la toolbox Python Scipy [58].

De plus, on souligne que SUR, PESC, EFI, NOMAD et COBYLA ne prennent en compte que les contraintes d'inégalité. Afin de gérer les contraintes d'égalité, chaque contrainte de la forme $h(\boldsymbol{x}) =$ est transformée en deux contraintes d'inégalité de la forme $h(\boldsymbol{x}) \geq \epsilon_c$ et $h(\boldsymbol{x}) \leq \epsilon_c$.

On remarque également que contrairement aux algorithmes CBO, NOMAD et COBYLA n'ont besoin que d'un point initial pour commencer le processus d'optimisation. On choisit donc le meilleur point faisable du DoE initial comme point initial pour NOMAD et COBYLA. S'il n'y a pas de point faisable dans le DoE, on sélectionne le point avec la violation des contraintes la plus faible.

Pour finir, on utilise l'implémentation de SEGO-UTB introduite dans la Section 4.2.1.

4.3.2 Comparaison sur les quatre problèmes représentatifs

Dans cette section, les quatre problèmes représentatifs (voir Section 4.2.3.1) servent à comparer les algorithmes d'optimisation introduits dans la Section 4.3.1. Pour SEGO-UTB, on choisit la meilleure mise à jour du seuil de doute relative à chaque problème (voir Section 4.2.3.3).

1. https ://github.com/HIPS/Spearmint

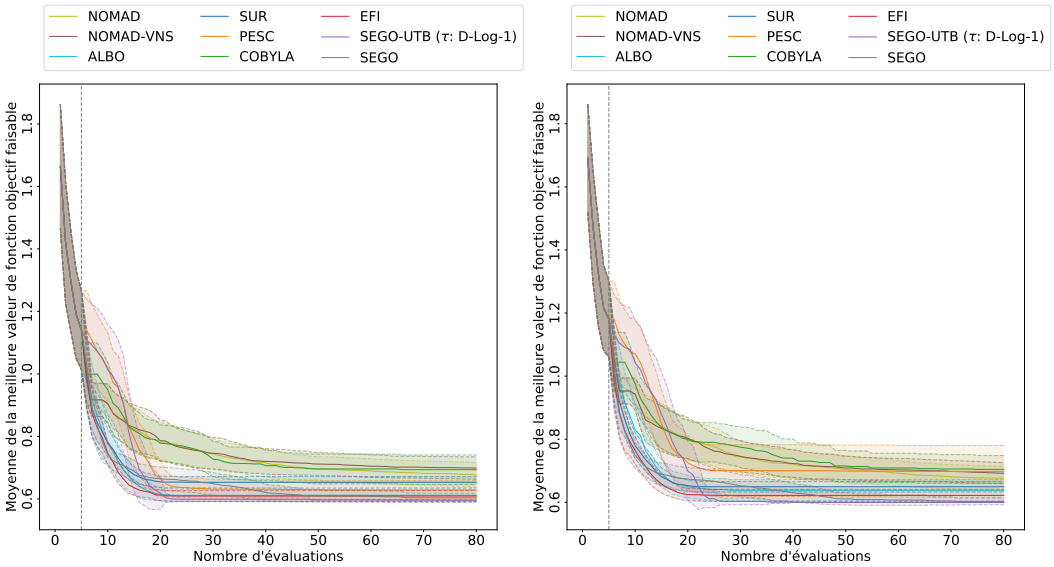
4.3.2.1 Détails relatifs aux tests

On étudie, avec la méthode des courbes de convergence introduite dans la Section 4.2.3.2, l'impact de la violation des contraintes sur les performances de l'ensemble des algorithmes. Les tests sont réalisés pour deux valeurs de violation : $\epsilon_c = 10^{-4}$ et $\epsilon_c = 10^{-2}$. On choisit un noyau de covariance gaussien pour l'ensemble des méthodes puisqu'il a montré de meilleurs résultats dans la Section 4.2. Toutefois, la plupart des algorithmes CBO pris en compte utilisent le noyau Matérn 5/2 par défaut. Dans un souci d'équité, on teste également l'ensemble des algorithmes avec un noyau Matérn 5/2 couplé à une violation de $\epsilon_c = 10^{-4}$.

4.3.2.2 Analyse des résultats

Dans cette section, on compare les algorithmes d'optimisation de la littérature avec SEGO-UTB. Pour cela, on construit trois courbes de convergence pour chaque problème. Les deux premières utilisent un noyau de covariance gaussien et les violations des contraintes $\epsilon_c = 10^{-4}$ et $\epsilon_c = 10^{-2}$. Ceci permet d'étudier l'impact de la violation sur les performances de ces algorithmes. La troisième emploie un noyau Matérn 5/2 et une violation de $\epsilon_c = 10^{-4}$. On compare ainsi SEGO-UTB avec les paramètres par défaut des algorithmes de la littérature.

Avec la Figure 4.15, on voit que COBYLA et NOMAD offrent les moins bonnes performances, c.-à-d. qu'ils convergent vers une valeur plus forte que les autres algorithmes. L'utilisation de l'option VNS ne permet pas une meilleure convergence pour NOMAD. Ceci est certainement dû à leur caractère plus local que les algorithmes CBO. De plus, on constate que PESC converge vers une valeur plus faible lorsque l'on relâche les contraintes. De la même manière que pour SEGO-UTB, les algorithmes ALBO, SUR et EFI convergent tous vers l'optimum global. Toutefois, seul SEGO-UTB trouve systématiquement l'optimum global comme le montre l'écart type nul. Pour les résultats avec le noyau Matérn 5/2 (voir Figure 4.16), on constate que SEGO-UTB et ALBO sont légèrement plus lents à converger qu'avec un noyau gaussien. Au contraire, la rapidité de convergence de EFI



(a) $\epsilon_c = 10^{-2}$.

(b) $\epsilon_c = 10^{-4}$.

FIGURE 4.15 – Courbes de convergence de l'optimisation du problème LSQ pour 8 optimiseurs bénéficiant d'un noyau gaussien et pour deux violations des contraintes de $\epsilon_c = 10^{-2}$ et de $\epsilon_c = 10^{-4}$.

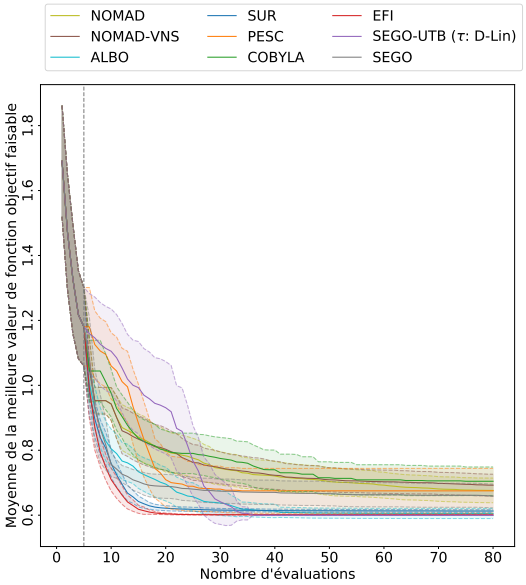
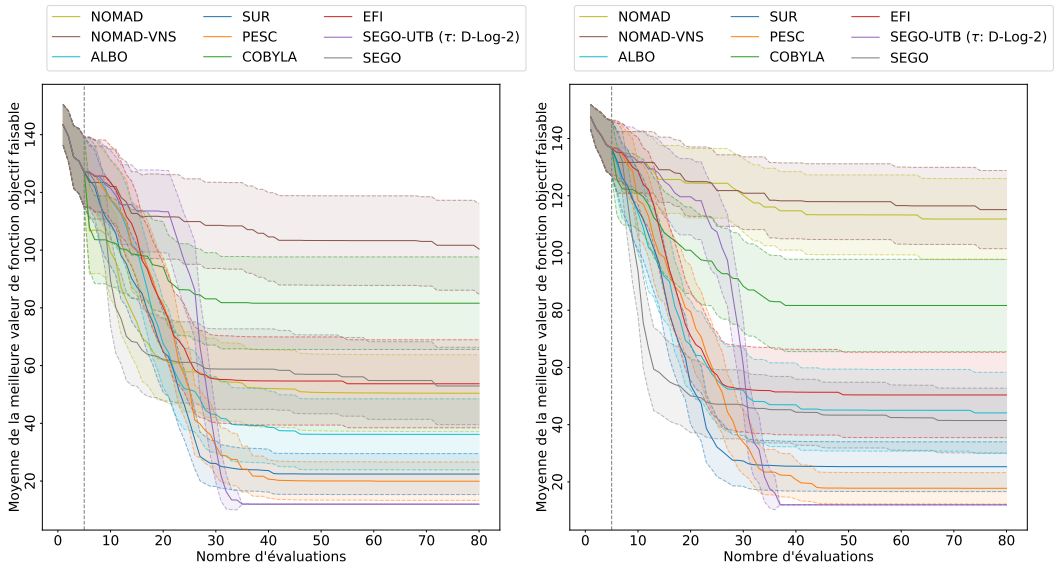


FIGURE 4.16 – Courbes de convergence de l'optimisation du problème LSQ pour 8 optimiseurs avec un noyau Matérn 5/2 et une violation des contraintes de $\epsilon_c = 10^{-4}$.

et SUR est la même. Pour finir, SEGO ne parvient pas à converger aussi bien qu'avec un noyau gaussien. En effet, la meilleure valeur faisable trouvée est légèrement plus forte.

Pour le problème MB et un noyau gaussien, on remarque clairement sur la Figure 4.17 que COBYLA, NOMAD et NOMAD-VNS donnent les moins bonnes performances pour une tolérance des contraintes $\epsilon_c = 10^{-4}$. Toutefois, lorsque l'on relâche les contraintes, NOMAD converge vers des valeurs plus faibles. Par contre, les performances de NOMAD-VNS ne changent pas. En outre, seul SEGO-UTB tend systématiquement vers la valeur optimale de fonction objective. Aucun des autres algorithmes, c.-à-d. SEGO, EFI, ALBO et SUR, ne converge vers cette valeur. Au contraire,



(a) $\epsilon_c = 10^{-2}$.

(b) $\epsilon_c = 10^{-4}$.

FIGURE 4.17 – Courbes de convergence de l'optimisation du problème MB pour 8 optimiseurs bénéficiant d'un noyau gaussien et pour deux violations des contraintes de $\epsilon_c = 10^{-2}$ et de $\epsilon_c = 10^{-4}$.

les résultats de la Figure 4.18 pour le noyau Matérn 5/2 montrent que EFI, ALBO et SUR trouvent

presque toujours la valeur optimale du problème MB. SEGO ne parvient toujours pas à converger

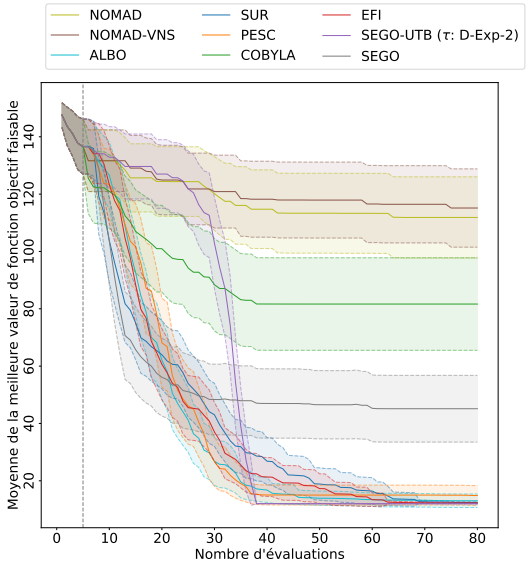


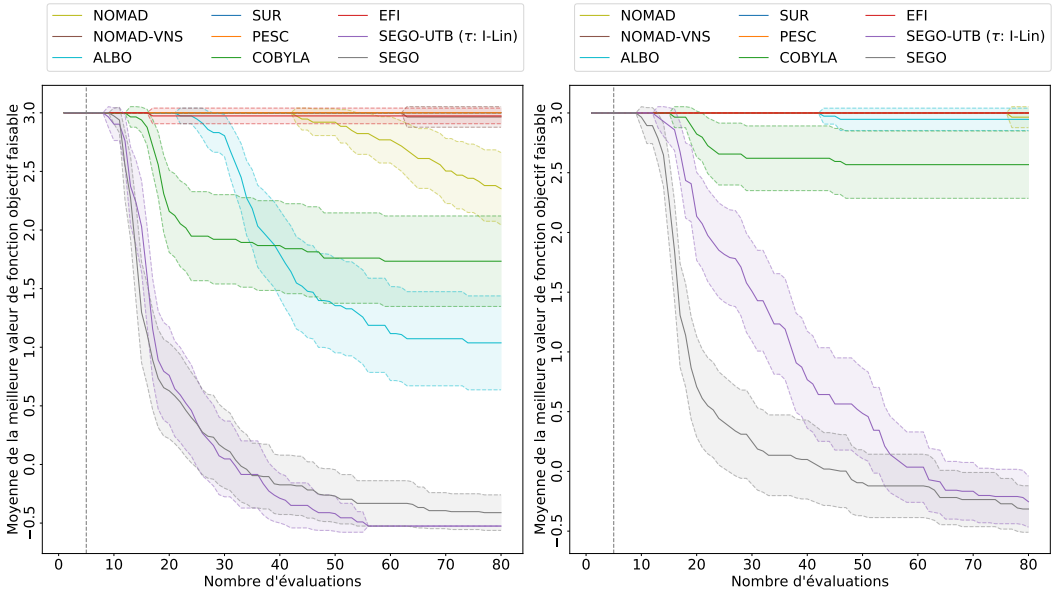
FIGURE 4.18 – Courbes de convergence de l'optimisation du problème MB pour 8 optimiseurs avec un noyau Matérn 5/2 et une violation des contraintes de $\epsilon_c = 10^{-4}$.

systématiquement vers cette valeur au vu de l'écart type. La caractéristique d'exploitation exclusive de SEGO restreint la recherche à des zones faisables sous optimales. Pour finir, SEGO-UTB reste la méthode convergeant le plus vite et systématiquement vers la solution du problème. Ainsi, SEGO-UTB arrive toujours à trouver la zone faisable optimale avec son caractère exploratoire. On le repère, sur la Figure 4.18, grâce au plateau se trouvant en début d'optimisation, c.-à-d. pour moins de 30 évaluations.

Concernant le problème GSBP, la Figure 4.19 montre que SEGO et SEGO-UTB surpassent largement les autres méthodes pour les deux tolérances prises en compte pour le noyau gaussien. On remarque que NOMAD, NOMAD-VNS, EFI, SUR et PESC sont incapables de trouver un point faisable pour la tolérance $\epsilon_c = 10^{-4}$. On suppose que ces mauvaises performances sont dues aux contraintes d'égalité présentes dans ce problème. Au contraire, ALBO trouve des points faisables et converge vers des valeurs plus faibles lorsque l'on relâche les contraintes. On souligne également que SEGO-UTB est la seule méthode convergeant systématiquement pour la tolérance $\epsilon_c = 10^{-2}$. Par contre, SEGO converge vers une valeur plus faible pour $\epsilon_c = 10^{-4}$. Ceci suggère que SEGO-UTB nécessite plus d'évaluations pour converger avec une tolérance plus stricte. En utilisant un noyau Matérn 5/2 (voir Figure 4.20), on ne constate pas de changement significatif par rapport au noyau gaussien.

Pour le problème LAH, les Figures 4.21 et 4.22 indiquent que SEGO-UTB et SEGO dominent les autres méthodes pour les deux noyaux considérés bien que EFI montre de bons résultats pour une violation des contraintes $\epsilon_c = 10^{-4}$. Plus particulièrement, on remarque que les performances de NOMAD, NOMAD-VNS, COBYLA, PESC, SUR, EFI et ALBO s'améliorent lorsque l'on relâche les contraintes. Ce phénomène est certainement provoqué par la présence de contraintes d'égalité dans le problème LAH. Pour finir, on voit que SEGO-UTB converge légèrement plus rapidement que SEGO vers des valeurs plus faibles.

Pour conclure, on a réalisé une comparaison de SEGO-UTB avec les algorithmes d'optimisation de la littérature. Pour cela, on a commencé par introduire les algorithmes de la littérature



(a) $\epsilon_c = 10^{-2}$. (b) $\epsilon_c = 10^{-4}$.

FIGURE 4.19 – Courbes de convergence de l'optimisation du problème GSBP pour 8 optimiseurs bénéficiant d'un noyau gaussien et pour deux violations des contraintes de $\epsilon_c = 10^{-2}$ et de $\epsilon_c = 10^{-4}$.

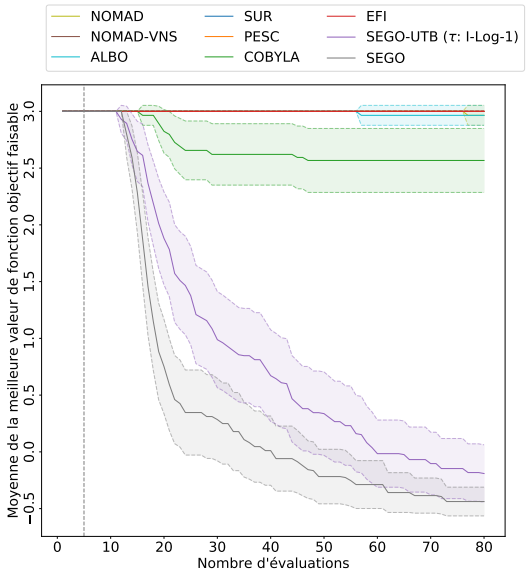
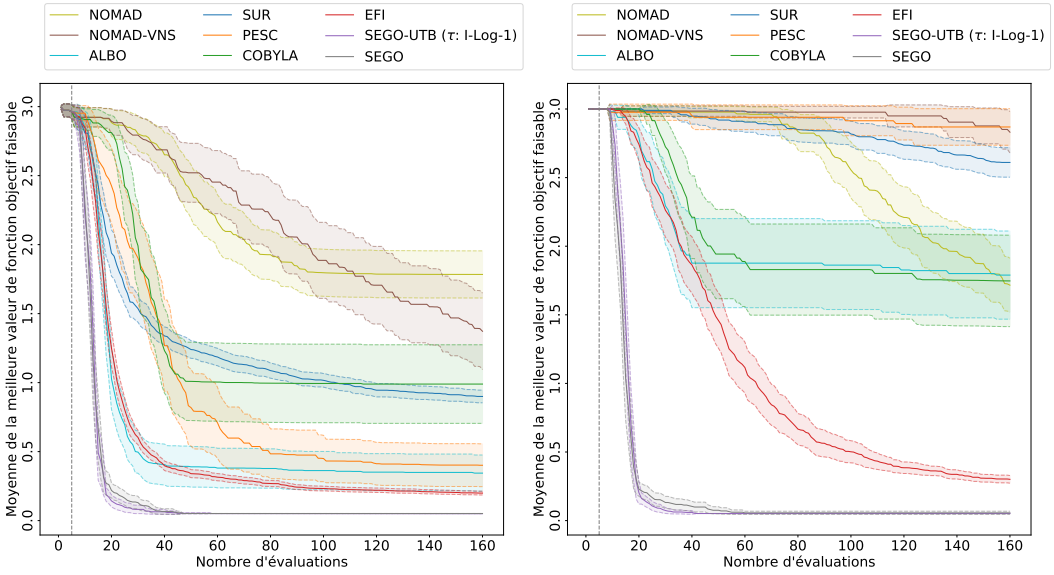


FIGURE 4.20 – Courbes de convergence de l'optimisation du problème GSBP pour 8 optimiseurs avec un noyau Matérn 5/2 et une violation des contraintes de $\epsilon_c = 10^{-4}$.



(a) $\epsilon_c = 10^{-2}$. (b) $\epsilon_c = 10^{-4}$.

FIGURE 4.21 – Courbes de convergence de l'optimisation du problème LAH pour 8 optimiseurs bénéficiant d'un noyau gaussien et pour deux violations des contraintes de $\epsilon_c = 10^{-2}$ et de $\epsilon_c = 10^{-4}$.

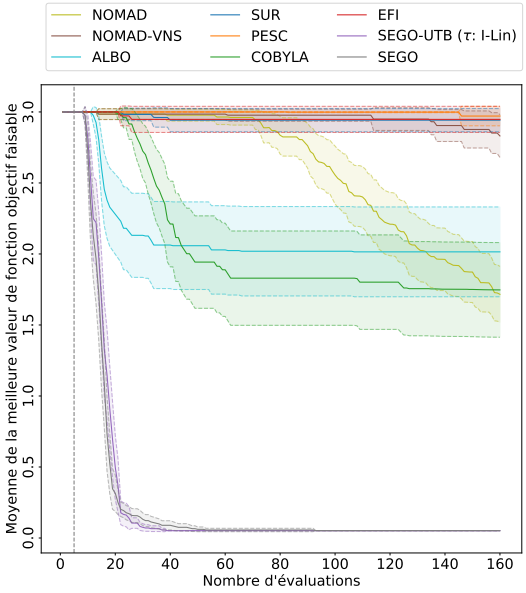


FIGURE 4.22 – Courbes de convergence de l'optimisation du problème LAH pour 8 optimiseurs avec un noyau Matérn 5/2 et une violation des contraintes de $\epsilon_c = 10^{-4}$.

ainsi que les toolboxes utilisées. Ensuite, on a précisé la méthode employée pour confronter ces algorithmes. Elle repose sur les courbes de convergence. On a notamment choisi de comparer les algorithmes pour deux tolérances de contrainte et deux noyaux de covariance. On a constaté que SEGO-UTB était plus performant sur la plupart des problèmes. Plus particulièrement avec la tolérance $\epsilon_c = 10^{-2}$, SEGO-UTB est le plus compétitif. Afin de confirmer ces résultats, on compare SEGO-UTB avec ces mêmes algorithmes sur un ensemble de 29 problèmes d'optimisation ayant des contraintes mixtes dans la section suivante.

4.3.3 Comparaison sur l'ensemble de 29 problèmes

Comme dans la Section 4.2, on tente de valider les résultats obtenus sur les quatre problèmes représentatifs à partir d'un ensemble de 29 problèmes (voir Section 4.2.4.1). Ces 29 problèmes comprennent des contraintes d'égalité et/ou d'inégalité. Pour valider ces résultats, on utilise la méthode des DP introduite dans la Section 4.2.4.2.

4.3.3.1 Détails relatifs aux tests et limitations

Pour réaliser ces tests, on utilise la méthode des DP présentée dans la Section 4.2.4. On utilisera donc les mêmes instances de problèmes que dans la Section 4.2.4.1. On rappelle toutefois que ALBO, PESC et SUR ne sont pas adaptés pour résoudre des problèmes de grande dimension (voir Section 2.3.1). Pour cela, on analyse le temps de calcul CPU pour cinq problèmes (LAH, MB, LSQ, GSBP et G07). On réalise donc 10 optimisations pour chacun des problèmes et pour chacune des méthodes CBO avec un budget d'évaluations de $40d$ où d est le nombre de variables de conception du problème. On souligne que l'implémentation de SUR dans DiceOptim ne permet pas de prendre en compte plus de 4 contraintes. Le Tableau 4.4 présente la moyenne du temps CPU pour 10 optimisations pour les 6 méthodes CBO sur 5 problèmes pour un budget de $40d$ évaluations. On voit clairement que EFI, SUR, PESC et ALBO nécessitent beaucoup de temps pour

TABLEAU 4.4 – Moyenne du temps CPU (en secondes) pour 10 optimisations pour les 6 méthodes CBO sur 5 problèmes pour un budget de $40d$ évaluations.

Problème \ Méthode	SEGO	SEGO-UTB	ALBO	SUR	EFI	PESC
LAH	203,64	2 708,99	5 131,97	2 592,32	1 212,04	2 729,64
GBSP	164,84	266,76	608,33	1 510,61	497,04	777,06
LSQ	157,43	177,04	833,83	417,97	207,82	592,70
MB	79,10	93,52	328,41	1 059,24	535,18	615,98
G07	2 098,86	4 245,55	611 025,72	–	1 195 346,89	55 755,87

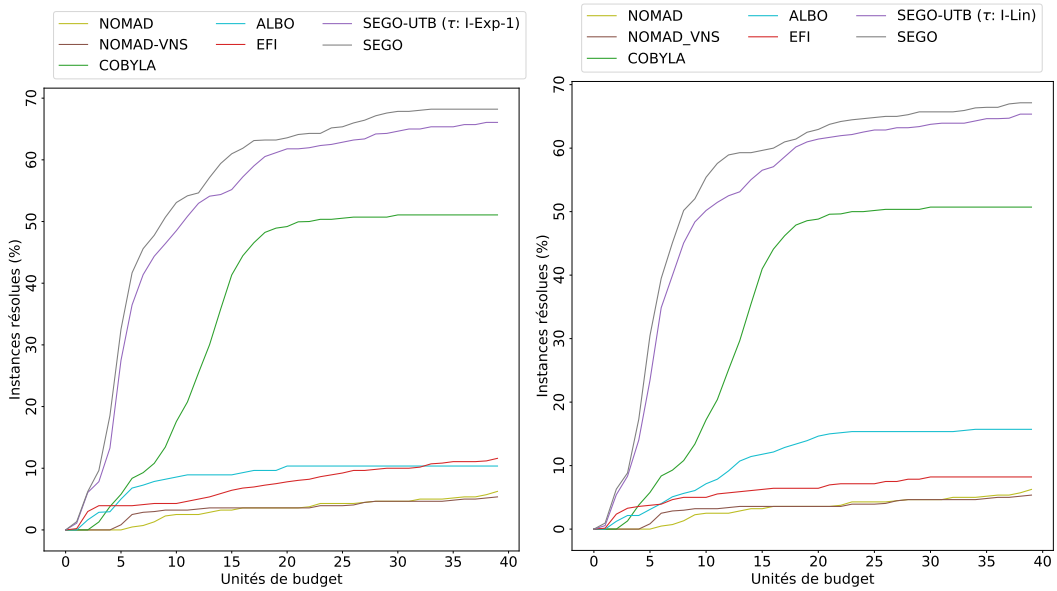
le problème G07 en particulier. Pour cette raison, dans la suite on ne teste que EFI et ALBO puisqu'ils ont montré de meilleurs résultats sur les 2 problèmes comportant des contraintes d'égalité et d'inégalité (LAH et GSBP). A cause de cette limitation, on ne prend en compte qu'une violation absolue $\epsilon_c = 10^{-4}$ pour chaque contrainte et deux noyaux de covariance (gaussien et Matérn 5/2) .

4.3.3.2 Analyse des résultats

Dans cette section, on compare les méthodes d'optimisation NOMAD, NOMAD-VNS, COBYLA, EFI, ALBO, SEGO et SEGO-UTB pour deux noyaux de covariance (gaussien et Matérn 5/2)

et pour une tolérance $\epsilon_c = 10^{-4}$. On utilise plus particulièrement SEGO-UTB (τ : I-Exp-1) à noyau gaussien et SEGO-UTB (τ : I-Lin) à noyau Matérn 5/2. On construit un DP pour chaque noyau considéré.

La Figure 4.23b montre que SEGO résout le plus d'instances de problèmes quel que soit le budget pris en compte. En effet, il résout plus de 65% des instances avec un budget de $40d$ évaluations alors que COBYLA, ALBO, EFI, NOMAD et NOMAD-VNS ne résolvent pas plus de 50% des instances. De plus, SEGO-UTB montre des capacités semblables bien qu'avec 2 à 4% d'instances de



(a) Noyau gaussien.

(b) Noyau Matérn 5/2.

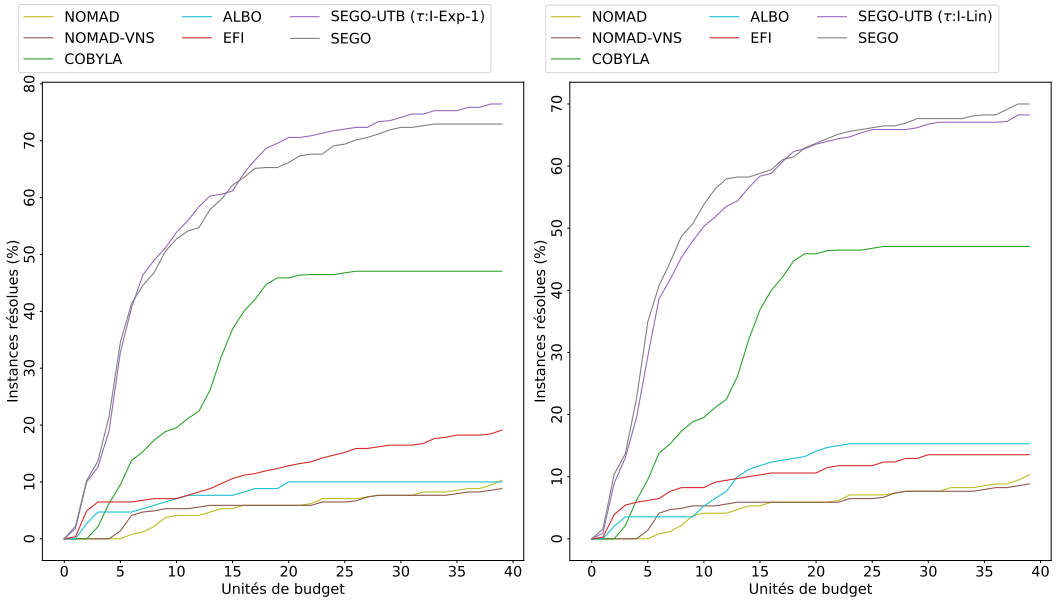
FIGURE 4.23 – Data profile pour 29 problèmes pour 7 optimiseurs et pour deux noyaux de covariance avec une violation des contraintes de $\epsilon_c = 10^{-4}$.

problèmes résolus en moins quel que soit le budget. Le grand nombre de problèmes à contraintes linéaires peut expliquer ces moins bons résultats. En effet, un ensemble de contraintes linéaires va définir une unique zone faisable convexe ce qui est fortement favorable à SEGO. Par contre, on voit que COBYLA surpasse ALBO, EFI et NOMAD avec 50% d'instances de problèmes résolus après un budget de $20d$ évaluations. NOMAD montre les moins bonnes performances avec 6 à 7% des instances résolues avec un budget de $40d$ évaluations. NOMAD-VNS montre des résultats du même type que NOMAD. Celles-ci sont certainement dues à la présence de contraintes d'égalité que NOMAD n'est pas capable de prendre en compte sans transformation en contraintes d'inégalité.

En utilisant un noyau gaussien (voir Figure 4.23a), on constate que les performances des algorithmes sont similaires au noyau Matérn 5/2. Toutefois, on remarque que ALBO résout moins d'instances de problèmes avec ce noyau de covariance.

Pour supprimer le biais introduit en comparant des méthodes d'optimisation ne pouvant pas gérer les contraintes d'égalité avec des méthodes pouvant les prendre en compte, on choisit de toutes les comparer sur l'ensemble des problèmes d'optimisation restreint aux problèmes ne comportant que des contraintes d'inégalité. On considère donc un ensemble de 17 problèmes d'optimisation. Ceci permet de comparer NOMAD, COBYLA, PES et EFI sur des problèmes pour lesquels ils sont développés. La Figure 4.24 représente les DP pour les 7 algorithmes d'optimisation consi-

dérés pour deux noyaux de covariances : le noyau Matérn 5/2 et le noyau gaussien. Les mêmes



(a) Noyau gaussien.

(b) Noyau Matérn 5/2.

FIGURE 4.24 – Data profile pour 17 problèmes comportant uniquement des contraintes d'inégalité pour 7 optimiseurs et pour deux noyaux de covariance avec une violation des contraintes de $\epsilon_c = 10^{-4}$.

tendances que précédemment sont observées. On constate toutefois que SEGO-UTB est plus performant que SEGO si on ne considère que les contraintes d'inégalité.

Pour conclure, on a comparé SEGO-UTB aux méthodes de la littérature sur un ensemble de 29 problèmes comportant des contraintes d'égalité et/ou d'inégalité. On a remarqué que SEGO est le plus performant sur cet ensemble de problèmes bien que SEGO-UTB soit aussi très compétitif. Les moins bons résultats de SEGO-UTB comparés à SEGO sont certainement dus au grand nombre de contraintes linéaires dans cet ensemble. Ceci est en effet défavorable à SEGO-UTB à cause de son caractère fortement exploratoire. Les autres méthodes offrent de moins bonnes performances, notamment NOMAD, EFI et ALBO. Ces performances ne sont pas surprenantes puisque NOMAD et EFI ne prennent pas en compte les contraintes d'égalité sans transformation et ALBO n'est pas adapté à la grande dimension, c.-à-d. plus de 4 à 5 variables de conception. Finalement, SEGO et SEGO-UTB apparaissent comme les seules alternatives viables pour résoudre des problèmes d'optimisation sous contraintes mixtes et en grande dimension.

4.3.4 Conclusion

Dans cette section on a cherché à comparer SEGO-UTB avec les algorithmes de la littérature.

On a d'abord introduit les huit méthodes prises en compte dans cette étude ainsi que les toolboxes utilisées. Six de ces méthodes sont des algorithmes CBO (SEGO-UTB, SEGO, EFI, SUR, ALBO et PESC) et les deux autres sont des algorithmes classiques de l'optimisation sous contraintes sans dérivées (NOMAD et COBYLA).

Ensuite, on a étudié la convergence de ces algorithmes vers l'optimum global de quatre problèmes représentatifs comprenant des contraintes d'égalité et/ou d'inégalité. On a testé deux valeurs de tolérance pour les contraintes ($\epsilon_c = 10^{-2}$ et $\epsilon_c = 10^{-4}$) et deux noyaux de covariance pour les GP (gaussien et Matérn 5/2). Ceci a permis de déduire que SEGO-UTB est particulièrement

efficace sur l'ensemble de ces problèmes et est la seule méthode à converger systématiquement vers l'optimum global avec une tolérance de $\epsilon_c = 10^{-2}$. SEGO montre aussi de bons résultats mais n'est pas capable de converger systématiquement vers l'optimum lorsque les contraintes forment des zones faisables disjointes (pour les problèmes MB et GSBP par exemple). Les autres méthodes sont moins compétitives sur les problèmes qui comportent des contraintes d'égalité.

Afin de valider ces résultats, on a testé ces 8 algorithmes sur l'ensemble de 29 problèmes déjà introduits dans la Section 4.2.4.1. Ceci a permis d'évaluer les performances de SEGO-UTB sur un ensemble de problèmes plus large et donc d'avoir des propriétés de convergence plus globales. Au vu des mauvais résultats de PESC sur l'ensemble des problèmes à contraintes d'égalité, on a choisi de ne pas le prendre en compte dans cette étude. De plus, SUR ne gèrant pas plus de 4 contraintes dans l'implémentation DiceOptim, il n'a pas été pris en compte dans cette étude. On a également remarqué que EFI et ALBO nécessitent un temps de calcul CPU important. C'est pourquoi on n'a testé que deux noyaux de covariance pour une unique tolérance $\epsilon_c = 10^{-4}$. SEGO a montré les meilleurs performances de convergence en résolvant le plus de problèmes quel que soit le budget d'évaluations pris en compte. Ceci est certainement dû aux contraintes linéaires qui ne nécessitent pas une exploration massive du domaine de conception pour découvrir la zone faisable optimale. Toutefois, SEGO-UTB reste très compétitif. Les autres algorithmes ont offert de moins bons résultats à cause de la présence de contraintes d'égalité.

Pour finir, ces résultats soulignent que SEGO-UTB doit être utilisé si on sait que l'optimisation nécessite une exploration forte du domaine de conception ; par exemple, lorsque les contraintes sont fortement multi-modales et/ou forment des zones faisables disjointes. Dans le cas où aucune information n'est disponible sur le problème, il semble préférable d'utiliser SEGO.

4.4 Synthèse du chapitre

Dans ce chapitre, l'objectif était de mettre en place et d'étudier une méthode d'optimisation pour les problèmes coûteux à évaluer en temps de calcul, comportant un grand nombre de contraintes multi-modales d'égalité et/ou d'inégalité (≈ 40) et un grand nombre de variables de conception (≈ 20).

Pour cela, on a d'abord étudié les méthodes d'optimisation CBO existantes afin d'identifier celle qui correspond le mieux à ces exigences. La méthode SEGO-UTB s'est avérée la plus prometteuse bien qu'elle ne prenne pas en compte les contraintes d'égalité. C'est pourquoi on a étendu SEGO-UTB aux contraintes d'égalité avec un critère de faisabilité spécifique. Celui-ci permet notamment de prendre en compte l'écart type des GP modélisant les contraintes grâce à un seuil de doute sur la qualité du GP.

Ainsi, SEGO-UTB dépend de deux paramètres principaux : le seuil de doute et le noyau de covariance de GP. On a donc examiné l'impact du seuil de doute, à travers l'étude de treize mises à jour dépendant du nombre de points utilisé pour construire le GP, et du noyau de covariance, avec les noyaux gaussien et Matérn 5/2, sur les performances de convergence de SEGO-UTB. On a d'abord analysé les résultats de SEGO-UTB avec ces différents paramètres sur un ensemble de quatre problèmes comportant des contraintes multi-modales d'égalité et/ou d'inégalité. On a montré que le noyau gaussien et qu'un seuil de doute décroissant sont préférables bien que le choix de la mise à jour du seuil de doute soit fortement problème dépendant. On en a déduit

qu'une phase d'exploration en début d'optimisation est nécessaire lorsque les contraintes sont multi-modales. Afin de valider ces résultats, on a évalué les performances de SEGO-UTB pour ces différents paramètres sur un ensemble de 290 instances de problèmes. Ceci a permis de réduire le biais introduit par les problèmes considérés. Ces tests ont validé la supériorité du noyau gaussien pour SEGO-UTB. Toutefois, une mise à jour croissante ((τ : I-Exp-1) à noyau gaussien) est ici préférée certainement à cause du grand nombre de contraintes linéaires utilisées dans cet ensemble d'instances. En effet, une exploration intensive du domaine n'est plus nécessaire dans ce cas puisque les contraintes linéaires ne sont pas multi-modales.

Ensuite, on s'est intéressé aux performances de SEGO-UTB comparées aux méthodes d'optimisation de la littérature (NOMAD, COBYLA, EFI, SUR, ALBO, PESC). Comme précédemment, on a confronté ces algorithmes sur l'ensemble de quatre problèmes comportant des contraintes multi-modales d'égalité et/ou d'inégalité. On a constaté que SEGO-UTB converge presque systématiquement vers l'optimum global du problème quel que soit le noyau de covariance, la tolérance sur les contraintes ou le budget considéré. SEGO est aussi très compétitif bien qu'il ne parvienne pas toujours à résoudre les problèmes à zones faisables disjointes, c.-à-d. les problèmes MB et GSBP. Par contre, les autres algorithmes luttent pour converger lorsque le problème comporte des contraintes d'égalité. Pour finir, on a validé ces résultats sur l'ensemble de 290 instances de problèmes. Cependant, EFI, SUR, PESC et ALBO sont très gourmands en temps de calcul CPU. C'est pourquoi seuls ALBO, EFI, NOMAD, COBYLA, SEGO et SEGO-UTB sont pris en compte dans cette étude et l'ensemble de problèmes test est réduit aux problèmes avec moins de 10 variables de conception. SEGO a offert ici les meilleures performances en résolvant le plus d'instances quel que soit le budget considéré. SEGO-UTB a aussi fourni de bons résultats bien que légèrement moins bons que SEGO. Comme précédemment, ceci est dû aux contraintes linéaires présentes dans la majorité des problèmes de l'ensemble. En effet, il n'est pas nécessaire, dans ce cas, d'explorer le domaine de conception pour trouver la zone faisable optimale. Ainsi, SEGO est fortement avantagé sur cet ensemble de problèmes. Par contre, les autres algorithmes n'ont pas de résultats convaincants ce qui est certainement lié à la présence de contraintes d'égalité ou au caractère local de certains algorithmes ou encore au nombre de variables de conception à prendre en compte.

Pour résumer, SEGO-UTB semble adéquat pour résoudre des problèmes d'optimisation sous contraintes multi-modales d'égalité et/ou d'inégalité qui nécessitent une exploration intensive du domaine de conception ; par exemple lorsque les zones faisables sont disjointes ou non convexes. En dehors de ces situations, SEGO semble plus adapté. Toutefois, aucun test sur des problèmes de conception aéronautique n'a encore été mené. Dans le chapitre suivant, on s'intéresse à l'utilisation de SEGO et SEGO-UTB pour des problèmes de conception aéronautique.

Pour conclure, une grande partie du travail présenté dans ce chapitre à fait l'objet d'une publication dans un journal [96] et de communication lors de conférences [92, 94].

Récapitulatif du chapitre

On a répondu aux objectifs introduit en début de chapitre grâce aux points suivants :

- Extension aux contraintes d'égalité d'un critère de faisabilité.
- Étude de l'impact de l'évolution du seuil de doute (13 évolutions possibles) sur 29 problèmes analytiques.
- Utilisation de DP et de courbes de convergence pour l'étude d'impact.
- Étude de l'impact du noyau de covariance des GP (Gaussien et Matérn 5/2) sur 29 problèmes analytiques.
- Comparaison de la méthode avec les algorithmes de la littérature (PESC, EFI, ALBO SUR, SEGO, NOMAD et COBYLA) sur 29 problèmes analytiques.

Chapitre 5

Application à la conception avion avant-projet

« *Je me souviens.* »

Devise du Québec

Sommaire

Objectifs du chapitre

— Évaluer la méthode SEGO-UTB sur un problème de conception avion avant-projet.

— Évaluer la toolbox SEGOMOE sur un problème de conception avion avant-projet dans un cadre industriel.

Dans ce chapitre, l'objectif principal est de tester SEGO-UTB sur des problèmes de conception avion avant-projet. Pour cela, on évalue d'abord les performances de SEGO-UTB sur la conception d'un avion à propulsion électrique distribuée. Ce problème d'optimisation est un cas test de recherche développé à l'ONERA et à l'ISAE-SUPAERO dans le cadre de la thèse de Alessandro SGUEGLIA [115]. Ensuite, on analyse les performances de SEGO-UTB et SEGOMOE sur un problème d'optimisation multi-disciplinaire et multi-niveaux dans un contexte industriel. Plus spécifiquement, on teste SEGO-UTB et SEGOMOE sur la configuration de recherche mise en place par Bombardier Aviation sur deux niveaux de fidélité. Ces tests sont réalisés lors d'une mobilité internationale dans le locaux de Bombardier Aviation à Montréal (CA) de Mai à Août 2019. Ceci nous permet d'évaluer la portabilité de SEGO-UTB et SEGOMOE dans un contexte industriel. Pour finir, on souligne que l'ensemble de ces travaux a fait l'objet d'une publication dans un journal [96] et d'une communication lors de conférence [93, 95, 97].

5.1 Conception d'un avion hybride électrique

Dans cette section, on cherche à évaluer les performances de SEGO-UTB pour la conception d'un avion hybride à propulsion électrique distribuée [116]. La Figure 5.1 montre le concept de cette configuration. La principale caractéristique est la chaîne de propulsion placée sur le long

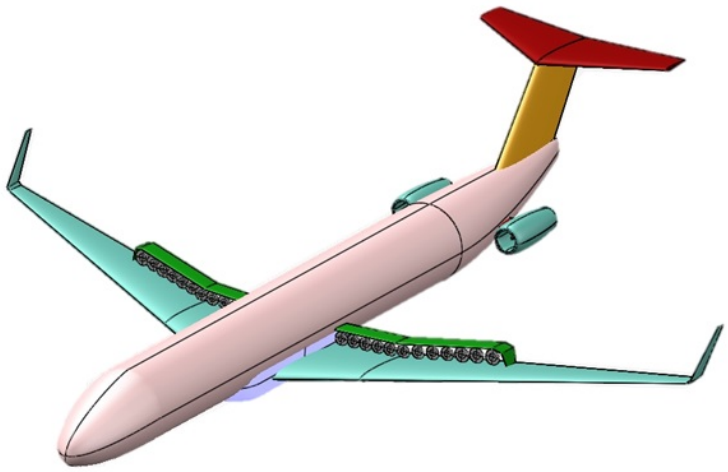


FIGURE 5.1 – Un concept d'avion hybride comportant une propulsion distribuée [116].

du bord de fuite de l'extrados des ailes. Elle se compose de batteries et de turbo-générateurs qui fournissent l'énergie électrique à un ensemble de moteurs à soufflante carénée. Les deux turbo-générateurs sont placés à l'arrière de l'avion (voir Figure 5.1) tandis que les batteries se trouvent dans la soute. L'avion doit être capable de voler jusqu'à 3 000 ft en utilisant uniquement le mode électrique et doit transporter 150 passagers dans un rayon de 1 200 nmi. De plus, on souhaite réduire les émissions de cette configuration au maximum.

L'outil de conception pour configuration avion à voilure fixe, ou fixed-wing aircraft sizing tool (FAST) [110] est utilisé pour évaluer les performances de ce concept et est entièrement codé en Python. Il repose sur des méthodes d'ingénierie offrant des résultats fiables avec un coût de calcul faible [104]. Certains des modules, développés par ROSKAM [104], sont modifiés pour prendre en compte les caractéristiques introduites par la chaîne de propulsion hybride [116]. En outre,

l'utilisation de la MDO s'avère primordiale pour réaliser le dimensionnement de ce concept non conventionnel puisque les disciplines sont fortement interdépendantes (par exemple aérodynamique et structure) [38, 48]. Une version open-source de cet outil est disponible depuis peu, outil de conception pour configuration avion du futur - conception d'avion complet, ou future aircraft sizing tool - overall aircraft design (FAST-OAD) [1], bien qu'elle ne soit pas totalement identique au code FAST utilisé ici.

Ainsi, la recherche d'un avion optimal (par exemple au sens de la consommation de carburant) définit un problème d'optimisation comportant des contraintes, parfois multi-modales, d'égalité et d'inégalité. C'est pourquoi, il est intéressant d'utiliser SEGO-UTB pour résoudre ce problème. Dans cette section, on commence par présenter le problème d'optimisation lié à la configuration avion introduite. Ensuite, on détaille les tests réalisés suivi de l'analyse des résultats obtenus à travers des courbes de convergence et des courbes de coordonnées parallèles. Ce travail a été réalisé en collaboration avec Alessandro Sgueglia [2], en thèse dans l'équipe de recherche [115].

5.1.1 Présentation du problème d'optimisation

Le problème d'optimisation lié à cette configuration consiste à minimiser la quantité totale d'énergie consommée, ou total energy consumption (TEC) (c.-à-d. la somme de l'énergie consommée en carburant et en électricité) en modifiant la géométrie de l'avion (par exemple les ailes, l'empennage horizontal et l'empennage vertical).

L'aile se définit plus particulièrement avec la surface S_w, le rapport de forme AR_w, la position sur le fuselage x_w et l'angle de flèche à 25% de la corde $\Lambda_{25,w}$. L'empennage horizontal et l'empennage vertical sont respectivement paramétrés par la surface S_{HT} et S_{VT}, le rapport de forme AR_{HT} et AR_{VT} et l'angle de flèche à 25% de la corde $\Lambda_{25,HT}$ et $\Lambda_{25,VT}$. En ce qui concerne la chaîne de propulsion, on considère le volume des batteries τ_b. On prend aussi en compte l'altitude de croisière h_{cruise} pour obliger l'avion à voler au maximum de l'efficacité aérodynamique. Grâce à ces notations, on écrit le vecteur des 12 variables de conception

$$\boldsymbol{x} = \left[x_w, S_w, AR_w, \Lambda_{25,w}, S_{HT}, AR_{HT}, \Lambda_{25,HT}, S_{VT}, AR_{VT}, \Lambda_{25,VT}, \tau_b, h_{cruise}\right]^\top$$

qui contient les paramètres géométriques de l'aile, de l'empennage horizontal et de l'empennage vertical.

Les bornes du domaine de conception $\Omega \subset \mathbb{R}^{12}$ sont données par le Tableau 5.1 pour lequel les valeurs sont choisies de manière classique pour le type de configuration considérée [104].

Pour assurer la faisabilité de l'avion, l'aile doit contenir suffisamment de carburant pour une mission complète (c.-à-d. MFW $\geq m_f$ avec MFW la masse maximale de carburant pouvant être contenue dans l'aile et m_f la masse nécessaire de carburant pour réaliser la mission définie dans la Section 5.1) et générer la portance nécessaire pour la phase d'approche (c.-à-d. $C_{L_{max}} \geq C_{L_{app}}$ où $C_{L_{max}}$ est le coefficient de portance maximal développé par l'aile et $C_{L_{app}}$ est le coefficient de portance nécessaire en phase d'approche). L'empennage horizontal doit assurer un moment de tangage positif au décollage [99] ($\mathcal{M}_{takeoff} = 0$). L'empennage vertical est contraint de contrebalancer le lacet du fuselage en vol de croisière [99] ($\mathcal{N}_{cruise} = 0$). Les batteries sont sujettes à

1. https://github.com/fast-aircraft-design/FAST-OAD
2. ONERA, DTIS, Université de Toulouse, Toulouse, France & ISAE-Supaéro, DCAS, Université de Toulouse, Toulouse, France

TABLEAU 5.1 – Définition du domaine de conception.

Variable	Unit	Min.	Max.	Symbol	Variable	Unit	Min.	Max.	Symbol
h_{cruise}	[kft]	27	35	DV_11	S_{HT}	[m^2]	20	80	DV_4
τ_b	[m^3]	1.5	3.0	DV_10	AR_{HT}	–	3	5	DV_5
x_w	[m]	15	18	DV_0	$\Lambda_{25,HT}$	[°]	25	45	DV_6
S_w	[m^2]	100	130	DV_1	S_{VT}	[m^2]	20	50	DV_7
AR_w	–	9	12	DV_2	AR_{VT}	–	1	2	DV_8
$\Lambda_{25,w}$	[°]	20	45	DV_3	$\Lambda_{25,VT}$	[°]	30	45	DV_9

deux contraintes liées à des exigences de puissance et d'énergie disponible. D'une part, les batteries sont forcées de produire suffisamment de puissance au décollage ($P_b \geq P_{takeoff}$). D'autre part, on oblige qu'il y ait au moins 20% de l'énergie maximale disponible dans les batteries lors de l'atterrissage. Le paramètre qui contrôle la quantité d'énergie restante est l'état de charge, ou state of charge (SoC) défini comme le ratio entre l'énergie consommée et la capacité énergétique totale. Cette condition permet de définir la deuxième exigence sur les batteries ($SoC_{min} \geq 0.20$), où SoC_{min} est la limite de sécurité pour ne pas endommager le système [130]. On introduit aussi deux contraintes de stabilité. La première permet d'assurer la stabilité statique longitudinale de l'avion. Elle utilise la marge statique SM qui doit être supérieure à $SM_{min} = 0.05$ [99]. La deuxième assure que l'avion réponde suffisamment rapidement aux commandes. Il est nécessaire de ne pas avoir une stabilité statique longitudinale trop forte. On a donc une marge statique maximale $SM_{max} = 0.10$ à ne pas dépasser [99]. D'autres contraintes proviennent de restrictions classiques des aéroports pour le concept d'avion considéré. La longueur de piste au décollage, ou takeoff field length (TOFL) et l'envergure de l'aile b_w ne doivent pas dépasser respectivement $TOFL_{max} = 2.2$ km et $b_{w_{max}} = 36$ m [28]. Pour finir, la dernière contrainte oblige l'avion à voler à l'altitude maximale d'efficacité, c.-à-d. $C_{L_{cruise}} = C_{L_{opt}}$ où $C_{L_{cruise}}$ est le coefficient de portance en croisière et $C_{L_{opt}}$ est le coefficient optimal. Pour résumer, les contraintes d'égalité sont données par $\boldsymbol{h} = [\mathcal{M}, \mathcal{N}, C_{L_{cruise}} - C_{L_{opt}}]^\top$ et les contraintes d'inégalité par $\boldsymbol{g} = [b_{w_{max}} - b_w, MFW - m_f, C_{L_{max}} - C_{L_{app}}, SoC - SoC_{min}, P_b - P_{takeoff}, TOFL_{max} - TOFL, SM - SM_{min}, SM_{max} - SM]^\top$.

Pour chacune des contraintes prises en compte dans ce problème, il existe une tolérance sur la violation qui est conduite par les propriétés physiques du problème [104]. Ces tolérances sont données par $\epsilon_h = \left[10^{-2}, 10^{-2}, 10^{-2}\right]^\top$ pour les contraintes d'égalité $\boldsymbol{h}$ et $\epsilon_g = \left[10^{-2}, 100, 10^{-2}, 10^{-2}, 1\,000, 100, 10^{-2}, 10^{-2}\right]^\top$ pour les contraintes d'inégalité $\boldsymbol{g}$.

Finalement, on obtient le problème MDO suivant :

$$\min_{\boldsymbol{x}\in\Omega} \left\{ TEC(\boldsymbol{x}) \text{ s.t. } \boldsymbol{g}(\boldsymbol{x}) \geq 0 \text{ et } \boldsymbol{h}(\boldsymbol{x}) = 0 \right\}, \tag{5.1}$$

qui comprend 12 variables de conception, 11 contraintes dont 3 d'égalité et 8 d'inégalité. Le Tableau 5.2 résume le problème d'optimisation introduit dans cette section.

5.1.2 Choix des algorithmes et détails des tests réalisés

On ne considère dans cette section que COBYLA [90], NOMAD [3, 65], SEGO [108] et SEGO-UTB à cause du temps de calcul CPU prohibitif des autres méthodes CBO (voir Tableau 4.4). Comme, NOMAD-VNS ne s'est pas révélé plus performant que NOMAD sur l'ensemble de 29 pro-

TABLEAU 5.2 – Définition du problème d'optimisation du concept hybride.

Catégorie	Nom	Taille	Inférieure	Supérieure	Égalité	Unité
Minimiser	TEC	1	–	–	–	[GJ]
Variables	h_{cruise}	1	27	35	–	[kft]
	τ_b	1	1.5	3.0	–	[m^3]
	x_w	1	15	18	–	[m]
	S_w	1	100	130	–	[m^2]
	AR_w	1	9	12	–	–
	$\Lambda_{25,w}$	1	20	45	–	[°]
	S_{HT}	1	20	80	–	[m^2]
	AR_{HT}	1	3	5	–	–
	$\Lambda_{25,HT}$	1	25	45	–	[°]
	S_{VT}	1	20	50	–	[m^2]
	AR_{VT}	1	1	2	–	–
	$\Lambda_{25,VT}$	1	30	45	–	[°]
Total variables	–	12	–	–	–	–
Contraintes	b_w	1	–	36	–	[m]
	$MFW - m_f$	1	0	–	–	[kg]
	$C_{L_{max}} - C_{L_{app}}$	1	0	–	–	–
	SoC	1	0.2	–	–	–
	$P_b - P_{takeoff}$	1	0	–	–	[W]
	TOFL	1	–	2 200	–	[m]
	SM	1	0.05	0.10	–	–
	$\mathcal{M}$	1	–	–	0	[N/m]
	$\mathcal{N}$	1	–	–	0	[N/m]
	$C_{L_{cruise}} - C_{L_{opt}}$	1	–	–	0	–
Total contraintes	11	–	–	–	–	–

blèmes en Section 4.3, on a donc fait le choix de ne pas l'inclure dans les méthodes d'optimisation à tester. Toutes les implémentations des méthodes restent les mêmes que dans la Section 4.3.1. Pour SEGO-UTB, on utilise la mise à jour du seuil de doute qui donne les meilleurs résultats sur l'ensemble des 29 problèmes, c.-à-d. SEGO-UTB (τ : I-Exp-1) (voir Section 4.2). On utilise également le noyau de covariance gaussien pour SEGO et SEGO-UTB car il a montré les meilleurs performances de convergence (voir Section 4.2). Pour finir et afin de réduire le biais introduit par le choix du DoE initial, on réalise 10 optimisations par méthode à partir de 10 DoE initiaux préalablement générés et communs aux méthodes qui nécessitent ce type d'initialisation.

5.1.3 Analyse des résultats

Dans cette section, on analyse les résultats des optimisations réalisées avec SEGO-UTB, SEGO, COBYLA et NOMAD. Pour cela, on trace d'abord les courbes de convergence des différentes méthodes. Ensuite, on affiche la forme en plan obtenue par quelques méthodes d'optimisation. Enfin, on compare les différentes configurations obtenues grâce à des courbes de coordonnées parallèles.

5.1.3.1 Courbes de convergence

Dans cette section, on trace les courbes de convergence pour chacune des méthodes prises en compte en utilisant les données des 10 optimisations réalisées par méthode. Comme dans la Section 4.2.3.2, on choisit une valeur de pénalisation de $4 \cdot 10^5$ et un facteur d'écart type de 4. Pour plus d'information sur la création des courbes de convergence, on invite le lecteur à se référer à la Section 4.2.3.2.

La Figure 5.2 représente les courbes de convergence pour le problème de conception d'avion hybride. On remarque clairement que SEGO et SEGO-UTB surpassent COBYLA et NOMAD. En

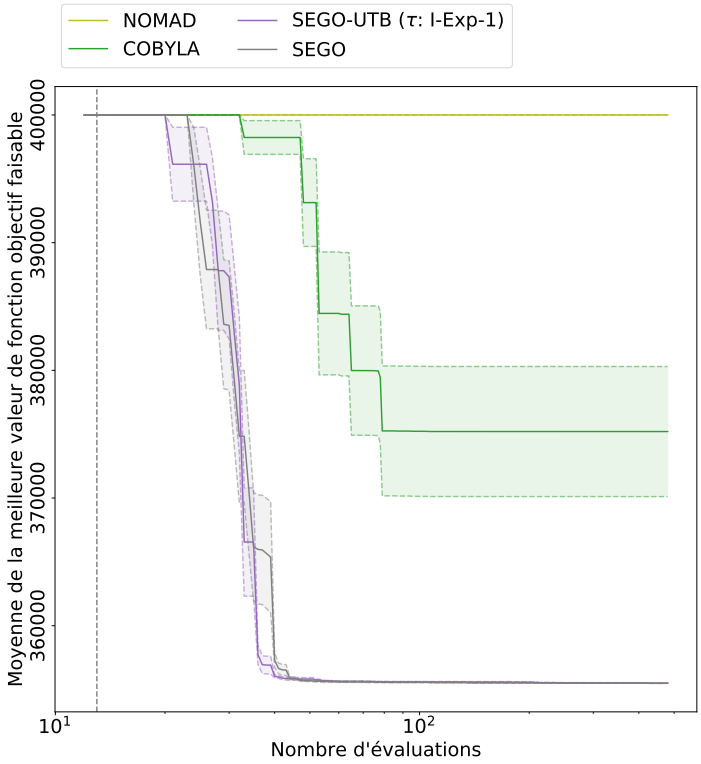


FIGURE 5.2 – Courbes de convergence du problèmes d'optimisation du concept hybride pour SEGO-UTB, SEGO, COBYLA et NOMAD. La ligne en pointillé grise verticale indique le nombre de points d'échantillonage dans le DoE initial.

réalité, on voit que NOMAD semble incapable de trouver un point faisable. Ceci est dû à la présence de contraintes d'égalité que NOMAD est incapable de prendre en compte efficacement. Le fort écart type pour COBYLA suggère que la méthode ne converge pas toujours vers le même optimum pour toutes les optimisations. Pour finir, SEGO-UTB (τ : I-Exp-1) converge systématiquement et le plus rapidement vers la meilleure valeur du problème d'optimisation découverte. Les courbes de convergence de SEGO-UTB pour les autres mises à jour du seuil de doute sont disponibles dans la Figure A.8.

5.1.3.2 Courbes de coordonnées parallèles et formes en plan

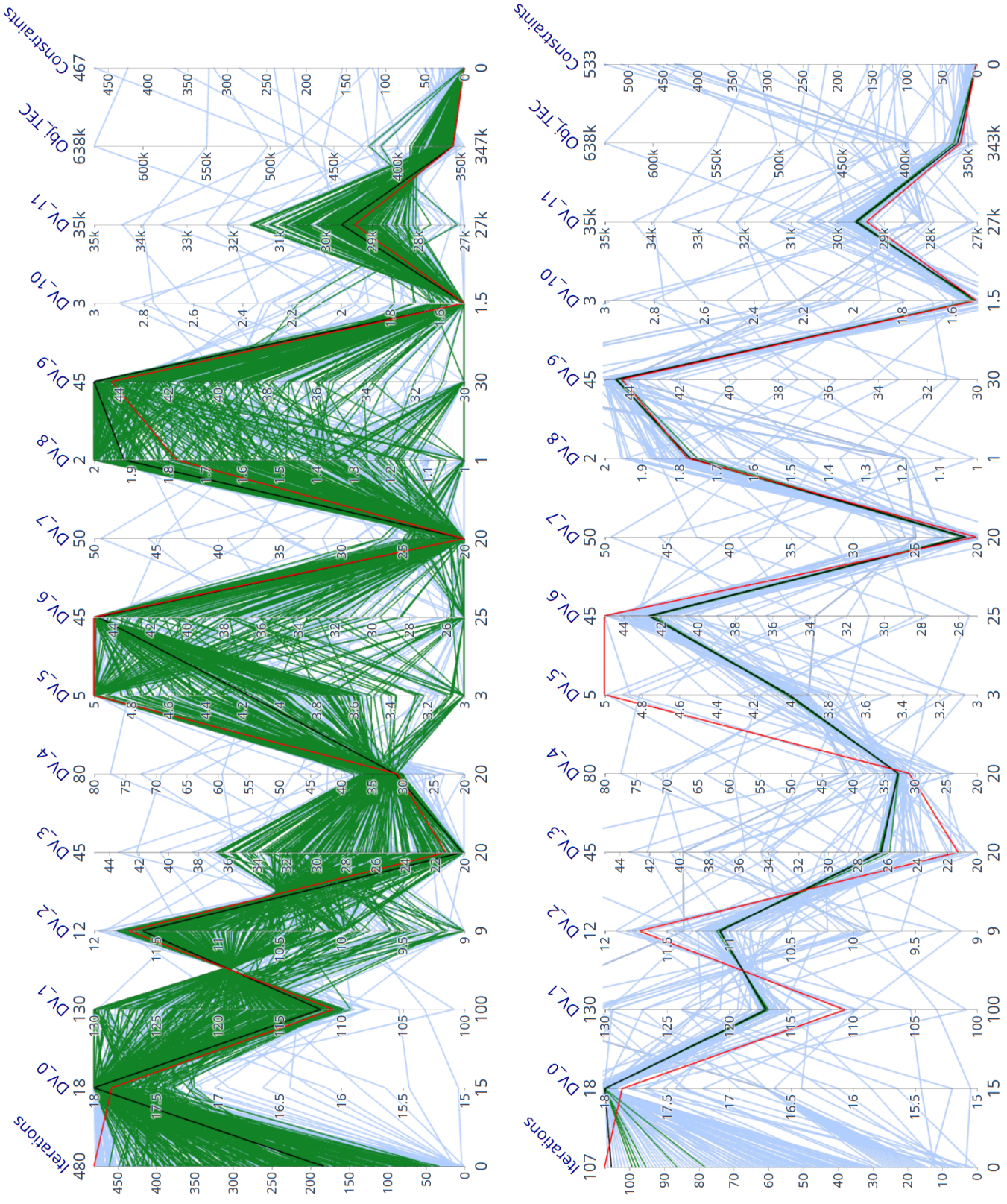
Dans ce qui suit, on utilise les courbes de coordonnées parallèles, ou parallel plots pour illustrer le comportement exploratoire des méthodes d'optimisation testées.

Avec les courbes de coordonnées parallèles, on affiche les valeurs de certaines données ciblées pendant le processus d'optimisation pour une méthode donnée (par exemple les valeurs

des variables de conception évaluées). Plus particulièrement, on choisit de représenter sur l'axe des abscisses le nom des données ciblées (le nombre d'itérations, les douze variables de conception, la fonction objectif et la violation des contraintes) et sur l'axe des ordonnées, les valeurs de ces quantités d'intérêt. En plus de cela, on utilise le rouge pour la configuration optimale (c.-à-d. la meilleure configuration trouvée par l'ensemble des méthodes), le noir pour l'optimum trouvé par la méthode considérée, le vert pour les configurations faisables évaluées et le bleu pour les configurations non faisables. A cause de la nature stochastique des tests, on construit les courbes de coordonnées parallèles avec l'optimisation médiane d'une méthode. Pour construire ces courbes de coordonnées parallèles, on procède comme suit : pour chaque optimisation réalisée, on stocke la meilleure valeur faisable (voir Section 4.2.3.3). Si aucune des optimisations ne converge vers un point faisable, on collecte la violation minimale rencontrée. L'optimisation médiane est finalement sélectionnée en se reposant sur les valeurs collectées sur les 10 optimisations.

Les Figures 5.3 et 5.4 représentent les courbes de coordonnées parallèles obtenues pour SEGO-UTB, SEGO, COBYLA et NOMAD. On remarque que SEGO-UTB et SEGO exploitent beaucoup plus l'ensemble des données que COBYLA et NOMAD. Ceci permet à SEGO-UTB et SEGO de trouver un point faisable plus rapidement que COBYLA et NOMAD. En fait, la majorité des configurations testées par SEGO et SEGO-UTB sont faisables, ce qui est montré par la quantité importante de courbes vertes, contrairement à COBYLA et NOMAD. En particulier, NOMAD ne trouve aucun point faisable. Le caractère exploratoire de SEGO-UTB se repère entre les évaluations 450 et 480 où la majorité des courbes sont bleues. Mis à part les facteurs de forme de l'empennage horizontal et vertical (DV_5 et DV_8), SEGO et SEGO-UTB convergent vers les valeurs de l'optimum. On remarque également que COBYLA évalue des points en dehors du domaine de conception. Par exemple, pour l'angle de flèche de l'aile (DV_3), on voit une courbe sortant du domaine de conception. Toutefois, la meilleure configuration découverte par COBYLA est proche de la configuration optimale ($\approx 5\%$) même si certaines variables de conception sont différentes de l'optimum notamment le facteur de forme de l'empennage horizontal. En réalité, ces variables sont de faible importance [116]. En effet, la variable qui contrôle l'optimisation est le volume de batteries dont la valeur optimale est trouvée par COBYLA, SEGO et SEGO-UTB.

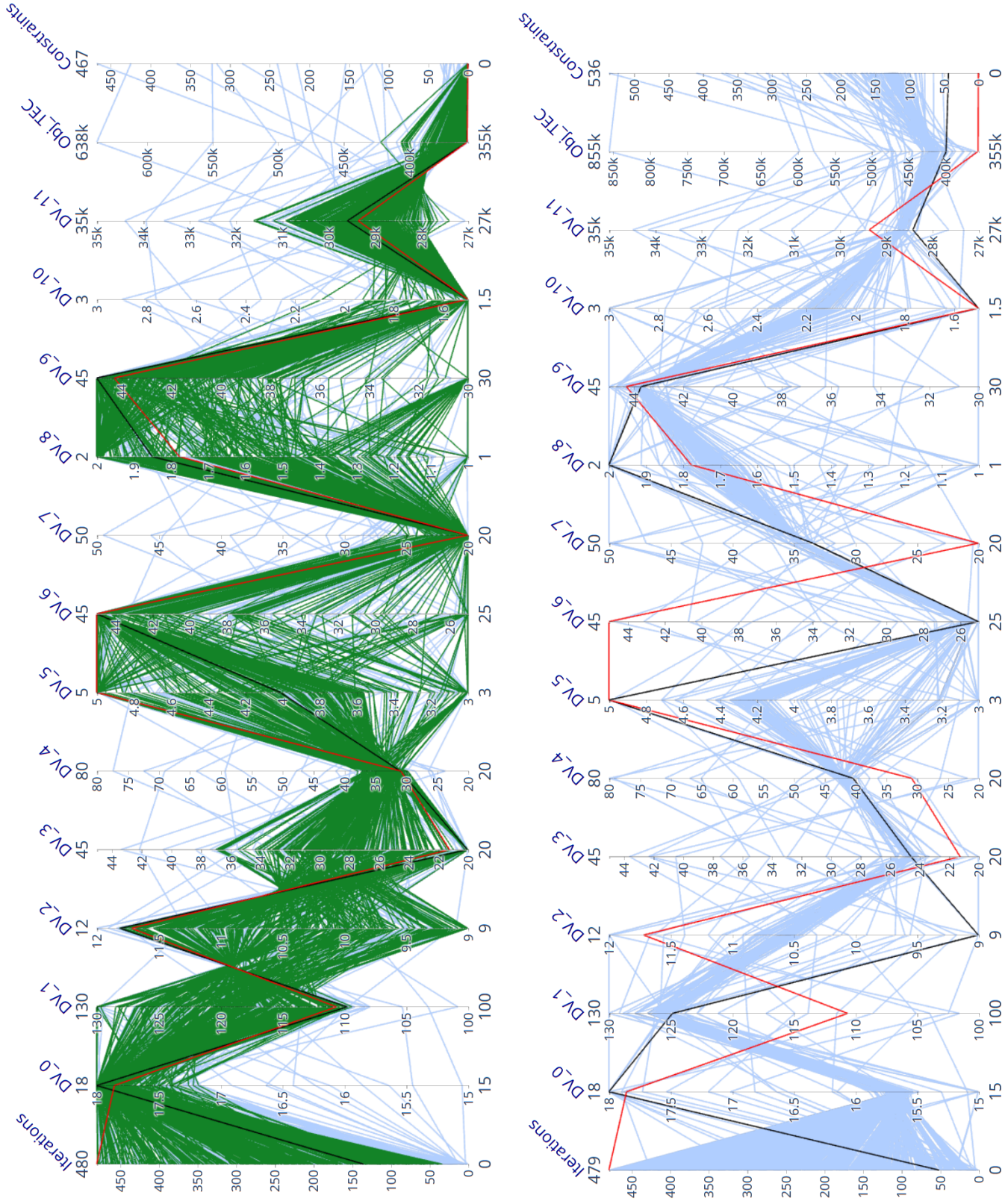
Pour finir, on commente les formes en plan de l'optimisation médiane de COBYLA et SEGO-UTB. On choisit de ne pas inclure la forme en plan de SEGO car il donne les mêmes résultats que SEGO-UTB. De plus, NOMAD n'est pas représenté puisqu'il ne fournit pas de configuration faisable. Enfin, on ajoute la forme en plan de la configuration de référence donnée par les experts. Les trois formes en plan sont finalement représentées sur la Figure 5.5. La différence entre ces formes en plan est clairement observable. On note que les solutions de SEGO-UTB et COBYLA ont un angle de flèche de l'aile plus faible que la configuration de référence. Cette réduction d'angle de flèche est suggérée par les livres de conception [104] afin de réduire les effets de compressibilité. Malgré le régime transsonique, avec un nombre de Mach de 0.78, la traînée d'onde ne diverge pas encore et l'optimum se trouve dans une zone de faible angle de flèche ce qui améliore l'aérodynamique. Le minimum découvert par COBYLA correspond à la meilleure solution aérodynamique parmi les trois formes en plan représentées. Cependant, un angle de flèche plus grand tend à faire augmenter la masse de l'aile ce qui est pénalisant pour les performances globales de l'avion. Au contraire, le minimum choisi par SEGO-UTB est un équilibre entre les performances aérodynamiques et la réduction de masse de l'avion. Par conséquent, les empennages sont plus petits afin



(a) SEGO-UTB (τ : I-Exp-1).

(b) COBYLA.

FIGURE 5.3 – Courbes de coordonnées parallèles utilisant l'optimisation médiane du concept hybride pour SEGO-UTB et COBYLA. En bleu : les configurations non faisables ; en vert ; les configurations faisables ; en noir : l'optimum de la méthode considérée ; en rouge : l'optimum.



(a) SEGO.

(b) NOMAD.

FIGURE 5.4 – Courbes de coordonnées parallèles utilisant l'optimisation médiane du concept hybride pour SEGO et NOMAD. En bleu : les configurations non faisables ; en vert ; les configurations faisables ; en noir : l'optimum de la méthode considérée ; en rouge : l'optimum.

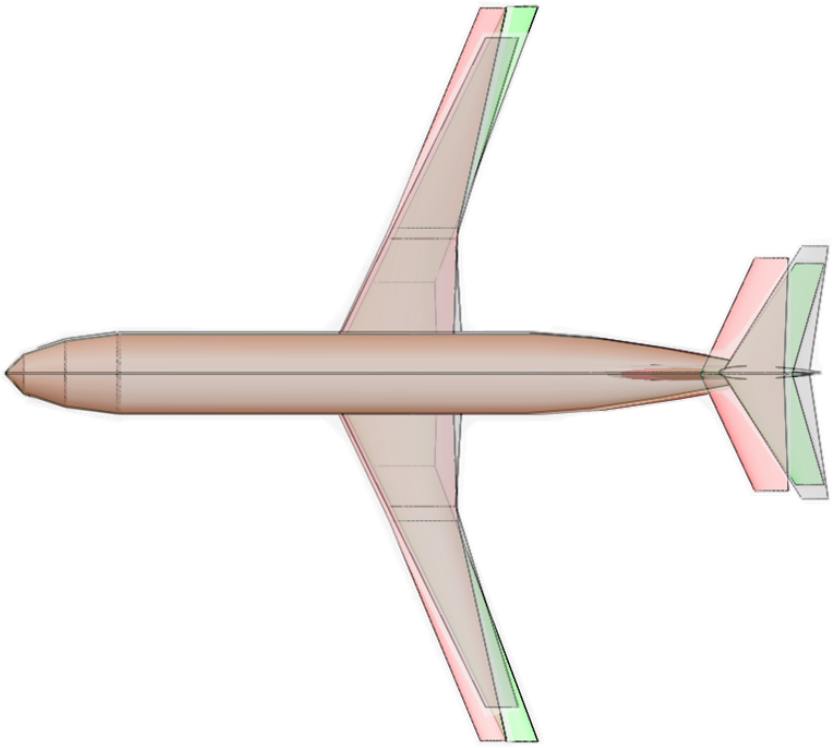


FIGURE 5.5 – Une des formes en plan obtenue par SEGO-UTB (en rouge), COBYLA (en vert) et la configuration de référence (en gris) du concept hybride.

de prendre en compte la réduction de la masse et de la surface de l'aile.

5.1.4 Conclusion

Dans cette section, on a comparé les performances de SEGO et SEGO-UTB avec les méthodes d'optimisation présentées dans la Section 4.3.1.

Pour cela, on a d'abord introduit la configuration d'avion étudiée. C'est une configuration hybride comportant des moteurs électriques distribués le long des ailes. L'ensemble des dépendances entre les disciplines nécessaires à la conception est géré par l'outil FAST. L'objectif est ensuite de concevoir cet avion pour qu'il consomme le moins d'énergie possible. Mathématiquement parlant, cet objectif se traduit par un problème MDO comprenant 12 variables de conception, 8 contraintes d'inégalité et 3 contraintes d'égalité.

Ensuite, on a réalisé l'optimisation de cette configuration avec SEGO, SEGO-UTB, NOMAD et COBYLA. On n'a pas utilisé les méthodes PESC [51], SUR [83], ALBO [84] et EFI [112] au vu du temps de calcul CPU prohibitif qu'elles nécessitent pour converger (voir Tableau 4.4).

Par la suite, on a choisi de comparer les résultats grâce à trois outils : les courbes de convergence, les courbes de coordonnées parallèles et les formes en plan.

1. Les courbes de convergence ont montré que SEGO et SEGO-UTB convergent systématiquement vers l'optimum global du problème. Au contraire, NOMAD et COBYLA convergent difficilement à chaque fois. En particulier, NOMAD ne réussit pas à trouver un avion faisable. Ceci est dû à la présence de contraintes d'égalité que NOMAD est incapable de prendre en compte efficacement.

2. Les courbes de coordonnées parallèles ont révélés que certaines variables sont de faible importance vis à vis de l'optimum ; par exemple le facteur de forme de l'empennage horizontal. De plus, on a remarqué que lorsque COBYLA converge, il converge vers l'optimum global du problème.
3. Les formes en plan ont offert une autre perspective vis à vis des résultats. On a constaté que SEGO et SEGO-UTB convergent vers un compromis entre l'aérodynamique et la résistance structurelle de l'avion alors que COBYLA converge vers un optimum aérodynamique.

Dans la section suivante, on s'intéresse à une application industrielle de SEGO et ses autres variantes développées à l'ONERA pour évaluer leur capacité sur un problème industriel.

5.2 Conception d'une configuration avion de recherche à Bombardier Aviation

A Bombardier Aviation, la conception avion avant-projet repose sur un processus MDO qui n'a pas cessé d'évoluer au cours de la dernière décennie. Ceci assure que les nouveaux programmes de conception produisent un avion efficace et compétitif [85]. Pour réussir, Bombardier Aviation a développé un processus multi-niveaux, multi-fidélité et multi-disciplinaire pour affiner la configuration d'un avion du début de la conception jusqu'à la conception détaillée [88].

Le premier élément de ce processus est l'optimisation de conception multi-disciplinaire conceptuelle, ou conceptual multi-disciplinary design optimization (CMDO). Le principal objectif du processus CMDO est d'explorer l'espace de conception tout en respectant des contraintes définies grâce aux programmes de développement avion précédents et à des nécessités de commercialisation. Grâce à ce processus d'optimisation, on est capable d'obtenir une configuration avion qui considère le dimensionnement structurel des principaux composants et systèmes. Cet avion conceptuel génère ensuite des exigences de conception qui sont validées par des processus de plus haute-fidélité : les processus d'optimisation de conception multi-disciplinaire préliminaire, ou preliminary multi-disciplinary design optimization (PMDO) et d'optimisation de conception multi-disciplinaire détaillée, ou detailed multi-disciplinary design optimization (DMDO). Le deuxième élément, correspondant au processus PMDO, se concentre sur l'analyse aérodynamique et structurelle de l'aile. L'objectif principal est ici de raffiner le dimensionnement de l'aile afin de valider certaines entrées du processus CMDO ; par exemple les polaires de traînée et le poids de l'aile. Avec ce processus de plus haute-fidélité, on produit une version détaillée des géométries internes et externes de l'aile, ainsi que sa structure initiale. On souligne qu'on ne traite pas le processus DMDO qui n'est pas encore automatisé comme le sont les processus PMDO et CMDO.

Les principales entrées pour initier la conception d'une nouvelle configuration avion sont des ensembles d'exigences provenant de l'équipe de commercialisation ; par exemple le nombre de Mach en croisière, le rayon d'action de l'avion, les performances et l'opérabilité en aéroports critiques. Chacune de ces exigences, ou contraintes, nécessite une convergence entre la CMDO et la PMDO afin de produire une configuration robuste. Toutefois, le nombre d'exigences à analyser est grand ce qui nécessite un temps de calcul CPU important.

Le processus CMDO se compose d'un ensemble de nombreux modules de basse fidélité représentant les disciplines aéronautiques. Chacun de ces modules utilise ses propres variables

de conception ce qui augmente fortement la dimension du problème d'optimisation. Par conséquent, l'exploration du domaine de conception devient coûteuse en temps de calcul CPU. Ainsi, un processus d'optimisation peut prendre plusieurs jours à être réalisé. De même, le processus PMDO est formé d'un grand nombre de variables de conception nécessaires à la définition de la géométrie de l'aile et du modèle aéro-structure. Le processus complet prend dans ce cas plusieurs semaines. Ces temps de calcul prohibitifs poussent donc à l'utilisation de méthodes d'optimisation peu gourmandes en temps de calcul. C'est pourquoi dans cette section, on évalue les performances de la toolbox SEGOMOE [9], comprenant les algorithmes SEGO [108], SEGOMOE [9], SEGO-UTB et SEGOMOE utilisant UTB (SEGOMOE-UTB), sur ce problème MDO de conception avion avant-projet.

Pour cela, on détaille d'abord les processus CMDO et PMDO. Ensuite, on introduit la configuration avion de recherche de Bombardier, ou Bombardier research aircraft configuration (BRAC) utilisée pour évaluer la toolbox SEGOMOE. Puis, on présente les algorithmes d'optimisation, qui sont classiquement utilisés à Bombardier Aviation pour concevoir les nouvelles configurations, que l'on compare avec la toolbox SEGOMOE. On souligne que ce travail s'inscrit dans un accord de collaboration sur la MDO entre l'ONERA, l'ISAE-SUPAERO et Bombardier Aviation. L'évaluation de la toolbox SEGOMOE sur les processus CMDO et PMDO est réalisée lors d'une mobilité internationale à Bombardier Aviation à Montréal (CA) de Mai 2019 à Août 2019. De plus, cette étude est possible grâce à la collaboration active de Hugo Gagnon [3], Stéphane Dufresne [4] et Ian Chittick [5].

5.2.1 Une optimisation industrielle multi-niveaux, multi-fidélité et multi-disciplinaire

Dans cette section, on introduit les deux premiers niveaux du processus d'optimisation multi-disciplinaire développés à Bombardier Aviation. On présente d'abord le processus CMDO. Ensuite, le processus PMDO est décrit.

5.2.1.1 Cadre de l'optimisation multi-disciplinaire conceptuelle, ou conceptual multi-disciplinary design optimisation framework

Le processus CMDO est le premier niveau du déroulement multi-niveaux de la MDO à Bombardier Aviation. Il considère les principales disciplines aéronautiques comme l'aérodynamique, la structure, les systèmes, le poids et l'équilibrage, les performances, la stabilité et les coûts. A cause de ce nombre important de disciplines, on considère un niveau de fidélité faible correspondant à des méthodes simples et empiriques ou reposant sur la physique. Par exemple, pour le module aérodynamique, une méthode interne à Bombardier Aviation est utilisée pour évaluer le coefficient de portance maximum $C_{L_{max}}$ tandis qu'une méthode de vortex-lattice [35] couplée à un code de calcul aérodynamique 2D est adoptée pour calculer la traînée à haute vitesse des surfaces de contrôle. De la même manière, d'autres disciplines peuvent combiner plusieurs analyses de fidélité différente afin d'obtenir la précision et la vitesse de calcul désirées. On utilise le processus CMDO décrit ci-dessus pour explorer l'espace de conception, optimiser des exigences de commercialisation, valider des études de cas, valider l'insertion de nouvelles technologies, sélectionner des configurations avion prometteuses, et définir des cibles de performance [88].

3. Engineering Professional, Advanced Design, hugo.gagnon@aero.Bombardier.com.
4. Senior Engineering Specialist, Specialized Aircraft, stephane.dufresne@aero.Bombardier.com.
5. Engineering Specialist, Advanced Aerodynamics, ian.chittick@aero.Bombardier.com.

Les variables de conception classiquement utilisées dans le processus CMDO sont la surface de l'aile, la forme en plan de l'aile ainsi qu'un facteur d'échelle moteur dans le cas d'une architecture à motorisation choisie. On peut aussi prendre en compte le ratio entre la corde et l'épaisseur de l'aile ainsi que la position des longerons le long de la corde pour différentes localisations le long de l'envergure de l'aile. Les contraintes du processus d'optimisation sont définies à partir des performances de l'avion comme la longueur de piste équivalente, ou balanced field length (BFL), la vitesse d'approche (V_{ref}), l'altitude initiale de croisière, ou initial cruise altitude (ICA), le rayon d'action pour différentes missions, et la capacité à atterrir et à décoller sur des aéroports critiques. Des considérations systèmes et géométriques peuvent aussi être prises en compte comme l'intégration du train d'atterrissage ou la taille de la corde en bout d'aile. Différentes fonctions objectifs peuvent être utilisées ; par exemple la masse maximale au décollage, ou maximum take-off weight (MTOW), le coût, l'impact climatique, ou une combinaison des trois. Dans ce qui suit, seulement la MTOW est employée puisqu'elle combine les effets relatifs au coût et à l'impact climatique.

Le processus CMDO est implémenté dans Isight [132] qui est un logiciel commercial d'intégration de processus de conception. Il est distribué par Dassault Systèmes.

Pour finir, la configuration obtenue avec le processus CMDO est donnée en entrée au processus PMDO. Le processus PMDO est introduit dans la section suivante.

5.2.1.2 Cadre de l'optimisation multi-disciplinaire préliminaire, ou preliminary multi-disciplinary design optimisation framework

Le processus PMDO est le deuxième niveau du déroulement multi-niveaux de la MDO à Bombardier Aviation. Comparé à CMDO, on réduit la portée de la conception au couplage entre l'aérodynamique et la structure en considérant des outils de plus haute-fidélité. L'objectif est d'obtenir une ligne de moule externe, ou outer mold lines (OML) de l'aile qui est aérodynamiquement efficace pour des vols à haute et à basse vitesses tout en assurant de bonnes performances pour des conditions critiques hors conception. De plus, l'OML doit être faisable structurellement, c.-à-d. avec un poids faible et respectant des contraintes de volume.

PIPERNI et al. [88] ont décrit le processus PMDO en détail. Bien que de nombreux modules soient inchangés, le processus PMDO a continué d'évoluer. On explique, dans ce manuscrit, les grandes lignes de cette procédure de conception qui sont nécessaires à la compréhension global du travail. Les surfaces de l'avion sont mises à jour à partir d'un modèle paramétrique utilisant l'outil de conception assistée tridimensionnelle interactive appliquée version 5 (CATIA v5) développé par Dassault Systèmes. Le générateur de maillage reposant sur CATIA développé à Bombardier (MBGRID) [87] et le full aircraft Navier-Stokes code (FANSC) sont utilisés pour calculer l'aérodynamique de l'avion pour différentes hautes vitesses et chargements statiques. Les chargements extérieurs sont ensuite transférés au maillage structurel de l'aile généré par l'Automatic Wing Structural Optimization Module (AWSOM) qui minimise le poids du caisson de voilure principale en respectant diverses marges et contraintes de sécurité. Pour les basses vitesses, on utilise le critère semi-empirique de Valarezo [131] avec VSAERO [68]. Pour finir, la traînée à haute vitesse et le poids de la structure sont reliés à travers une équation fractionnaire décrite dans le travail de PIPERNI et al. [86].

L'architecture du processus PMDO a fortement changé avec la maturation de la méthode d'optimisation reposant sur l'adjoint développé à Bombardier (BOOST) [102, 114]. La méthode ad-

jointe est un outil efficace pour calculer les dérivées nécessaires à l'optimisation de la forme aérodynamique. Cette optimisation utilise des simulations très longues en temps CPU et considère un grand nombre de variables de conception. Pour intégrer l'adjoint aérodynamique dans un environnement multi-disciplinaire, un processus hybride adjoint-MDO a été développé [29] comme le montre la Figure 5.6. D'abord, une optimisation aéro-structure est réalisée dans Isight (voir

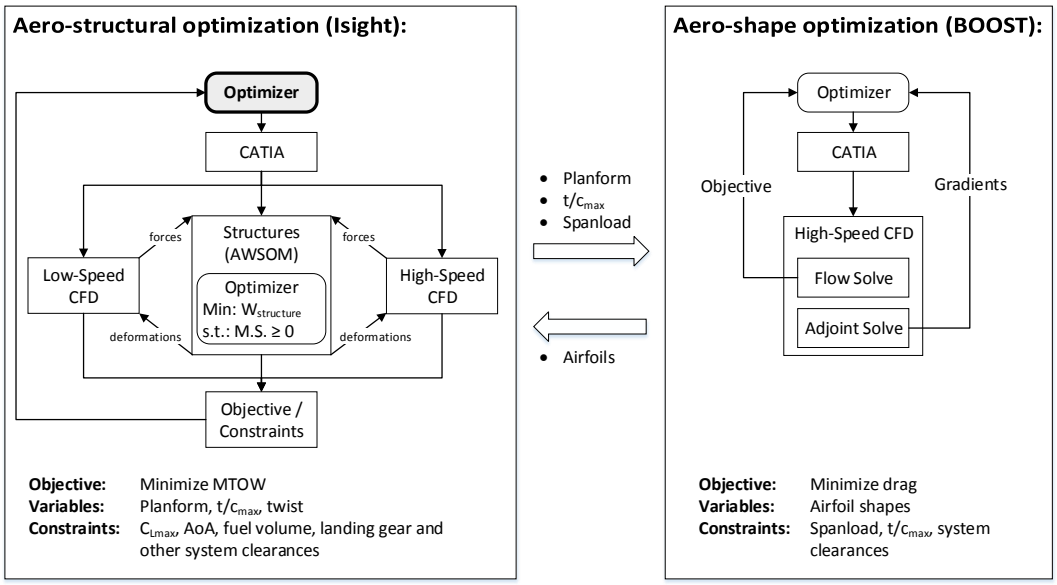


FIGURE 5.6 – Schéma simplifié du processus hybride adjoint-MDO

schéma de gauche de la Figure 5.6) en faisant varier la forme en plan de l'aile, le vrillage de l'aile et la distribution de l'épaisseur. Ensuite, une optimisation de la forme aérodynamique est effectuée avec BOOST, voir schéma de droite de la Figure 5.6, où la traînée est minimisée en faisant varier la forme des surfaces portantes et le chargement de l'aile. Ces deux étapes sont répétées jusqu'à convergence. On s'intéresse ici à la méthode d'optimisation principale utilisée pour contrôler l'optimisation aéro-structure (en gras dans le schéma de gauche de la Figure 5.6). Celle-ci est actuellement le verrou du processus PMDO.

Pour conclure cette section sur le processus multi-niveaux MDO développé à Bombardier Aviation, on a d'abord introduit le processus CMDO qui prend en compte un grand nombre de disciplines de basse fidélité. Ensuite, le résultat obtenu par le processus CMDO est donné en entrée du processus PMDO qui raffine le forme et la structure de l'aile à l'aide de modèles de plus haute fidélité. Dans la section suivante, on présente la configuration avion sur laquelle ces deux processus MDO sont appliqués.

5.2.2 Le problème d'optimisation de la configuration de recherche de Bombardier, ou Bombardier research aircraft configuration

La configuration considérée dans ce travail est représentative d'un petit avion d'affaires que l'on nomme BRAC. Elle repose sur le Challenger 300 et est développée pour les partenariats entre Bombardier Aviation et le milieu académique afin de faciliter la publication. Une représentation de la référence de la BRAC est donnée par la Figure 5.7. Dans la suite de cette section, on introduit les problèmes CMDO et PMDO liés à cette configuration.

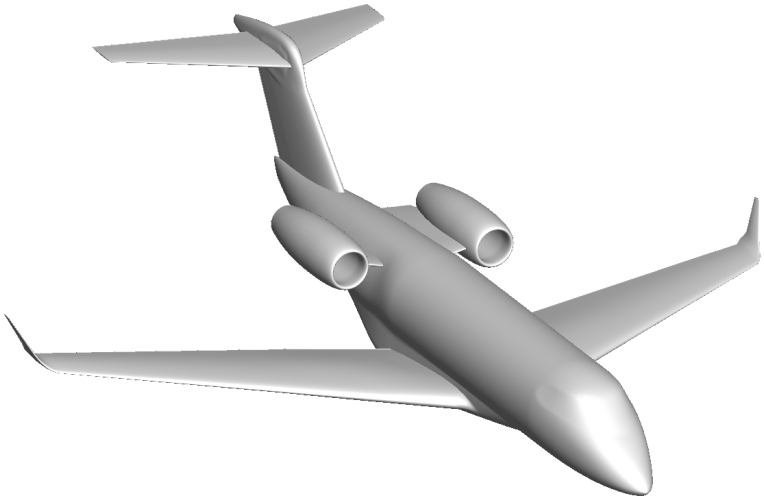


FIGURE 5.7 – La configuration de référence de la BRAC.

5.2.2.1 Le problème d'optimisation multi-disciplinaire conceptuel, ou conceptual multidisciplinary design optimisation case

Le problème CMDO de la BRAC repose sur une version modifiée des exigences de commercialisation du Challenger 300. C'est donc une configuration représentative d'un problème de conception avion industriel. Pour des raisons de confidentialité, les bornes exactes des variables de conception et des contrainte ne sont pas données ici.

Mathématiquement, on écrit le problème CMDO de la BRAC comme suit :

$$\min_{\boldsymbol{x}\in[0,1]^{12}} \left\{\text{MTOW}(\boldsymbol{x}) \text{ s.t. } \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{B}\right\}, \tag{5.2}$$

où $\boldsymbol{x} \in [0,1]^{12}$ est le vecteur contenant les variables de conception décrites dans le Tableau 5.3, $\boldsymbol{g} : \mathbb{R}^{12} \mapsto \mathbb{R}^{8}$ est le vecteur des fonctions de contrainte exposées dans le Tableau 5.4 et $\mathbf{B}$ est le vecteur des bornes associées à ces contraintes. Pour des raisons de confidentialité, $\mathbf{B}$ n'est pas donné explicitement.

TABLEAU 5.3 – Définition de l'espace de conception CMDO.

Indice Var.	Nom Var.	Symbole Var.
$\boldsymbol{x}[0]$	Facteur d'échelle du moteur	DV_0
$\boldsymbol{x}[1]$	Facteur de forme de l'aile	DV_1
$\boldsymbol{x}[2]$	Surface de l'aile	DV_2
$\boldsymbol{x}[3]$	Angle de flèche du bord de fuite intérieur de l'aile	DV_3
$\boldsymbol{x}[4-5]$	Emplacements le long de la corde du longeron arrière de l'aile	DV_4-5
$\boldsymbol{x}[6]$	Angle de flèche de l'aile	DV_6
$\boldsymbol{x}[7]$	Coefficient d'effilement de l'aile	DV_7
$\boldsymbol{x}[8-11]$	Rapports entre le maximum d'épaisseur de l'aile et la corde	DV_8-11

On souligne que le rayon d'action nominal est toujours atteint car satisfait implicitement dans le processus CMDO présenté dans la Section 5.2.1.1. De plus, la référence de la BRAC (voir Figure 5.7) n'est pas faisable dans le processus CMDO. Ceci est sans conséquence puisqu'elle n'est pas utilisée comme point initial de l'optimisation. Pour finir, pour des raisons industrielles, le pro-

TABLEAU 5.4 – Définition des contraintes CMDO.

Indice Cst.	Nom Cst.	Indice des bornes
$\boldsymbol{g}[0]$	BFL	$\mathbf{B}[0]$
$\boldsymbol{g}[1]$	ICA	$\mathbf{B}[1]$
$\boldsymbol{g}[2]$	V_{ref}	$\mathbf{B}[2]$
$\boldsymbol{g}[3]$	Excédant de carburant	$\mathbf{B}[3]$
$\boldsymbol{g}[4]$	Performance de montée	$\mathbf{B}[4]$
$\boldsymbol{g}[5]$	Rayon d'action à grande vitesse	$\mathbf{B}[5]$
$\boldsymbol{g}[6]$	Espacement du train d'atterrissage	$\mathbf{B}[6]$
$\boldsymbol{g}[7]$	Corde en bout d'aile	$\mathbf{B}[7]$

cessus CMDO de la BRAC ne doit pas dépasser plus de 8 heures de calcul.

5.2.2.2 Le problème d'optimisation multi-disciplinaire préliminaire, ou preliminary multidisciplinary design optimisation case

Comme mentionné dans la Section 5.2.1.2, on s'intéresse à l'aspect aéro-structure du processus PMDO représenté par la moitié gauche de la Figure 5.6. Comme pour le processus CMDO de la BRAC, l'objectif du problème PMDO de la BRAC est de minimiser la MTOW :

$$\min_{\boldsymbol{x}\in[0,1]^{19}} \left\{\text{MTOW}(\boldsymbol{x}) \text{ s.t. } \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{B}\right\}. \tag{5.3}$$

Les variables de conception concernent la géométrie des ailes et des ailettes. Elles sont décrites dans le Tableau 5.5. La surface de l'aile et la taille du moteur sont constantes au niveau du proces-

TABLEAU 5.5 – Définition de l'espace de conception PMDO.

Indice Var.	Nom Var.	Symbole Var.
$\boldsymbol{x}[0]$	Envergure de l'aile	DV_0
$\boldsymbol{x}[1]$	Angle de flèche du bord d'attaque de l'aile	DV_1
$\boldsymbol{x}[2]$	Emplacement de cassure de l'aile	DV_2
$\boldsymbol{x}[3]$	Corde en bout d'aile	DV_3
$\boldsymbol{x}[4-7]$	Distributions du vrillage de l'aile	DV_4-7
$\boldsymbol{x}[8-13]$	Rapports maximale entre l'épaisseur et la corde de l'aile	DV_8-13
$\boldsymbol{x}[14]$	Angle de flèche du bord de fuite intérieur de l'aile	DV_14
$\boldsymbol{x}[15]$	Angle du dièdre de l'ailette	DV_15
$\boldsymbol{x}[16]$	Envergure de l'ailette	DV_16
$\boldsymbol{x}[17-18]$	Vrillages de l'ailette	DV_17-18

sus PMDO et sont données par l'optimum de la CMDO de la BRAC. De même, l'emplacement des longerons est fixé à la valeur du résultat de la CMDO. Les contraintes imposées pour la PMDO de la BRAC sont listées dans le Tableau 5.6. La fraction de carburant disponible est le rapport entre la quantité de carburant disponible dans l'aile comparé à la quantité de carburant nécessaire pour réaliser la mission. La déformation du vrillage en bout d'aile est une contrainte dynamique qui empêche l'aile d'être trop flexible. Comme pour la CMDO de la BRAC, les bornes ne sont pas données pour des raisons de confidentialité. Finalement, la PMDO de la BRAC prend en compte 19 variables de conception et 5 contraintes ce qui est représentatif d'une application industrielle du

TABLEAU 5.6 – Définition des contraintes PMDO.

Indice Cst.	Nom Cst.	Indice des bornes
$\boldsymbol{g}[0]$	Fraction de volume disponible	$\mathbf{B}[0]$
$\boldsymbol{g}[1]$	$C_{L_{max}}$ avec train atterrissage rétracté	$\mathbf{B}[1]$
$\boldsymbol{g}[2]$	Angle d'inclinaison en croisière	$\mathbf{B}[2]$
$\boldsymbol{g}[3]$	Espacement du train d'atterrissage	$\mathbf{B}[3]$
$\boldsymbol{g}[4]$	Déformation du vrillage en bout d'aile	$\mathbf{B}[4]$

processus PMDO.

Pour conclure cette section, on a introduit les deux problèmes liés à la CMDO et à la PMDO de la BRAC. D'une part, la CMDO de la BRAC prend en compte 12 variables de conception et 8 contraintes ce qui est représentatif d'un problème industriel de CMDO. D'autres part, la PMDO de la BRAC considère 19 variables de conception et 5 contraintes ce qui est également représentatif d'un problème industriel de PMDO. Dans la section suivante, on introduit les différentes méthodes d'optimisation évaluées pour résoudre ces deux problèmes.

5.2.3 Les méthodes d'optimisation

Dans cette section, on s'intéresse aux méthodes d'optimisation qui sont utilisées pour résoudre les problèmes CMDO et PMDO de la BRAC. Ces deux problèmes peuvent être reformulés de la manière suivante :

$$\min_{\boldsymbol{x}\in\Omega}\left\{f(\boldsymbol{x}) \text{ s.t. } \boldsymbol{g}(\boldsymbol{x})\geq 0\right\}, \tag{5.4}$$

où $f:\mathbb{R}^d\mapsto\mathbb{R}$ est la fonction objectif, $\boldsymbol{g}:\mathbb{R}^d\mapsto\mathbb{R}^m$ sont les contraintes d'inégalité et $\boldsymbol{x}\in\Omega\subset\mathbb{R}^d$ est le vecteur des variables de conception. Deux des méthodes d'optimisation sont directement incluses dans le logiciel Isigth que l'on est obligé d'utiliser à cause des processus CMDO et PMDO. Pour la toolbox SEGOMOE, on a développé une interface permettant son utilisation dans Isigth. La Figure 5.8 représente cette interface, résultat d'une partie du travail réalisé lors de la mobilité internationale.

Dans cette section, on va commencer par introduire les méthodes CBO de la toolbox SEGOMOE qui sont utilisées. Ensuite, on introduit les deux méthodes implémentées dans Isight qui servent de référence.

5.2.3.1 Les méthodes d'optimisation Bayésienne

Les algorithmes SEGO, SEGOMOE, SEGO-UTB et SEGOMOE-UTB sont les quatre méthodes CBO utilisées dans cette étude, et disponibles dans la toolbox SEGOMOE. Pour SEGO-UTB et SEGOMOE-UTB, on utilise une mise à jour du seuil de doute exponentielle décroissante de type 1 (voir Figure 4.3). Pour chacune de ces méthodes CBO, on utilise la fonction d'acquisition WB2S [9] (voir (2.35)) combinée à des modèles KPLS à noyau gaussien [17] (voir Section 2.1.4.2) puisque le nombre de variables de conception est plus grand que 10. On fixe le nombre de composantes du modèle KPLS à 4 correspondant à la valeur par défaut.

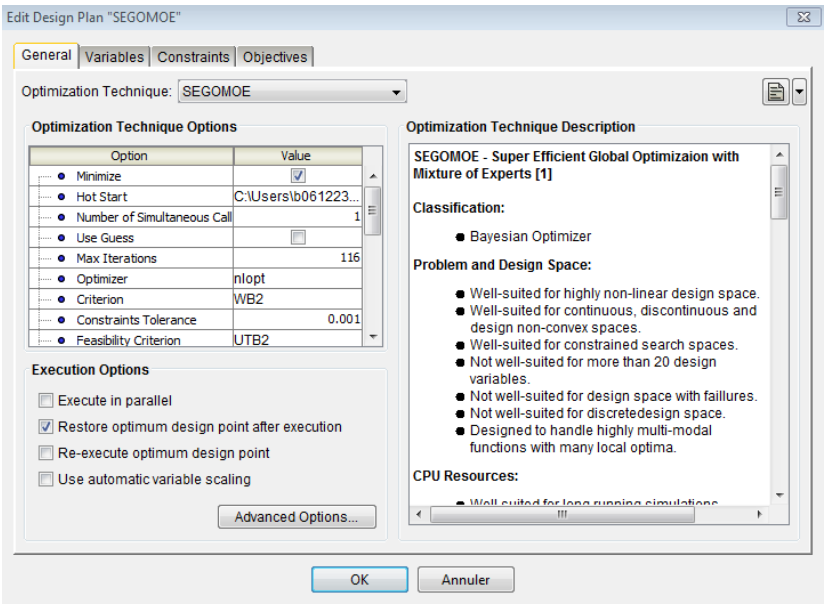


FIGURE 5.8 – Interface de la toolbox SEGOMOE dans Isight.

5.2.3.2 Les méthodes d'optimisation d'Isight

Le logiciel Isigth [132] fournit plusieurs méthodes d'optimisation déjà implémentées. Dans ce travail, on se concentre sur deux des méthodes d'optimisation sans dérivées les plus populaires : l'algorithme d'optimisation évolutionnaire, ou evolutionary optimization algorithm (Evol) [113] et une variante de l'algorithme du "chien de détection", ou variant of the Pointerdog algorithm (Pointer-2) [132].

Evol est un algorithme évolutionnaire qui fait muter le meilleur point connu en ajoutant une perturbation normalement distribuée aux variables de conception. L'écart type de la loi normale est adapté au cours de la procédure d'optimisation afin de résoudre le problème considéré avec un nombre minimal d'évaluations. Cet algorithme simple est amélioré par différentes options comme : (1) une vérification que tous les points calculés sont différents, (2) une augmentation de l'écart type quand le même point est toujours évalué, (3) une recherche dans une direction canonique choisie aléatoirement, et (4) une exécution parallèle de l'algorithme pour accélérer le processus d'optimisation.

Pointer-2 est une procédure d'optimisation reposant sur un algorithme propriétaire de Isight. Il gère plusieurs méthodes d'optimisation comme Evol [113] ; la méthode de recherche directe Hooke-Jeeves, ou (Hooke-Jeeves direct search method) [52] ; l'algorithme de programmation non-linéaire par Lagrangien quadratique, ou non-linear programming by quadratic Lagrangian (NLPQL) [109] ; l'algorithme de descente par simplex, ou downhill simplex algorithm [75] ; l'outil de système d'optimisation multi-fonctions, ou multi-function optimization system tool (MOST) [132] ; et l'algorithme d'optimisation multi-objectifs par essaims particulaires, ou multi-objective particle swarm optimization (MOPSO) [23]. Le choix de l'ensemble des méthodes d'optimisation prises en compte est géré par un système de classification utilisant les informations disponibles sur le problème. Celles-ci sont souvent données par l'utilisateur. La meilleure méthode d'optimisation est ensuite mise à jour grâce aux informations collectées tout au long du processus d'optimisation. De plus, Pointer-2 peut être employé de deux manières. D'une part, les méthodes d'optimisation et leurs options sont sélectionnées pour obtenir l'amélioration la plus forte et mener vers la solution en un

temps réduit. D'autre part, on peut chercher une solution robuste aux incertitudes ce qui engendre un processus d'optimisation plus lent avec plus d'appels au problème. Enfin, Pointer-2 réalise une optimisation reposant sur des modèles de substitution pour accélérer la convergence. En réalité, Pointer-2 est capable de résoudre un grand nombre de problèmes d'optimisation sous contraintes et ainsi permet aux non spécialistes de résoudre des problèmes d'optimisation en ne fournissant que la fonction objectif, les contraintes, les variables de conception et le temps maximal autorisé pour le processus d'optimisation.

Dans cette section, on a introduit les six méthodes d'optimisation qui sont évaluées pour la résolution des processus CMDO et PMDO associés à la BRAC. On a d'abord présenté les quatre méthodes CBO : SEGO, SEGOMOE , SEGO-UTB et SEGOMOE-UTB. Ensuite, on a décrit deux méthodes proposées dans la plate-forme commerciale Isight : Evol et Pointer-2. Dans la section suivante, on présente et commente les résultats obtenus par ces méthodes sur les deux problèmes d'optimisation industriels.

5.2.4 Évaluation des performances de la toolbox SEGOMOE sur la CMDO et la PMDO de la BRAC

Dans cette section, on cherche à évaluer les performances de la toolbox SEGOMOE sur les problèmes CMDO et PMDO de la BRAC. Pour cela, on divise le travail en deux grandes parties. D'une part, on commente les résultats obtenus pour la CMDO de la BRAC. D'autre part, on interprète les résultats de la PMDO de la BRAC.

5.2.4.1 Évaluations des performances sur la CMDO de la BRAC

On s'intéresse ici à l'évaluation des performances des six méthodes d'optimisation introduites dans la Section 5.2.3.2. Pour réaliser cette évaluation, on introduit d'abord la méthode de comparaison des méthodes d'optimisation. Ensuite, on compare les six algorithmes d'optimisation grâce à des courbes de convergence et des courbes de coordonnées parallèles. Pour finir, on commente la forme en plan de la meilleure BRAC obtenue avec le processus CMDO.

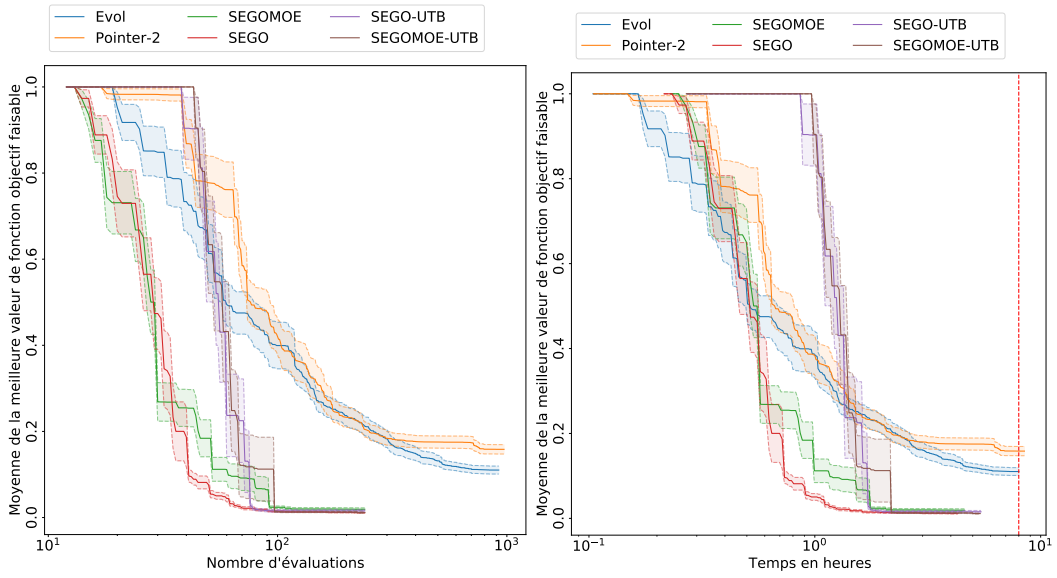
Mise en place des tests : Les méthodes d'optimisation introduites dans la Section 5.2.3.2 sont comparées grâce à la mise en place des tests suivants. On réalise 10 optimisations indépendantes pour chaque méthode en utilisant 10 DoE initiaux générés par la méthode LHS. Pour chaque optimisation, toutes les méthodes utilisent le même DoE initial. Au total, on ne génère que 10 DoE différents. La taille des DoE est fixée à $n_{start} = 13$, c.-à-d. $d+1$ où d est le nombre de variables de conception de la CMDO de la BRAC. On souligne qu'Evol et Pointer-2 ne demandent qu'un unique point initial pour commencer le processus d'optimisation. On fournit donc le point ayant la meilleure valeur faisable du DoE. Si il n'y a pas de point faisable dans le DoE, on fournit le point avec la violation minimale des contraintes. Cette méthode de mise en place des tests est comparable à celle déjà introduite en Section 4.2.3.2.

Pour les méthodes tirées de la toolbox SEGOMOE (c.-à-d. SEGOMOE, SEGO, SEGO-UTB et SEGOMOE-UTB), le nombre maximum d'évaluations est de max_it_nb = 227, c.-à-d. $20d - n_{start}$ ce qui donne un total de $20d = 240$ évaluations. Pour Evol et Pointer-2, on utilise les options utilisées classiquement pour le processus CMDO. Pour chaque optimisation, Evol réalise environ 970 évaluations ce qui prend approximativement 8 heures. Pour Pointer-2, on fixe le temps maximal

d'optimisation à 8 heures et le type du problème à *nonlinear*. De plus, on oblige Evol et Pointer-2 à ne travailler qu'avec un unique processeur. Pour finir, la meilleure solution obtenue parmi les six algorithmes d'optimisation est donnée en entrée de la PMDO de la BRAC.

Courbes de convergence : Pour évaluer les performances des six méthodes d'optimisation, on construit deux types de courbes de convergence. La première, que l'on nomme courbe de convergence en nombre d'évaluations, montre la moyenne et l'écart type de la meilleure valeur faisable de la fonction objectif pour un nombre croissant d'évaluations. La seconde, nommée courbe de convergence en temps, montre la moyenne et l'écart type de la meilleure valeur faisable de la fonction objectif pour un temps croissant. La meilleure valeur faisable de la fonction objectif se définit comme la meilleure valeur faisable de la fonction objectif si il y a au moins un point faisable. Sinon, on la fixe à une valeur de pénalisation. Ici on impose cette valeur de pénalisation à la plus haute valeur de fonction objectif faisable obtenue. Pour des raisons de confidentialité, on échelonne les courbes de convergence entre 0 et 1. On diminue d'un facteur 4 l'écart type représenté sur ces courbes pour les rendre plus lisibles. Pour plus de détails sur la création de ces courbes de convergence, le lecteur peut se reporter à la Section 4.2.3.2.

La Figure 5.9 montre que les algorithmes de la toolbox SEGOMOE surpassent clairement Evol et Pointer-2 en terme de valeur pour la fonction objectif découverte. En ce qui concerne le nombre



(a) Courbes de convergence en nombre d'évaluations.

(b) Courbes de convergence en temps.

FIGURE 5.9 – Courbes de convergence pour la CMDO de la BRAC. Dans les courbes de convergence en temps, la ligne droite verticale indique le temps maximum autorisé pour un processus d'optimisation industriel (8 heures ici).

d'évaluations, la Figure 5.9a affiche que SEGO converge le plus rapidement vers l'optimum même si SEGOMOE est le plus rapide en début d'optimisation. De plus, SEGO-UTB et SEGOMOE-UTB ne convergent pas aussi rapidement que SEGO et SEGOMOE à cause de leur caractère plus exploratoire. En s'intéressant au temps d'optimisation, la Figure 5.9b révèle que les méthodes de la toolbox SEGOMOE convergent en approximativement 2 heures alors que Evol et Pointer-2 ne convergent pas en moins de 8 heures. Cependant, le temps nécessaire à la convergence de Evol et Pointer-2 peut être réduit en utilisant l'option de calculs parallèles. Pour plus d'information sur les

résultats liés aux différentes mises à jour du seuil de doute pour SEGO-UTB et SEGOMOE-UTB, le lecteur est invité à se reporter à la Figure A.9.

Pour conclure, ces courbes de convergence montrent que les algorithmes de la toolbox SEGOMOE donnent de meilleurs résultats en un temps de calcul plus court que Pointer-2 et Evol. On remarque également que l'utilisation du critère de faisabilité UTB semble ne pas apporter d'amélioration à SEGO et SEGOMOE. Ceci suggère que l'exploration du domaine de conception n'est pas nécessaire pour découvrir la zone faisable contenant l'optimum.

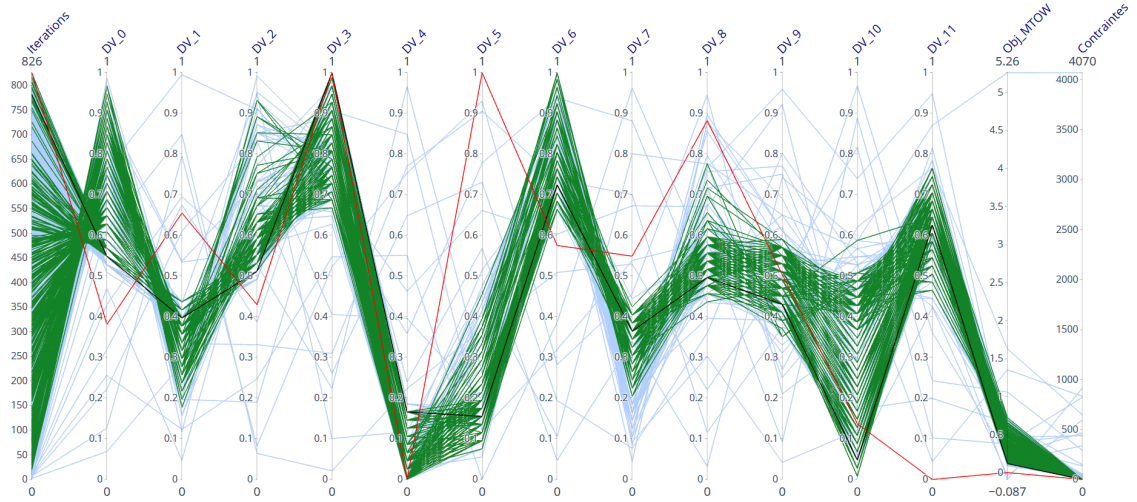
Courbes de coordonnées parallèles : Les courbes de coordonnées parallèles utilisées dans cette section sont construites de la même manière que dans la Section 5.1.3.2. On y trace les valeurs du nombre d'évaluations, des variables de conception, de la fonction objectif et de la violation des contraintes de l'optimisation médiane (définie dans la Section 5.1.3.2). Les couleurs utilisées sont les mêmes que dans la Figure 5.4. La meilleure configuration faisable obtenue est découverte par SEGO.

Les Figures 5.10 et 5.11 affichent les courbes de coordonnées parallèles de l'optimisation médiane de SEGO, SEGO-UTB, SEGOMOE, SEGOMOE-UTB, Evol et Pointer-2. Premièrement, on constate que tous les algorithmes de la toolbox SEGOMOE convergent vers une configuration donnant une valeur de fonction objectif similaire à la meilleure configuration trouvée par SEGO. Au contrainte, Evol et Pointer-2 ne convergent pas vers cette valeur comme indiqué par les Figures 5.10a et 5.10b. De plus, Evol n'est pas capable d'explorer le domaine complet comme le montrent les lignes vertes regroupées en Figure 5.10a. Deuxièmement, on remarque que les algorithmes de la toolbox SEGOMOE ne convergent pas tous vers la même configuration bien que la valeur de la fonction objectif soit la même. En effet, DV_8 et DV_9, c.-à-d. les rapports entre le maximum d'épaisseur de l'aile et la corde, sont différents de l'optimum. Ceci peut être dû à la présence soit d'un minimum local ou soit d'une variable de conception inactive, c.-à-d. que la fonction objective ne change peu ou pas si on modifie cette variable de conception. Du point de vue des experts de Bombardier Aviation, ce phénomène est provoqué par des variables de conception inactives. Pour finir, on constate que SEGOMOE-UTB et SEGO-UTB explorent plus le domaine non faisable que SEGO et SEGOMOE (voir Figures 5.10 et 5.11). En effet, le nombre de courbes bleues est beaucoup plus présent pour SEGO-UTB et SEGOMOE-UTB. Ceci justifie la lenteur de convergence de SEGO-UTB et SEGOMOE-UTB observée en début d'optimisation sur la Figure 5.9b.

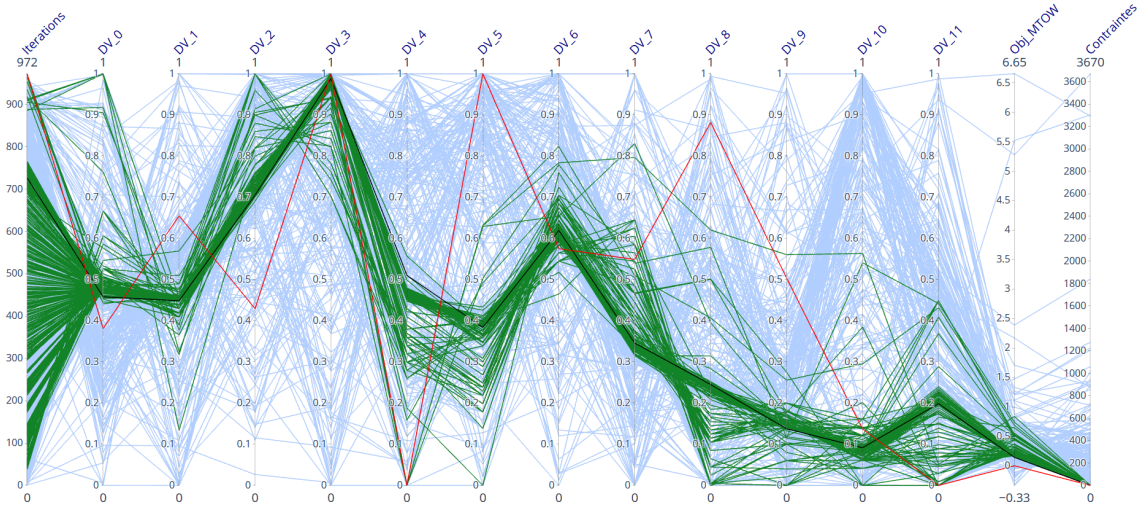
Pour conclure sur les courbes de coordonnées parallèles, on remarque que certaines variables de conception sont inactives comme DV_8 et DV_9. De plus, on constate que SEGO-UTB et SEGOMOE-UTB explorent plus le domaine de conception que SEGO et SEGOMOE

Forme en plan de la configuration optimale : La Figure 5.12 montre les formes en plan de la configuration de référence déjà présentée par la Figure 5.7 et de la configuration optimale découverte par SEGO. En réalité, cette image ne compare que la forme en plan des ailes bien qu'on ait également réalisé l'optimisation du moteur. On constate que l'envergure des ailes est légèrement plus grande pour apporter de meilleures performances aérodynamiques. De plus, l'angle de flèche de l'aile est aussi augmenté afin d'apporter une meilleure performance à haute vitesse. En effet, la BRAC s'inspire du Challenger 300 qui est un avion d'affaires.

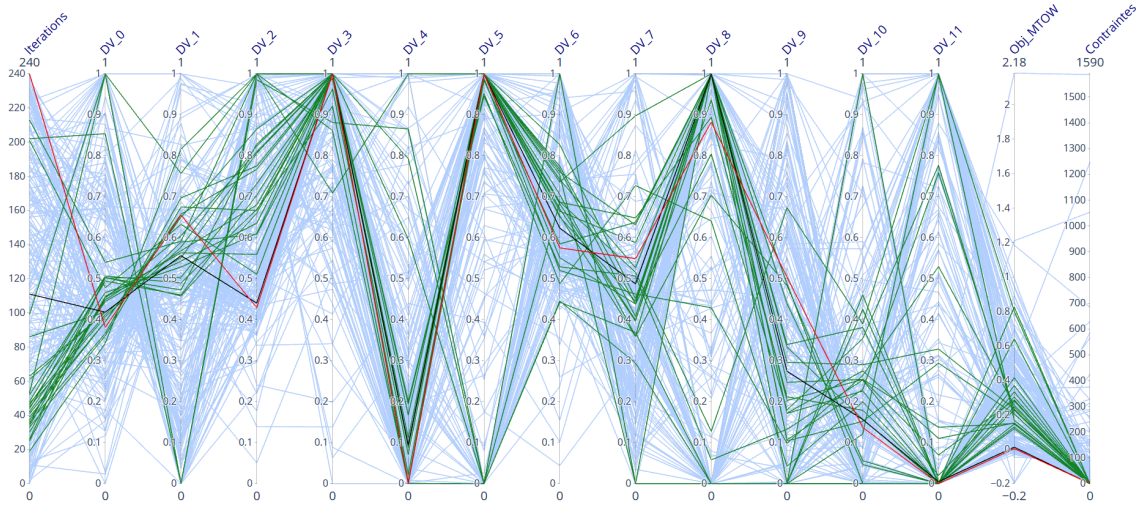
Pour conclure sur l'évaluation de la toolbox SEGOMOE sur le processus CMDO, on a constaté que l'ensemble des algorithmes de la toolbox est capable de converger vers une configuration



(a) Evol.

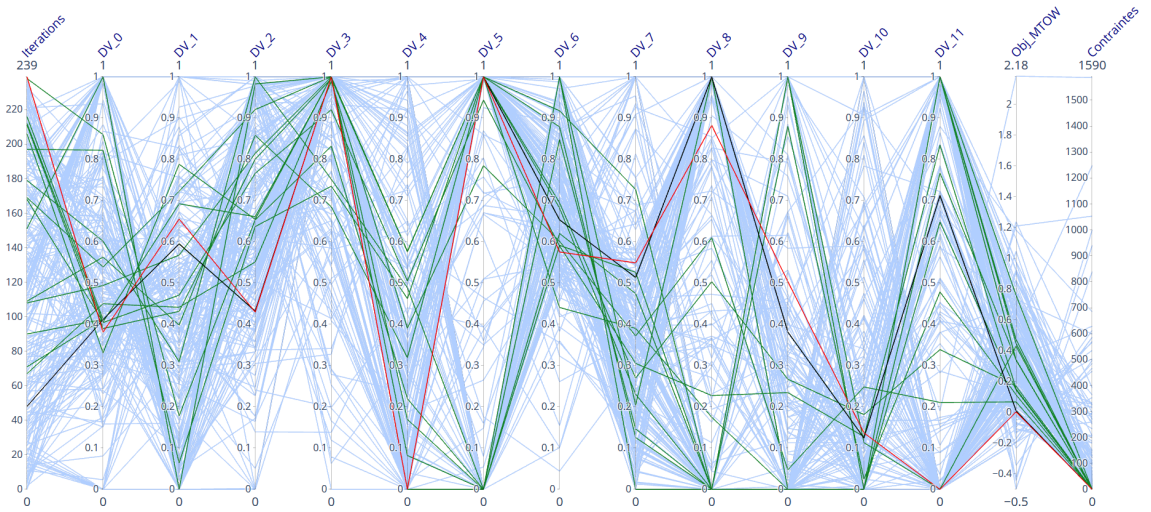


(b) Pointer-2.

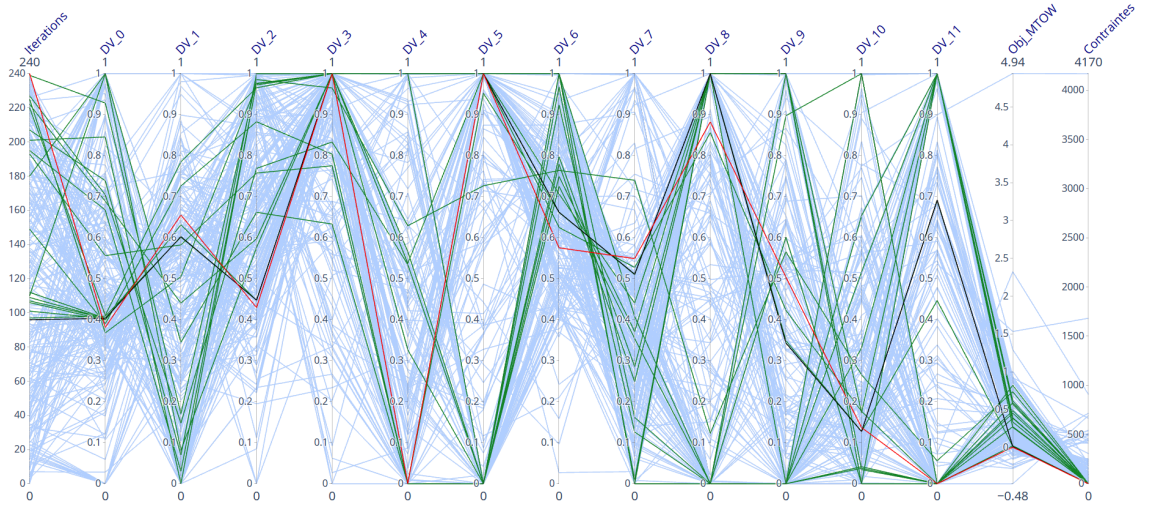


(c) SEGO.

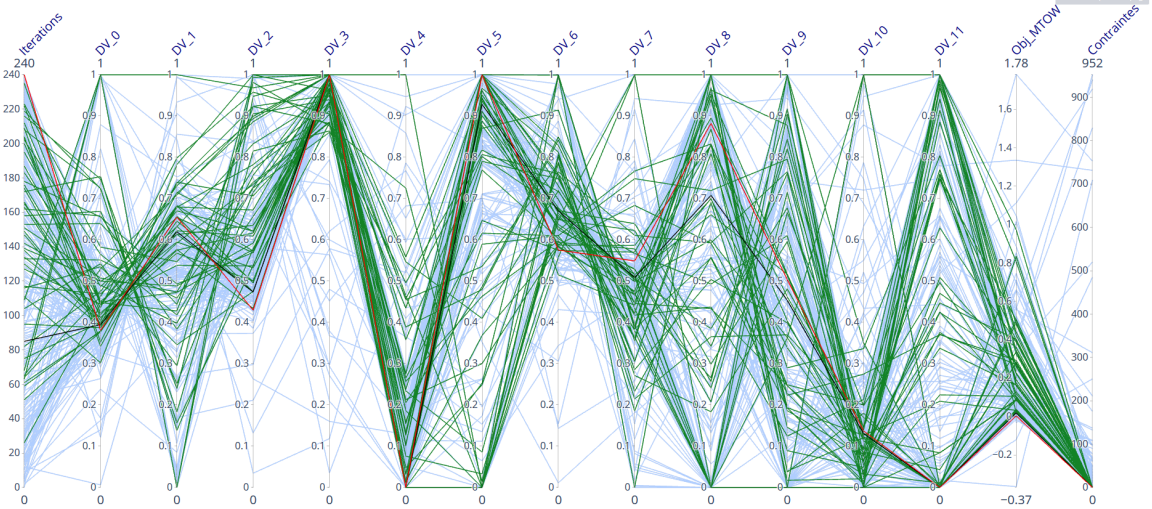
FIGURE 5.10 – Courbes de coordonnées parallèles de l'optimisation médiane de la CMDO de la BRAC pour Evol, Pointer-2 et SEGO. En bleu, les configurations non faisables ; en vert, les configurations faisables ; en noir, l'optimum de la méthode considérée ; en rouge, la solution optimale du problème.



(a) SEGO-UTB.



(b) SEGOMOE-UTB.



(c) SEGOMOE.

FIGURE 5.11 – Courbes de coordonnées parallèles de l'optimisation médiane de la CMDO de la BRAC pour SEGOMOE, SEGO-UTB et SEGOMOE-UTB. En bleu, les configurations non faisables ; en vert, les configurations faisables ; en noir, l'optimum de la méthode considérée ; en rouge, la solution optimale du problème.

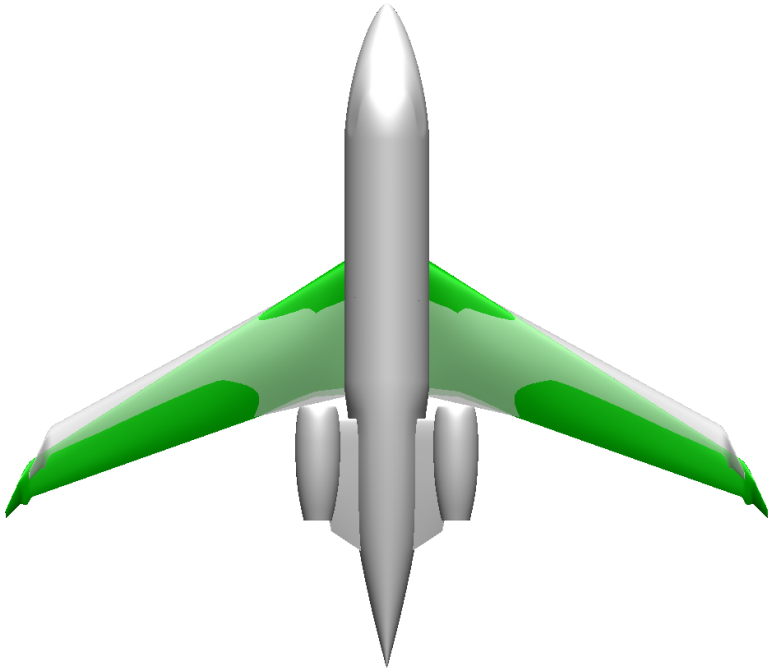


FIGURE 5.12 – Configuration optimale découverte par SEGO (en vert) et la configuration de référence (en gris) de la BRAC.

donnant une MTOW la plus faible. Au contraire, Evol et Pointer-2 convergent difficilement dans les 8 heures imparties. De plus, on a constaté que certaines variables de conception ne sont pas actives, c.-à-d. qu'une modification de leur valeur ne modifie pas la valeur de la fonction objectif. Ces variables sont les rapports entre le maximum d'épaisseur de l'aile et la corde (DV_8 et DV_9). Pour finir, on a représenté la forme en plan des ailes de la configuration optimale découverte par SEGO.

5.2.4.2 Évaluations des performances sur la PMDO de la BRAC

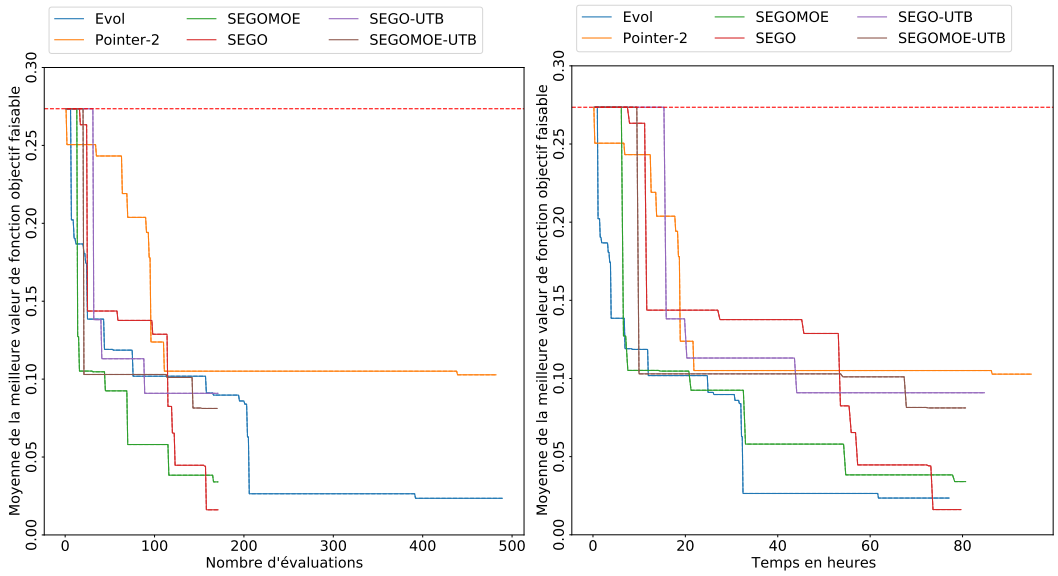
On cherche ici à évaluer les performances de la toolbox SEGOMOE sur le problème de CMDO de la BRAC. Pour cela, on va d'abord introduire la méthode de comparaison des différentes méthodes d'optimisation. Ensuite, on compare les six algorithmes d'optimisation avec des courbes de convergence et des courbes de coordonnées parallèles. Pour finir, on commente la forme en plan de la meilleure BRAC obtenue avec le processus PMDO.

Plan de test : Les six méthodes d'optimisation sont comparées cette fois sur le problème PMDO de la BRAC. A cause du temps nécessaire pour évaluer le problème (environ 25 min sur 256 processeurs), on ne réalise qu'une seule optimisation. Chacune des méthodes d'optimisation est lancée avec le même DoE initial de $n_{start} = 20$ points d'échantillonnage générés par la méthode LHS. A ce DoE, on ajoute également le point correspondant à la configuration optimale découverte par SEGO en Section 5.2.4.1. Comme dans la Section 5.2.4.1, Evol et Pointer-2 ne nécessitent qu'un seul point pour commencer le processus d'optimisation. C'est pourquoi, on leur fournit le point correspondant à la configuration optimale de la CMDO de la BRAC.

Pour les quatre algorithmes de la toolbox SEGOMOE, le nombre d'évaluations maximum est de max_it_nb = 170, c.-à-d. $10d - n_{start} - 1$ soit au total $10d = 190$ évaluations. On utilise 4 processeurs

en parallèle pour accélérer la construction des GP dans la toolbox SEGOMOE. Toutefois, la toolbox SEGOMOE ne comprend que des algorithmes séquentiels qui n'évaluent qu'un point à chaque itération. Le nombre maximum d'évaluations pour Evol et Pointer-2 est imposé à 510 à l'instar des valeurs historiquement utilisées dans le processus CMDO. De plus, on peut évaluer 4 points à la fois grâce aux options de parallélisation de Evol et Pointer-2. De plus, on choisit un type de problème *smooth* pour Pointer-2.

Courbes de convergence Les courbes de convergence utilisées dans cette section sont légèrement différentes de celles utilisées dans la Section 5.2.4.1. En effet, on ne réalise ici qu'une seule optimisation pour chacune des six méthodes. Ainsi, la Figure 5.13 montre la meilleure valeur faisable de la fonction objectif pour chaque algorithme d'optimisation pour un temps et un nombre d'évaluations croissants.



(a) courbes de convergence en nombre d'évaluations.

(b) Courbes de convergence en temps.

FIGURE 5.13 – Courbes de convergence pour la PMDO de la BRAC. La ligne rouge horizontale indique la valeur de l'optimum de la CMDO de la BRAC dans le processus PMDO.

Premièrement, la Figure 5.13a montre qu'aucune des méthodes d'optimisation ne converge vers le même optimum dont la meilleure valeur est obtenue par SEGO. Des valeurs similaires sont aussi obtenues par SEGOMOE et Evol. On remarque que SEGO et SEGOMOE trouvent une solution similaire à Evol en 30 évaluations de moins. De plus, Pointer-2 donne les moins bonnes valeurs de fonction objectif avec un optimum à 0.1. Pour SEGO-UTB et SEGOMOE-UTB, on constate que la valeur de la fonction objectif optimale n'est pas aussi bonne que celle découverte par SEGO mais est meilleure que celle obtenue par Pointer-2. Pour plus d'information sur les résultats avec d'autres mise à jour du seuil de doute pour SEGO-UTB et SEGOMOE-UTB, le lecteur est invité à se reporter à la Figure A.9.

Deuxièmement, en ce qui concerne les courbes de convergence en temps, Evol et Pointer-2 bénéficient fortement des appels groupés au problème. En effet, les 220 évaluations nécessaires à Evol pour converger sont réalisées en moins de 40 heures, tandis que les algorithmes de la toolbox SEGOMOE ont besoin de 80 heures pour 190 évaluations malgré la parallèlisation de la construc-

tion des GP.

Pour conclure, la Figure 5.13 montre que le critère de faisabilité UTB n'est pas utile pour résoudre la PMDO de la BRAC tandis que SEGO et SEGOMOE montrent de bonnes performances de convergence. Cependant, la capacité de Evol à évaluer le problème en parallèle accélère fortement, en temps de calcul, sa convergence comparé aux autres algorithmes.

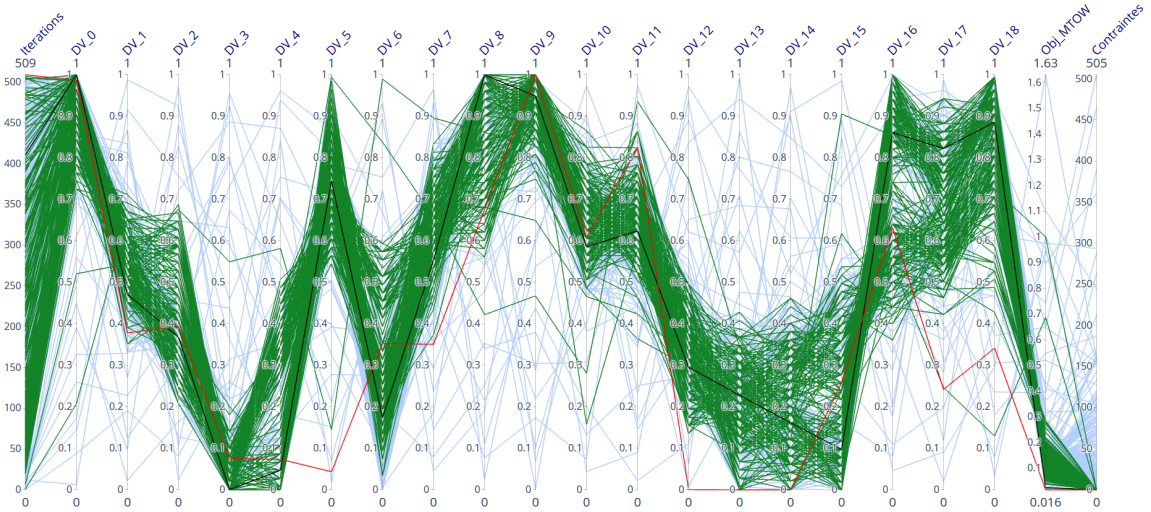
Courbes de coordonnées parallèles : Ici, on compare les six méthodes d'optimisation à l'aide de courbes de coordonnées parallèles. Elles sont précédemment présentées dans la Section 5.2.4.1 bien qu'une légère modification soit à noter. En effet, il n'est pas nécessaire de sélectionner une optimisation spécifique à afficher puisque l'on ne réalise qu'une seule optimisation pour chacun des algorithmes d'optimisation. Les Figures 5.14 et 5.15 représentent les courbes de coordonnées parallèles pour les quatre algorithmes de la toolbox SEGOMOE ainsi que pour Evol et Pointer-2.

Comme déjà mentionné dans le paragraphe précédent, SEGO trouve l'optimum de la CMDO de la BRAC qui offre la MTOW la plus faible. Evol et SEGOMOE trouvent également une configuration donnant une masse du même ordre. Il y a des similitudes dans ces configurations notamment pour l'envergure (DV_0) et l'angle de flèche (DV_1) de l'aile. Toutefois, il y a quelques différences sur la distribution du vrillage de l'aile (DV_4-7) et le rapport entre l'épaisseur maximale de l'aile et la corde (DV_8-13) comme le montrent les Figures 5.14a, 5.14c et 5.15c. Les ailettes ont un impact secondaire sur la MTOW et donc de plus grandes variations sont observées sur les paramètres la représentant (DV_15-18). La meilleure solution des trois autres algorithmes, c.-à-d. Pointer-2, SEGO-UTB et SEGOMOE-UTB, sont complètement différentes de l'optimum obtenu par SEGO. En effet, le nombre d'évaluations restreint imposé à ces algorithmes ne permet pas leur convergence au vu de leur caractère plus exploratoire que SEGO et SEGOMOE.

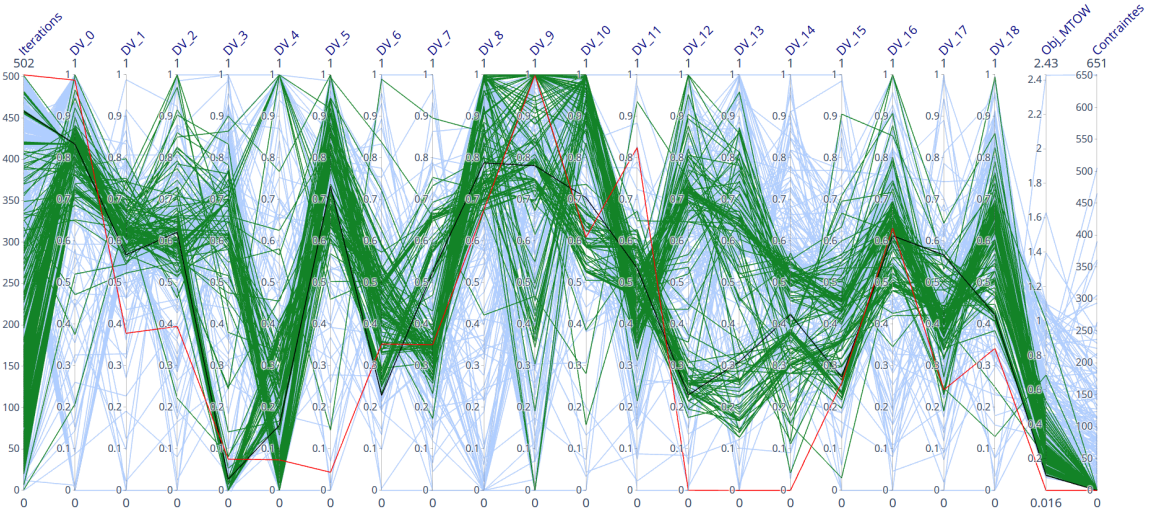
Forme en plan de la configuration optimale : La Figure 5.16 montre les formes en plan de la référence de la BRAC déjà représentée par la Figure 5.7 (en gris), de l'optimum de la CMDO de la BRAC (en vert) et de l'optimum de la PMDO de la BRAC (en bleu). De la même manière que dans la Section 5.2.4.1, on ne montre que la forme en plan des ailes même si la taille du moteur est optimisée par le processus CMDO de la BRAC.

On constate que l'envergure de l'aile de la BRAC après PMDO est plus grande que celle après CMDO. On a donc une correction de l'aile pour que les performances aérodynamiques soient accrues. De plus, on voit que l'angle de flèche est réduit par rapport à la configuration CMDO ; cette correction de l'angle de flèche s'est avérée nécessaire en prenant en compte des modèles aéro-structure de plus haute fidélité.

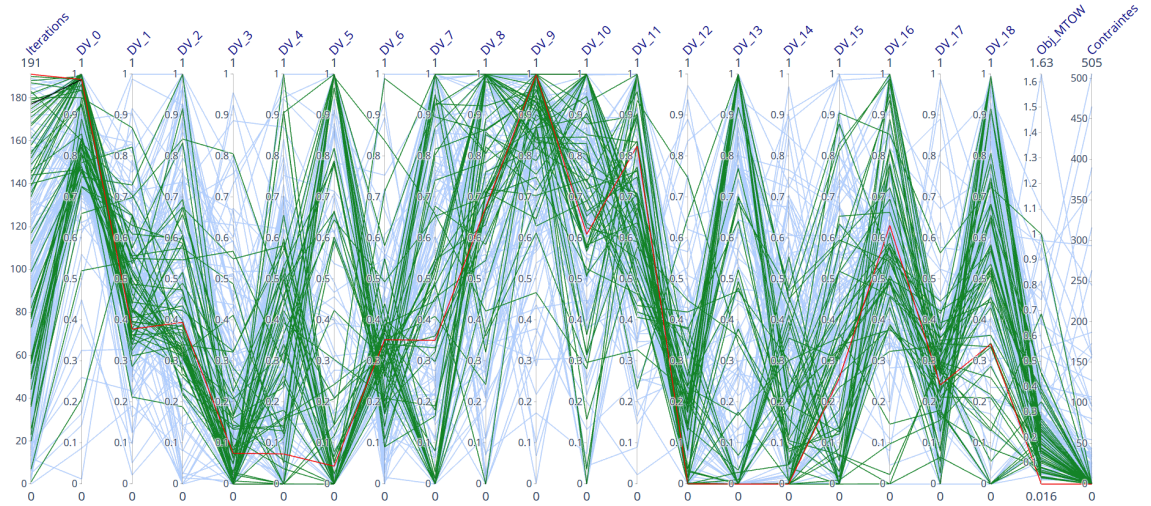
Pour conclure cette section sur la PMDO de la BRAC, on a d'abord introduit la méthode de comparaison entre la toolbox SEGOMOE et les algorithmes natifs de Isight. Cette comparaison est réalisée grâce à des courbes de convergence en nombre d'évaluations et en temps de calcul qui révèlent que la toolbox SEGOMOE est efficace en nombre d'évaluations et trouve une valeur plus faible que les algorithmes de Isigth. Cependant, la capacité de Evol à travailler en parallèle en utilisant plusieurs processeurs lui permet d'être plus attractif en temps de calcul. On a également comparé les optima grâce à des courbes de coordonnées parallèles qui montrent que certaines variables ne sont pas actives notamment les variables relatives aux ailettes, à la distribution du vrillage ou encore aux rapports entre l'épaisseur et la corde de l'aile. Pour finir, on a remarqué que



(a) Evol.

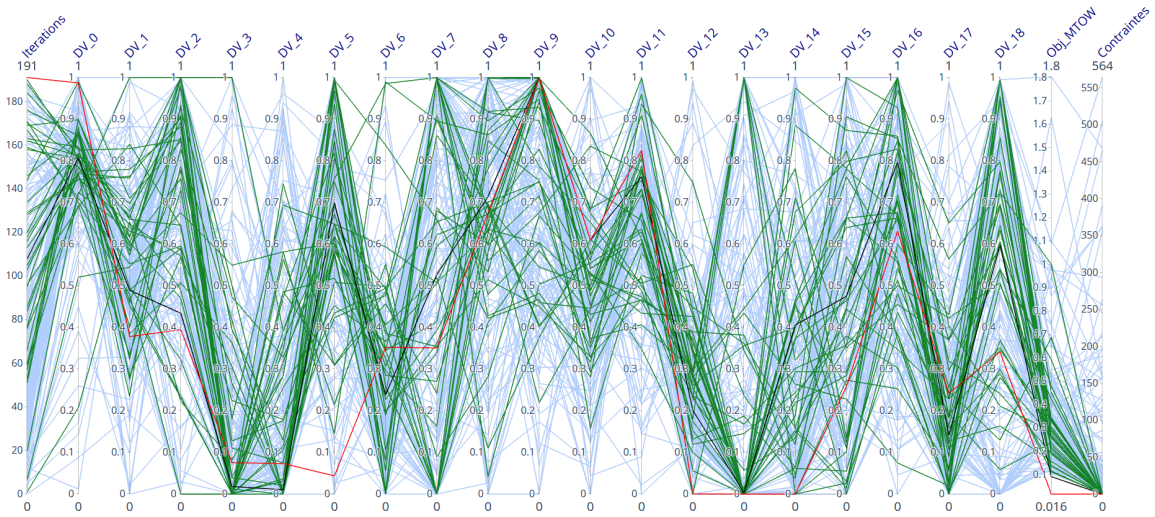


(b) Pointer-2.

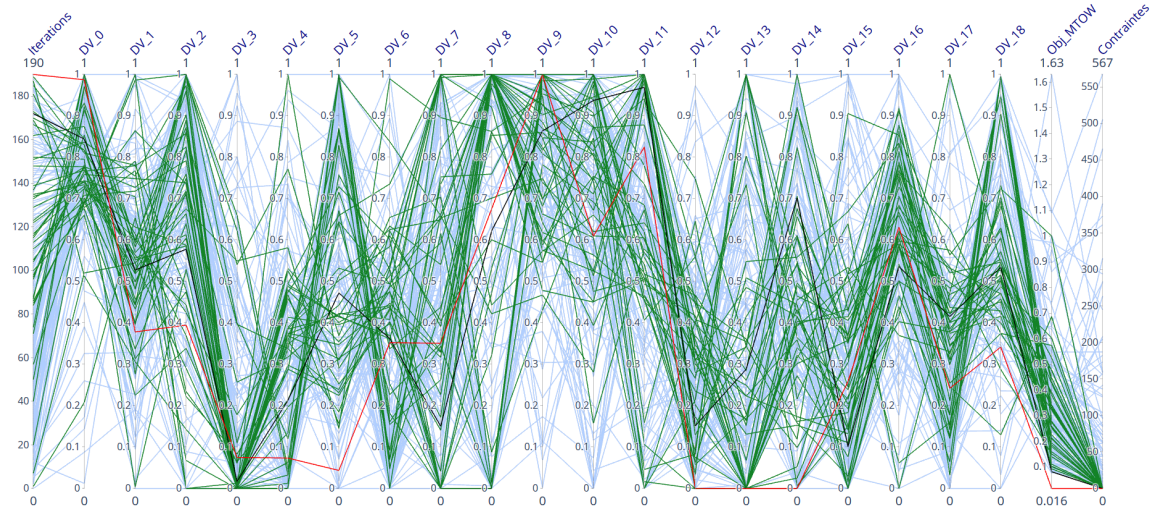


(c) SEGO.

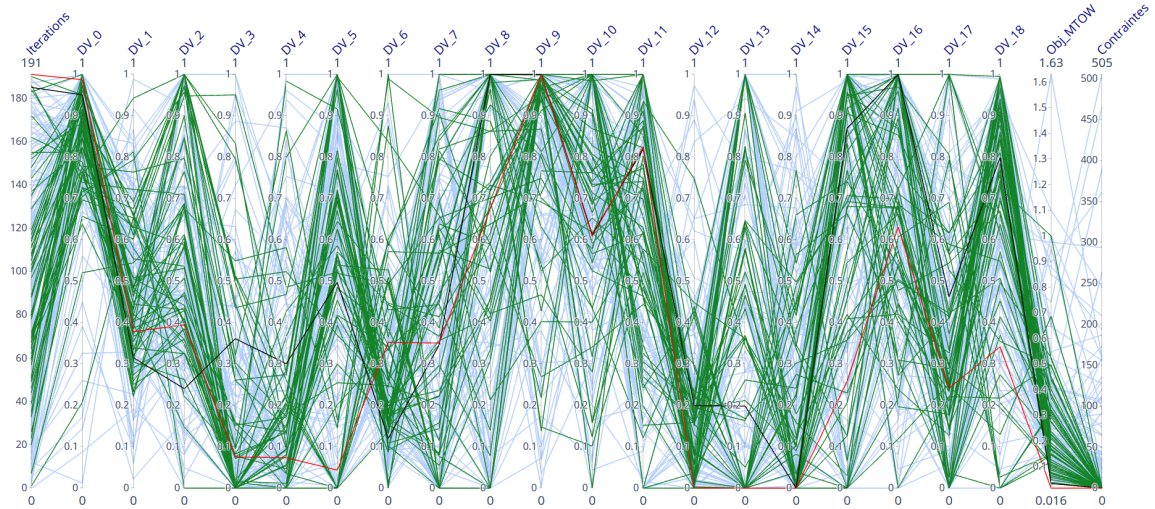
FIGURE 5.14 – Courbes de coordonnées parallèles de l'optimisation médiane de la PMDO de la BRAC pour Evol, Pointer-2 et SEGO. En bleu, les configurations non faisables ; en vert, les configurations faisables ; en noir, l'optimum de la méthode considérée ; en rouge, la solution optimale du problème.



(a) SEGO-UTB.



(b) SEGOMOE-UTB.



(c) SEGOMOE.

FIGURE 5.15 – Courbes de coordonnées parallèles de l'optimisation médiane de la PMDO de la BRAC pour SEGOMOE, SEGO-UTB et SEGOMOE-UTB. En bleu, les configurations non faisables ; en vert, les configurations faisables ; en noir, l'optimum de la méthode considérée ; en rouge, la solution optimale du problème.

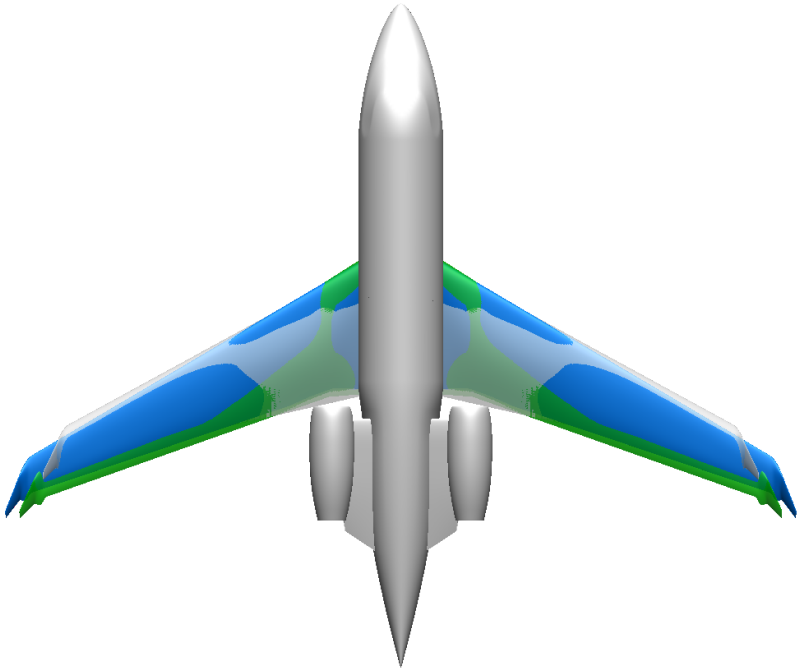


FIGURE 5.16 – Une des formes en plan obtenue par SEGO-UTB (en bleu), la configuration optimale de CMDO (en vert) et la configuration de référence (en gris) de BRAC.

la configuration obtenue par PMDO corrige l'envergure et l'angle de flèche de l'aile de la configuration obtenue par PMDO pour améliorer son aérodynamique.

5.2.5 Conclusion

Dans cette section, on a cherché à évaluer la toolbox SEGOMOE développée à l'ONERA et à l'ISAE-SUPAERO sur des problèmes MDO industriels.

Pour cela, on a d'abord introduit le processus MDO multi-niveaux, multi-fidélité de Bombardier Aviation. On s'est concentré plus spécifiquement sur les deux premiers niveaux : la CMDO et la PMDO. On a ensuite détaillé ces deux processus MDO qui sont notamment implémentés dans le logiciel commercial Isight.

Pour évaluer les algorithmes de la toolbox SEGOMOE, on a introduit la BRAC développée à Bombardier Aviation pour faciliter les collaborations avec le monde académique et qui s'inspire du Challenger 300. Enfin, on a présenté les problèmes CMDO et PMDO de la BRAC.

Par la suite, on a comparé quatre algorithmes de la toolbox SEGOMOE (SEGO, SEGOMOE, SEGO-UTB et SEGOMOE-UTB) avec deux algorithmes d'Isigth (Evol et Pointer-2) classiquement utilisés à Bombardier Aviation. Pour la CMDO de la BRAC, on a constaté que les méthodes de la toolbox SEGOMOE convergent plus vite et vers une MTOW plus faible que les méthodes d'Isight. En effet, Evol et Pointer-2 ne sont pas capables de trouver une valeur de MTOW comparable à SEGO en quatre fois plus de temps. Pour la PMDO de la BRAC, on a remarqué que SEGO converge vers la MTOW la plus faible bien que Evol et SEGOMOE trouvent des valeurs comparables. De plus, SEGO converge le plus rapidement en nombre d'évaluations mais il est deux fois plus lent que Evol en temps de calcul. Ceci est dû au potentiel d'Evol à exploiter l'appel à plusieurs points du domaine de conception en effectuant des calculs en parallèle.

Pour finir, on a noté que certaines variables de conception sont inactives dans la CMDO comme

dans la PMDO de la BRAC.

5.3 Synthèse du chapitre

Dans ce chapitre, on a évalué les performances de SEGO-UTB et de la toolbox SEGOMOE développée à l'ONERA et à l'ISAE-SUPAERO sur la MDO de configurations avion dans les milieux académique et industriel.

Premièrement, on a comparé SEGO-UTB à SEGO, NOMAD et COBYLA sur la MDO d'une configuration avion hybride à propulsion électrique distribuée. Les méthodes ALBO, EFI, SUR et PESC ne sont pas considérées dans cette étude puisqu'elles résolvent difficilement ce type de problème en un temps raisonnable (voir Tableau 4.4). L'analyse des résultats des tests a montré que SEGO et SEGO-UTB ont des performances équivalentes. Ils convergent systématiquement vers la même TEC bien que certaines variables de conception obtenues ne soient pas toujours les mêmes. D'après les experts, ceci correspond à des variables de conception inactives. Par contre, NOMAD ne parvient pas à trouver de configuration faisable à cause de la présence de contraintes d'égalité et COBYLA ne converge pas systématiquement vers la solution.

Deuxièmement, on a évalué quatre algorithmes de la toolbox SEGOMOE (SEGO, SEGOMOE, SEGO-UTB et SEGOMOE-UTB) sur le processus MDO muti-niveaux multi-fidélité développé par Bombardier Aviation. Cette évaluation s'est concentrée sur deux niveaux de fidélité : la PMDO et la CMDO de la BRAC. Comme ces deux niveaux sont implémentés dans le logiciel Isigth fourni par Dassault Systèmes, on compare les quatre méthodes CBO à deux méthodes d'Isigth classiquement utilisées à Bombardier Aviation (Evol et Pointer-2). De plus, Evol et Pointer-2 peuvent bénéficier des capacités de parallélisation de ces processus MDO. On a remarqué que la toolbox SEGOMOE converge plus rapidement, en temps et en nombre d'évaluations, vers une MTOW plus faible qu'Evol et Pointer-2. Pour la PMDO de la BRAC, les résultats sont plus mitigés. En effet, SEGO trouve une BRAC avec la MTOW la plus faible mais il est deux fois plus lent en temps de calcul qu'Evol. Ce phénomène s'explique par la capacité d'Evol à exploiter les appels groupés ce qui n'est pas le cas des algorithmes de la toolbox SEGOMOE. La toolbox SEGOMOE utilise toutefois les capacités de parallélisation de l'ordinateur pour construire les GP modélisant la fonction objectif et les contraintes. En outre, Evol trouve une MTOW similaire à SEGO. Les bonnes performances d'Evol en temps de calcul sont dues à son utilisation de la parallélisation. Pour finir sur la BRAC optimale, on a constaté que certaines variables de conception sont inactives.

Pour conclure, SEGO-UTB n'offre pas de meilleurs résultats que les algorithmes de l'état de l'art sur les cas tests de conception avion industriel et académique. On suspecte que les domaines de faisabilité de ces trois problèmes sont faiblement multi-modaux et ne nécessitent donc pas une exploration intensive du domaine de conception pour découvrir la zone faisable optimale. De plus, SEGO et SEGOMOE donnent de bonnes performances sur les deux configurations avion considérées mais leur caractère séquentiel peut les rendre moins attractifs qu'un algorithme profitant de la parallélisation comme Evol.

Récapitulatif du chapitre

On a répondu aux objectifs introduits en début de chapitre grâce aux points suivants :

— Présentation d'un cas test de conception avion académique.

— Comparaison de SEGO-UTB avec SEGO, NOMAD et COBYLA grâce à des courbes de convergence et des courbes de coordonnées parallèles.

— Présentation du cas test BRAC développé à Bombardier Aviation comportant deux niveaux de fidélité.

— Développement d'une interface pour la toolbox SEGOMOE dans le logiciel Isigth.

— Étude des performances des algorithmes de la toolbox (SEGOMOE, SEGO, SEGOMOE-UTB et SEGO-UTB) par rapport aux méthodes classiquement utilisées à Bombardier Aviation (Pointer-2 et Evol) sur les deux niveaux de fidélité grâce à des courbes de convergence et des courbes de coordonnées parallèles.

Chapitre 6

Conclusion et perspectives

« *L'homme qui doit vanter ses mérites sait que personne ne le fera à sa place.* »

Robin Hobb

Sommaire

Le contexte de cette thèse est celui de l'optimisation Bayésienne, ou Bayesian optimization (BO) [37, 117] sous contraintes et en grande dimension appliquée à la conception avant projet d'avion [26, 99]. La conception avion aspire à créer une configuration avion ayant les meilleures performances possibles, par exemple une consommation d'énergie minimale, et respectant toutes les exigences déjà mentionnées. Afin de pouvoir évaluer ces contraintes et performances, on a recourt à des modèles physiques qui sont souvent très gourmands en temps de calcul. A cause de la complexité des modèles utilisés, les informations de régularité et des dérivées de ces fonctions ne sont pas disponibles. Finalement, un problème de conception avion se traduit mathématiquement par un problème d'optimisation boîte noire sous-contraintes d'égalité et/ou d'inégalité comportant un grand nombre de variables de conception.

Au vue des caractéristiques spécifiques à ce type de problèmes, c.-à-d. gourmand en temps de calcul et boîte noire, on s'est intéressé aux méthodes BO [37, 71, 117]. En effet, ces méthodes d'optimisation sont spécialement développées pour résoudre ce type de problèmes. Dans ce manuscrit, on a cherché à développer des méthodes BO capables de résoudre efficacement des problèmes d'optimisation coûteux à évaluer, comportant des contraintes d'égalité et/ou d'inégalité et un grand nombre de variables de conception.

6.1 Contributions

Il existe de nombreuses méthodes BO mais aucune, à notre connaissance, n'est capable de prendre en compte un problème d'optimisation comprenant des contraintes d'égalité et/ou d'inégalité et un grand nombre de variables de conception. Dans le Chapitre 2, on a analysé les dif-

férentes méthodes BO de la littérature. Ceci nous a permis de révéler plusieurs verrous liés à ces algorithmes :

1. Le temps de calcul CPU nécessaire à la réalisation de l'optimisation est très important lorsque le nombre de variables de conception augmente. Ce phénomène est provoqué par l'estimation de certaines grandeurs nécessitant des procédures d'intégration et/ou de la construction de GP classiques en grande dimension.
2. Certaines méthodes réalisent l'optimisation dans des sous-espaces linéaires aléatoires. Ceci offre une garantie de convergence asymptotique vers le minimum de la fonction considérée mais elle se révèle lente en pratique.
3. Certains types de contraintes ne sont pas pris en compte. Par exemple, les contraintes d'égalité ne sont pas prises en compte ou sont transformées en deux contraintes d'inégalité ce qui double le nombre de contraintes et crée des contraintes antagonistes.
4. L'exploration du domaine de conception à la recherche de zones de faisabilité non-connues n'est pas permise par certaines méthodes.

Ainsi, on a repéré quatre verrous que l'on a tenté de lever dans les Chapitres 3, 4 et 5.

Dans le Chapitre 3, on a essayé de lever les verrous 1 et 2 des méthodes HDBO. Le verrou 1 est provoqué par l'apprentissage d'un GP classique en grande dimension, pour les méthodes TuRBO [34] et EGO-KPLS [18], ou par la résolution de nombreux problèmes quadratiques, pour la méthode RREMBO [15]. Seules les méthodes HeSBO [74] et REMBO [134] ne sont pas sujettes à ce verrou. Le verrou 2 ne concerne que REMBO, RREMBO et HeSBO qui réalisent l'optimisation dans un sous-espace linéaire aléatoirement généré. Ainsi, les directions de l'espace utilisées ne sont pas forcément celles qui apportent le plus d'information. Il n'y a donc aucune certitude que ce sous-espace linéaire permette de découvrir des valeurs de fonction objectif satisfaisantes. Les méthodes TuRBO et EGO-KPLS ne sont pas concernées par ce verrou.

Pour lever le verrou 1, on a proposé une extension de la méthode RREMBO, nommée EGORSE, qui permet de réduire drastiquement le nombre de résolutions de problèmes quadratiques grâce à une reformulation du problème d'optimisation dans le sous-espace linéaire considéré. Cette modification se traduit par l'ajout d'une contrainte qui intègre la résolution du problème quadratique. Ce problème d'optimisation sous-contrainte, défini dans le sous-espace, est résolu avec une méthode CBO classique. Dans notre cas, on a choisi d'utiliser la méthode SEGO [9, 108]. Le verrou 2 est supprimé grâce à un apprentissage adaptatif des meilleures directions de recherche. Ce processus repose sur l'utilisation de méthodes de réduction de dimension supervisées, comme les PLS [50] ou les MGP [40]. Ainsi, l'optimisation est pratiquée dans un sous-espace linéaire dans lequel la fonction objectif varie fortement.

Les capacités d'EGORSE ont été testées sur cinq problèmes analytiques : quatre sont dérivés de la fonction objectif du problème MB [79] et comportent 10, 20, 100 ou 1000 variables de conception, le dernier est un problème de routage d'un robot dans une forêt [136] et compte 60 variables de conception. On a comparé les résultats obtenus par EGORSE avec ceux obtenus par des méthodes de la littérature (TuRBO, EGO-KPLS, RREMBO et HeSBO). Toutefois, la méthode EGORSE n'a pas pu être comparée avec RREMBO, TuRBO et EGO-KPLS sur le problème MB_1000 car ces trois derniers algorithmes auraient été beaucoup trop coûteux en temps de calcul CPU. Ces tests ont montré qu'EGORSE est plus performant que HeSBO et RREMBO sur l'ensemble des problèmes

bien qu'aucun de ces trois algorithmes ne parvienne pas à trouver l'optimum des problèmes. Ceci est certainement dû au faible nombre de directions de recherche utilisées. On suppose qu'avec un nombre de directions efficaces plus important, on arrive à mieux prendre en compte la variabilité de la fonction et donc à converger vers des valeurs de fonction objectif plus faibles. On doit donc trouver un compromis entre la rapidité de l'algorithme, avec un nombre faible de directions de recherche, et la convergence de l'algorithme, avec un nombre plus élevé de directions de recherche. De plus, EGORSE est plus rapide que RREMBO à réaliser l'optimisation mais il est moins rapide que HeSBO. En effet, HeSBO ne résout aucun problème quadratique. Au contraire, RREMBO résout beaucoup plus de problèmes quadratiques qu'EGORSE. Toutefois EGORSE est moins performant que TuRBO et EGO-KPLS sur les problèmes ayant moins de 100 variables de conception. Effectivement, TuRBO et EGO-KPLS réalisent l'optimisation dans l'espace de grande dimension ce qui ne restreint pas l'exploration du domaine.

On souligne que lors de la rédaction du manuscrit, un article décrivant une méthode du même type qu'EGORSE, nommée S^3-BFO, a été publiée par SHILTON et al. [118]. Elle repose sur la construction d'un GP pour détecter les directions de recherche et centre le sous-espace de recherche sur le meilleur point du DoE. Cette méthode est évaluée sur des problèmes plus concrets qu'EGORSE, notamment sur un problème de précipitation dans un alliage et sur un problème d'optimisation d'hyper-paramètres d'un réseau de neurones. Toutefois, SHILTON et al. [118] ne comparent pas S^3-BFO avec les algorithmes de la littérature utilisés dans cette étude.

On a cherché, dans le Chapitre 4, à mettre en place une méthode CBO levant les verrous 3 et 4. De plus, on cherche à ce que la méthode utilisée ne soit pas limitée par le verrou 1. En effet, certaines méthodes sont limitées par ce verrou dû à l'intégration de grandeurs sur l'ensemble du domaine de conception. Cette intégration s'avère de plus en plus coûteuse lorsque le nombre de variables de conception augmente. Les méthodes ALBO [84], SUR [83] et PESC [51] sont concernées par ce verrou. Le verrou 3 touche les méthodes qui ne prennent pas en compte les contraintes d'égalité et d'inégalité simultanément comme EFI [112], PESC, SUR, SEGO-EV [2], SEGO-UTB [63]. Le verrou 4 touche les méthodes (SEGO et SEGO-EV) limitant l'exploration du domaine de conception lorsqu'une zone faisable est découverte.

Pour lever ces verrous, on a présenté une extension pour les contraintes d'égalité du critère de faisabilité UTB. De plus ce critère de faisabilité a l'avantage d'être analytique ce qui rend son évaluation rapide. Enfin, le critère UTB est capable d'explorer le domaine à la recherche de zones faisables ce qui n'est pas le cas des autres méthodes à critère de faisabilité (SEGO et SEGO-EV).

On a ensuite évalué la rapidité et la robustesse de convergence de SEGO-UTB par rapport aux méthodes de la littérature (SEGO, EFI, SUR, PESC et ALBO). On l'a également comparé avec deux méthodes classiques d'optimisation sans dérivées : NOMAD [3] et COBYLA [90]. On a d'abord analysé les résultats de quatre problèmes de faible dimension comportant des contraintes multimodales d'égalité et/ou d'inégalité. On a constaté que SEGO-UTB converge le plus rapidement vers les valeurs les plus faibles de fonction objectif faisable. Ainsi, on a validé la levée du verrou 3. On a également étudié les performances de ces algorithmes sur un ensemble de 29 problèmes. On a remarqué que SEGO est l'algorithme qui résout le plus de problèmes bien que SEGO-UTB reste fortement compétitif. Cependant, aucun de ces deux algorithmes n'est capable de résoudre un problème comportant plus d'une dizaine de variables de conception en un temps raisonnable.

On a ensuite évalué les capacités de la toolbox SEGOMOE, contenant l'algorithme SEGO-UTB,

sur deux cas tests de conception avion. La toolbox SEGOMOE est développée par l'ONERA et l'ISAE-SUPAERO.

Le premier cas test est un problème de conception d'un avion hybride à propulsion électrique distribuée sur le bord de fuite de l'aile. Le processus d'évaluation des performances et de faisabilité de cette configuration est codé avec FAST [110], un outil de conception avion avant-projet développé par l'ONERA et l'ISAE-SUPAERO. On a comparé SEGO, SEGO-UTB, NOMAD et COBYLA sur ce problème comprenant 12 variables de conception, 8 contraintes d'inégalité et 3 contraintes d'égalité. En effet, les autres méthodes CBO de la littérature ne sont pas capables de résoudre ce problème en un temps raisonnable. Au final, SEGO-UTB converge plus rapidement et vers une valeur de fonction objectif plus faible que les trois autres algorithmes. On a donc montré la capacité de SEGO-UTB à résoudre des problèmes de conception avion.

Afin de valider ces résultats sur une problématique industrielle, on s'est intéressé à un deuxième cas test de conception avion. Ce cas test, développé par Bombardier Aviation pour faciliter les interactions avec la recherche, se compose de deux niveaux de fidélité. Les tests relatifs à ce problème ont été exécutés lors d'une mobilité internationale à Bombardier Aviation à Montréal (CA). On a comparé, sur les deux niveaux de fidélité, les méthodes SEGO, SEGOMOE, SEGO-UTB et SEGOMOE-UTB codées dans la toolbox SEGOMOE avec les méthodes Pointer-2 et Evol [132] classiquement employées à Bombardier. Sur le niveau de plus basse fidélité, avec 12 variables de conception et 8 contraintes d'inégalité, les méthodes de la toolbox SEGOMOE ont montré les meilleures performances en terme de rapidité en temps et en nombre d'évaluations. De plus, les valeurs de fonction objectif découvertes sont plus faibles que celles obtenues par Pointer-2 et Evol. Les tests réalisés sur le niveau de plus haute fidélité, comportant 19 variables de conception et 5 contraintes, permettent de comparer les algorithmes de la toolbox SEGOMOE avec Evol et Pointer-2. En terme de nombre d'évaluations, les méthodes SEGO et SEGOMOE montrent des performances équivalentes à Evol. Au contraire, les méthodes SEGO-UTB et SEGOMOE-UTB ne parviennent pas à converger. Ce phénomène est dû au faible budget d'évaluations accordé pour la résolution. En s'intéressant à la convergence en temps, on constate qu'Evol est bien plus rapide que les algorithmes de la toolbox SEGOMOE. En effet, Evol utilise massivement les capacités de parallélisation ce qui n'est pas le cas de SEGO et SEGOMOE. Pour conclure, le critère de faisabilité UTB n'apporte pas d'amélioration par rapport aux méthodes de la littérature sur les cas tests réalisés à Bombardier Aviation. On suspecte que les contraintes ne sont pas "suffisamment" multi-modales ce qui rend SEGO plus performant par rapport à SEGO-UTB.

6.2 Perspectives

On termine avec les perspectives amenées par ces travaux de recherche. On rappelle que l'on a fixé le nombre de directions efficaces d_e dans la méthode EGORSE. Ce choix arbitraire ne permet pas de garantir que le sous-espace considéré contienne le minimum global de la fonction objectif. C'est pourquoi une évaluation de l'impact du nombre de directions efficaces d_e sur la rapidité et la robustesse de convergence d'EGORSE semble pertinente. De plus, ce nombre de directions efficaces dépend fortement du problème considéré, on pourrait imaginer une méthode capable de détecter, à partir des données du DoE, le nombre de directions efficaces le plus approprié. Par exemple, on pourrait réaliser une sélection du meilleur nombre de directions efficaces grâce à une

validation croisée sur la qualité des PLS ou encore étudié la sensibilité de la fonction objectif aux variables de conceptions à l'instar de BEN SALEM et al. [11], SPAGNOL et al. [123]. Dans le cas où le nombre de directions efficaces est important ($d_e \geq 5$), on pourrait utiliser la méthode EGO-KPLS qui a montré de bons résultats sur des problèmes comportant jusqu'à 100 variables de conception.

Les contraintes ne sont pas prises en compte par les méthodes EGORSE, RREMBO, REMBO et HeSBO. Un axe de recherche pourrait s'intéresser à la prise en compte des contraintes lorsque l'on utilise un algorithme réalisant l'optimisation dans des sous-espaces. Par exemple, on pourrait pénaliser les points non faisables lorsque l'on cherche les directions efficaces avec la méthode des PLS. On pourrait aussi, à l'instar de S^3-BFO, centrer le sous-espace linéaire sur le meilleur point du DoE. S'il est faisable, alors on aurait des points faisables dans le sous-espace considéré, sinon on rechercherait des points faisables autour du point ayant la plus faible violation des contraintes.

A ce jour, EGORSE n'a été testé que sur des problèmes académiques. Comme pour SEGO-UTB, il faut évaluer ses performances sur des problèmes plus concrets comme cela est fait dans les travaux de SHILTON et al. [118]. De plus, une comparaison avec une plus grande variété de méthodes HDBO pourrait être intéressante. Plus particulièrement, on peut penser aux méthodes de KANDASAMY et al. [59], LUKACZYK et al. [66], OH et al. [77], WANG et al. [136] et de ZHANG et al. [140]. La plupart de ces codes sont disponibles soit en Python 2.7, qui n'est plus supporté, soit en Matlab, qui est un logiciel propriétaire. Ainsi, le test de ces codes nécessite une implémentation en Python 3 pour être cohérent avec nos choix d'implémentation et qui pourrait demander un travail conséquent.

Pour finir avec la méthode EGORSE, on ne considère qu'une réduction de dimension linéaire. Cette hypothèse est très forte et peut fortement limiter l'algorithme si la fonction objectif est définie dans un sous-espace dont l'application de passage est non linéaire. C'est pourquoi on pourrait s'intéresser à des méthodes de réduction de dimension non-linéaires comme les Isomap [125] ou l'apprentissage de variétés [47]. Cependant, l'application de retour du sous-espace vers l'espace initial est beaucoup moins triviale que dans le cas linéaire. Ce dernier point est crucial pour une utilisation efficace de ces méthodes non linéaires.

Le Chapitre 5 a montré les lacunes de la toolbox SEGOMOE en terme de prise en compte d'évaluations groupées du problème d'optimisation. Cette fonctionnalité pourrait rendre la toolbox SEGOMOE plus compétitive par rapport à des algorithmes d'optimisation classiques. On peut notamment penser à la méthode qEI [46] et à l'utilisation de plusieurs fonctions d'acquisition ou critères de faisabilité en parallèle [129].

De plus, SEGO-UTB a montré de bonnes performances sur un ensemble conséquent de problèmes mais son caractère fortement exploratoire ne le rend pas compétitif sur des problèmes industriels. Une recherche pour une meilleure prise en compte des contraintes pourrait s'avérer nécessaire notamment pour autoriser une exploration du domaine tout en s'assurant qu'elle ne soit pas prépondérante. Le critère de BICHON et al. [13], reposant sur l'espérance de faisabilité d'un point, pourrait être une piste à approfondir. En effet, cette espérance pourrait, à l'instar de l'EI, contrôler automatiquement le compromis entre l'exploration du domaine et l'exploitation des données. On pourrait également penser à un seuil de doute adaptatif qui permettrait de mieux contrôler le compromis exploration/exploitation.

Dans un cadre plus général, de nombreuses variables en conception avion peuvent prendre des valeurs entières (nombre de moteurs, nombre d'ailes, nombre de panneaux solaires, etc.) ou

peuvent être catégorielles (type de moteur, type d'alliage ou de composite, forme globale des ailes, etc.). Cependant, ces deux types de variables sont difficilement pris en compte par les méthodes BO classiques qui ne considèrent que les problèmes à variables continues. Dans le cadre de la conception avion avant-projet, on pourrait s'intéresser à la prise en compte de ces types de variables. Par exemple, on pourrait s'inspirer des méthodes de PELAMATTI et al. [81], ROUSTANT et al. [106] et de GARRIDO-MERCHÁN et HERNÁNDEZ-LOBATO [41]. Ces méthodes reposent majoritairement sur une reformulation du noyau de covariance des GP modélisant la fonction objectif et les contraintes.

Les variables considérées dans le processus BO peuvent aussi être soumise à de l'incertitude, c.-a-d. qu'il est impossible de donner une valeur fixe pour une variable donnée. En réalité, on ne peut définir que des distributions représentant chaque variable. Cette variabilité sur les entrées doit être prise en compte afin de trouver des minima qui soient robustes à ces incertitudes. Pour cela, on pourrait s'inspirer de la méthode de BELAND et NAIR [10], qui utilisent une intégration de la fonction objectif conditionnée par les distributions des entrées, ou la méthode de OLIVEIRA et al. [78].

Dans le même esprit, les fonctions considérées peuvent donner des résultats différents pour un même ensemble de variables. On est donc dans le cas d'une incertitude sur les sorties du problème d'optimisation. La plupart des méthodes BO prennent en compte cette incertitude à travers l'ajout d'un terme constant sur la diagonale de la matrice de covariance des GP modélisant ces fonctions. Cependant, cette possibilité n'est pas prise en compte dans la toolbox SEGOMOE développée par l'ONERA et l'ISAE-SUPAERO. Dans l'état, si on ajoute ce terme fixe dans la diagonale, le sous-problème d'optimisation peut proposer plusieurs fois le même point à évaluer ce qui rendrait la matrice de covariance non inversible. Pour éviter cela, on pourrait utiliser la méthode de ré-interpolation de FORRESTER et al. [36].

Pour finir avec les perspectives de ce travail, on souligne que les processus MDO avion avant-projet couplent et rendent inter-dépendant de nombreuses disciplines comme la propulsion, l'aérodynamique, la structure, etc. Lors du processus d'optimisation, ces inter-dépendances peuvent causer des incohérences dans les variables échangées entre discipline. Ces incohérences engendrent souvent des défaillances dans l'évaluation de la fonction objectif ou des contraintes du problèmes. Le problème ne renvoie donc qu'un message d'erreur. Cependant, les méthodes BO classiques ne considèrent pas ce type de réponses. On pourrait donc s'intéresser à la gestion des défaillances dans les processus BO. Par exemple, on peut donner une valeur empêchant le processus BO d'évaluer des points dans ces zones de l'espace [36], ou utiliser des méthodes de classification [6].

Pour conclure, ces travaux de thèse ont été l'occasion de renforcer de nombreuses collaborations entre l'ONERA, l'ISAE-SUPAERO, Bombardier Aviation et le TsAGI. Ils ont aussi permis de réaffirmer la nécessité de faire le lien entre la rechercher académique et l'application industrielle bien que leur indépendance reste indispensable. Ils ont aussi été l'occasion de vérifier l'intérêt grandissant pour les méthodes de ML en ingénierie, notamment via les applications de l'AI.

Références

[1] ANDERSEN, M. S., J. DAHL et L. VANDENBERGHE. 2013, «Cvxopt : Python software for convex optimization», URL https://cvxopt.org/. 64

[2] AUDET, C., J. DENNI, D. MOORE, A. BOOKER et P. FRANK. 2000, «A surrogate-model-based method for constrained optimization», dans Proceedings of the 8th Symposium on Multidisciplinary Analysis and Optimization, p. 4891. 36, 42, 43, 87, 157

[3] AUDET, C. et J. E. DENNIS JR. 2006, «Mesh adaptive direct search algorithms for constrained optimization», SIAM Journal on optimization, vol. 17, p. 188–217. 111, 126, 157

[4] AUDET, C. et W. HARE. 2017, Derivative-Free and Blackbox Optimization, Springer International Publishing. 2

[5] BACHOC, F. 2013, «Cross validation and maximum likelihood estimations of hyper-parameters of Gaussian processes with model misspecification», Computational Statistics & Data Analysis, vol. 66, p. 55–69. 11

[6] BACHOC, F., C. HELBERT et V. PICHENY. 2020, «Gaussian process optimization with failures : Classification and convergence proof», Journal of Global Optimization, p. 1–24. 160

[7] BARTOLI, N., M. A. BOUHLEL, I. KUREK, R. LAFAGE, T. LEFEBVRE, J. MORLIER, R. PRIEM, V. STILZ et R. REGIS. 2016, «Improvement of efficient global optimization with application to aircraft wing design», dans Proceedings of the 17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, p. 4001. 42

[8] BARTOLI, N., T. LEFEBVRE, S. DUBREUIL, R. OLIVANTI, N. BONS, J. R. R. A. MARTINS, M. A. BOUHLEL et J. MORLIER. 2017, «An adaptive optimization strategy based on mixture of experts for wing aerodynamic design optimization», dans Proceedings of the 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, p. 4433. 42

[9] BARTOLI, N., T. LEFEBVRE, S. DUBREUIL, R. OLIVANTI, R. PRIEM, N. BONS, J. R. R. A. MARTINS et J. MORLIER. 2019, «Adaptive modeling strategy for constrained global optimization with application to aerodynamic wing design», Aerospace Science and Technology, vol. 90, p. 85–102. 2, 5, 20, 21, 23, 36, 42, 45, 64, 76, 87, 93, 111, 134, 139, 156

[10] BELAND, J. J. et P. B. NAIR. 2017, «Bayesian optimization under uncertainty», dans NIPS BayesOpt 2017 Workshop. 160

[11] BEN SALEM, M., F. BACHOC, O. ROUSTANT, F. GAMBOA et L. TOMASO. 2019, «Gaussian Process-Based Dimension Reduction for Goal-Oriented Sequential Design», SIAM/ASA Journal on Uncertainty Quantification, vol. 7, p. 1369–1397. 159

[12] BETTEBGHOR, D., N. BARTOLI, S. GRIHON, J. MORLIER et M. SAMUELIDES. 2011, «Surrogate modeling approximation using a mixture of experts based on EM joint estimation», Structural and multidisciplinary optimization, vol. 43, p. 243–259. 17

[13] BICHON, B. J., M. S. ELDRED, L. P. SWILER, S. MAHADEVAN et J. M. MCFARLAND. 2008, «Efficient global reliability analysis for nonlinear implicit performance functions», AIAA journal, vol. 46, p. 2459–2468. 159

[14] BINOIS, M., D. GINSBOURGER et O. ROUSTANT. 2015, «A warped kernel improving robustness in Bayesian optimization via random embeddings», dans Proceedings of the 9th International Conference on Learning and Intelligent Optimization, p. 281–286. 26

[15] BINOIS, M., D. GINSBOURGER et O. ROUSTANT. 2020, «On the choice of the low-dimensional domain for global optimization via random embeddings», Journal of global optimization, vol. 76, p. 69–90. xi, 26, 27, 28, 29, 30, 31, 32, 45, 53, 54, 55, 56, 57, 61, 65, 76, 156

[16] BONS, N. P., X. HE, C. A. MADER et J. R. R. A. MARTINS. 2019, «Multimodality in aerodynamic wing design optimization», AIAA Journal, vol. 57, p. 1004–1018. 45

[17] BOUHLEL, M. A., N. BARTOLI, A. OTSMANE et J. MORLIER. 2016, «Improving kriging surrogates of high-dimensional design models by Partial Least Squares dimension reduction», Structural and Multidisciplinary Optimization, vol. 53, p. 935–952. 12, 13, 16, 19, 139

[18] BOUHLEL, M. A., N. BARTOLI, R. G. REGIS, A. OTSMANE et J. MORLIER. 2018, «Efficient global optimization for high-dimensional constrained problems by using the Kriging models combined with the partial least squares method», Engineering Optimization, p. 1–16. 26, 27, 45, 54, 76, 156

[19] BOUHLEL, M. A., J. T. HWANG, N. BARTOLI, R. LAFAGE, J. MORLIER et J. R. R. A. MARTINS. 2019, «A Python surrogate modeling framework with derivatives», Advances in Engineering Software, vol. 135, p. 102 662. 12, 64, 65, 91

[20] CALANDRA, R., J. PETERS, C. E. RASMUSSEN et M. P. DEISENROTH. 2016, «Manifold Gaussian Processes for regression», dans Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), p. 3338–3345. 12, 26

[21] CHEN, J., G. ZHU, R. GU, C. YUAN et Y. HUANG. 2020, «Semi-supervised Embedding Learning for High-dimensional Bayesian Optimization», arXiv :2005.14601. 12

[22] CHEN, Y., A. HUANG, Z. WANG, I. ANTONOGLOU, J. SCHRITTWIESER, D. SILVER et N. DE FREITAS. 2018, «Bayesian Optimization in AlphaGo», arXiv :1812.06855. 25

[23] COELLO, C. A., G. T. PULIDO et M. S. LECHUGA. 2004, «Handling multiple objectives with particle swarm optimization», IEEE Transactions on evolutionary computation, vol. 8, p. 256–279. 140

[24] CONN, A. R., K. SCHEINBERG et L. N. VICENTE. 2009, Introduction to Derivative-Free Optimization, SIAM. 2

[25] CONSTANTINE, P. G. 2015, Active Subspaces : Emerging Ideas for Dimension Reduction in Parameter Studies, SIAM. 26

[26] CRAMER, E., J. J. DENNIS, P. FRANK, R. LEWIS et G. SHUBIN. 1994, «Problem Formulation for Multidisciplinary Optimization», SIAM Journal on Optimization, vol. 4, p. 754–776. 2, 155

[27] DAMIANOU, A. et N. LAWRENCE. 2013, «Deep gaussian processes», dans Proceedings of Artificial Intelligence and Statistics, p. 207–215. 12

[28] DE BARROS, A. G. et S. C. WIRASINGHE. 1997, «New aircraft characteristics related to airport planning», dans Proceedings of the 1st ATRG Conference, p. 25–27. 126

[29] DEBLOIS, A. 2013, «A new hybrid-adjoint mdo framework for high-fidelity aerostructural optimization», dans Proceedings of the SAE 2013 Aerotech Congress and Exhibition. 136

[30] DIGABEL, S. et S. M. WILD. 2015, «A taxonomy of constraints in simulation-based optimization», arXiv :1505.07881. 86

[31] DIOUANE, Y., S. GRATTON et L. N. VICENTE. 2015, «Globally convergent evolution strategies for constrained optimization», Computational Optimization and Applications, vol. 62, p. 323–346. 107

[32] DUVENAUD, D., J. LLOYD, R. GROSSE, J. TENENBAUM et G. ZOUBIN. 2013, «Structure Discovery in Nonparametric Regression through Compositional Kernel Search», dans Proceedings of the 28th International Conference on Machine Learning, p. 1166–1174. 10, 11, 12

[33] ERIKSSON, D., M. PEARCE, J. GARDNER, R. D. TURNER et M. POLOCZEK. 2019, «Scalable Global Optimization via Local Bayesian Optimization», dans Advances in Neural Information Processing Systems, p. 5497–5508. 27, 33, 45, 54, 76

[34] ERIKSSON, D. et M. POLOCZEK. 2020, «Scalable Constrained Bayesian Optimization», arXiv :2002.08526. 45, 46, 156

[35] FALKNER, V. M. 1946, «The Accuracy of Calculations Based on Vortex Lattice Theory, Rep. No. 9621», British ARC. 134

[36] FORRESTER, A., A. SOBESTER et A. KEANE. 2008, Engineering Design via Surrogate Modelling : A Practical Guide, John Wiley & Sons. 8, 20, 160

[37] FRAZIER, P. I. 2018, «A Tutorial on Bayesian Optimization», arXiv :1807.02811. 3, 20, 21, 23, 36, 155

[38] FREEMAN, J., P. OSTERKAMP, M. W. GREEN, A. R. GIBSON et B. T. SCHILTGEN. 2014, «Challenges and opportunities for electric aircraft thermal management», Aircraft Engineering and Aerospace Technology : An International Journal, vol. 86, p. 519–524. 125

[39] GARDNER, J., C. GUO, K. WEINBERGER, R. GARNETT et R. GROSSE. 2017, «Discovering and exploiting additive structure for Bayesian optimization», dans Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, p. 1311–1319. 26

[40] GARNETT, R., M. OSBORNE et P. HENNIG. 2014, «Active Learning of Linear Embeddings for Gaussian Processes», dans Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI 2014), p. 10. 12, 13, 14, 15, 17, 19, 57, 65, 156, V

[41] GARRIDO-MERCHÁN, E. C. et D. HERNÁNDEZ-LOBATO. 2020, «Dealing with categorical and integer-valued variables in Bayesian Optimization with Gaussian processes», Neurocomputing, vol. 380, p. 20–35. 160

[42] GAUDRIE, D., R. LE RICHE, V. PICHENY, B. ENAUX et V. HERBERT. 2020, «Modeling and optimization with Gaussian processes in reduced eigenbases», Structural and Multidisciplinary Optimization, vol. 61, p. 2343–2361. 61

[43] GELBART, M. A., J. SNOEK et R. P. ADAMS. 2014, «Bayesian optimization with unknown constraints», dans Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence, UAI 2014, p. 250–259. 36

[44] GELMAN, A., J. B. CARLIN, H. S. STERN et D. B. RUBIN. 2014, «Bayesian data analysis (Vol. 2)», Boca Raton, FL : Chapman. 11

[45] GILL, P. E., W. MURRAY et M. A. SAUNDERS. 2005, «Snopt : An sqp algorithm for large-scale constrained optimization», SIAM review, vol. 47, p. 99–131. 64, 92

[46] GINSBOURGER, D., R. LE RICHE et L. CARRARO. 2007, «A multi-points criterion for deterministic parallel global optimization based on kriging», dans Proceedings of the International Conference on Nonconvex Programming, Local and Global Approaches. Theory, Algorithms and 159

[47] GORBAN, A. N., B. KÉGL, D. C. WUNSCH et A. Y. ZINOVYEV. 2008, Principal Manifolds for Data Visualization and Dimension Reduction, vol. 58, Springer. 159

[48] GRAY, J. S., G. K. KENWAY, C. A. MADER et J. R. R. A. MARTINS. 2018, «Aero-propulsive design optimization of a turboelectric boundary layer ingestion propulsion system», dans Proceedings of the 2018 Aviation Technology, Integration, and Operations Conference, p. 3976. 125

[49] GRIFFITHS, R. R. et J. M. HERNÁNDEZ-LOBATO. 2020, «Constrained Bayesian optimization for automatic chemical design using variational autoencoders», Chemical Science. 37

[50] HELLAND, I. S. 1988, «On the structure of partial least squares regression», Communications in statistics-Simulation and Computation, vol. 17, p. 581–607. 16, 57, 65, 156

[51] HERNÁNDEZ-LOBATO, J. M., M. A. GELBART, R. P. ADAMS, M. W. HOFFMAN et Z. GHAHRAMANI. 2016, «A general framework for constrained Bayesian optimization using information-based search», Journal of Machine Learning Research. 21, 25, 36, 38, 39, 87, 111, 132, 157

[52] HOOKE, R. et T. A. JEEVES. 1961, «" Direct Search" Solution of Numerical and Statistical Problems», J. ACM, vol. 8, p. 212–229. 140

[53] HOULSBY, N., F. HUSZÁR, Z. GHAHRAMANI et M. LENGYEL. 2011, «Bayesian active learning for classification and preference learning», arXiv preprint arXiv :1112.5745. 12, 15

[54] JEYARAJ, A., F. SANCHEZ, P. EARNEST, E. MURUGESAN, R. PRIEM et S. LISCOUET-HANKE. 2019, «Exploring Collaborative and Multidisciplinary Aircraft Optimization through the AGILE Academy Challenge – A case study for an aircraft auxiliary solar power system», dans Proceedings of the 19th Astronautics Conference of the Canadian Aeronautics and Space Institute. 4, 5, 6

[55] JOHNSON, S. G. 2014, «The NLopt nonlinear-optimization package», Software available at https://nlopt.readthedocs.io. 64, 92

[56] JONES, D. R. 2008, «Large-scale multi-disciplinary mass optimization in the auto industry», dans Proceedings of the MOPTA 2008 Conference (20 August 2008). 45, 46

[57] JONES, D. R., M. SCHONLAU et W. J. WELCH. 1998, «Efficient global optimization of expensive black-box functions», Journal of Global optimization, vol. 13, p. 455–492. 19, 20, 21, 22

[58] JONES, E., T. OLIPHANT, P. PETERSON et al.. 2001–, «SciPy : Open source scientific tools for Python», Software available at http://www.scipy.org/. 111

[59] KANDASAMY, K., J. SCHNEIDER et B. PÓCZOS. 2015, «High dimensional Bayesian optimisation and bandits via additive models», dans Proceedings of the 32nd International Conference on Machine Learning, vol. 37, p. 295–304. 26, 159

[60] KIRSCHNER, J., M. MUTNY, N. HILLER, R. ISCHEBECK et A. KRAUSE. 2019, «Adaptive and Safe Bayesian Optimization in High Dimensions via One-Dimensional Subspaces», dans Proceedings of the 36th International Conference on Machine Learning, p. 3429–3438. 26

[61] KRIGE, D. G. 1951, «A statistical approach to some basic mine valuation problems on the Witwatersrand», Journal of the Southern African Institute of Mining and Metallurgy, vol. 52, p. 119–139. 8, 20, 92

[62] KUSHNER, H. J. 1964, «A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise», . 21

[63] LAM, R., D. L. ALLAIRE et K. E. WILLCOX. 2015, «Multifidelity optimization using statistical surrogate modeling for non-hierarchical information sources», dans Proceedings of the 56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, p. 0143. 36, 42, 43, 87, 92, 157

[64] LAMBE, A. B. et J. R. R. A. MARTINS. 2012, «Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes», Structural and Multidisciplinary Optimization, vol. 46, p. 273–284. xii, 63

[65] LE DIGABEL, S. 2011, «Algorithm 909 : NOMAD : Nonlinear Optimization with the MADS Algorithm», ACM Transactions on Mathematical Software (TOMS), vol. 37, p. 1–15. 111, 126

[66] LUKACZYK, T. W., P. CONSTANTINE, F. PALACIOS et J. J. ALONSO. 2014, «Active Subspaces for Shape Optimization», dans Proceedings of the 10th AIAA Multidisciplinary Design Optimization Conference, p. 1171. 26, 159

[67] MACKAY, D. J. C. 1998, Neural Networks and Machine Learning, Springer-Verlag. 12

[68] MASKEW, B. 1982, «PROGRAM VSAERO : A computer program for calculating the non-linear aerodynamic characteristics of arbitrary configurations : User's manual», . 135

[69] MEZURA-MONTES, E. et O. CETINA-DOMÍNGUEZ. 2012, «Empirical analysis of a modified Artificial Bee Colony for constrained numerical optimization», Applied Mathematics and Computation, vol. 218, p. 10 943–10 973. 105

[70] MINKA, T. P. 2001, «Expectation propagation for approximate Bayesian inference», dans Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence, p. 362–369. 25, 39

[71] MOČKUS, J. 1975, «On Bayesian methods for seeking the extremum», dans Proceedings of the Optimization Techniques IFIP Technical Conference, p. 400–404. 3, 19, 20, 21, 155

[72] MORÉ, J. J. et S. M. WILD. 2009, «Benchmarking Derivative-Free Optimization Algorithms», SIAM Journal on Optimization, vol. 20, p. 172–191. 105

[73] MUTNY, M. et A. KRAUSE. 2018, «Efficient high dimensional bayesian optimization with additivity and quadrature fourier features», dans Advances in Neural Information Processing Systems, p. 9005–9016. 26

[74] NAYEBI, A., A. MUNTEANU et M. POLOCZEK. 2019, «A Framework for Bayesian Optimization in Embedded Subspaces», dans Proceedings of the 36th International Conference on Machine Learning, p. 4752–4761. 26, 27, 28, 29, 32, 45, 54, 65, 76, 156

[75] NELDER, J. A. et R. MEAD. 1965, «A simplex method for function minimization», The computer journal, vol. 7, p. 308–313. 140

[76] NOCEDAL, J. et S. J. WRIGHT. 2006, Numerical Optimization, 2e éd., Springer Science & Business Media. 2, 36, 39

[77] OH, C., E. GAVVES et M. WELLING. 2018, «BOCK : Bayesian Optimization with Cylindrical Kernels», dans Proceedings of the 35th International Conference on Machine Learning, p. 3868–3877. 12, 26, 159

[78] OLIVEIRA, R., L. OTT et F. RAMOS. 2019, «Bayesian optimisation under uncertain inputs», dans Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, p. 1177–1184. 160

[79] PARR, J. M., A. J. KEANE, A. I. J. FORRESTER et C. M. E. HOLDEN. 2012, «Infill sampling criteria for surrogate-based optimization with constraint handling», Engineering Optimization, vol. 44, p. 1147–1166. 42, 65, 88, 89, 94, 95, 96, 105, 156

[80] PEDREGOSA, F., G. VAROQUAUX, A. GRAMFORT, V. MICHEL, B. THIRION, O. GRISEL, M. BLONDEL, P. PRETTENHOFER, R. WEISS, V. DUBOURG, J. VANDERPLAS, A. PASSOS, D. COURNAPEAU, M. BRUCHER, M. PERROT et E. DUCHESNAY. 2011, «Scikit-learn : Machine learning in Python», Journal of Machine Learning Research, vol. 12, p. 2825–2830. 65

[81] PELAMATTI, J., L. BREVAULT, M. BALESDENT, E. TALBI et Y. GUERIN. 2019, «Efficient global optimization of constrained mixed variable problems», Journal of Global Optimization, vol. 73, p. 583–613. 160

[82] PEREZ, R. E., P. W. JANSEN et J. R. R. A. MARTINS. 2012, «pyopt : a python-based object-oriented framework for nonlinear constrained optimization», Structural and Multidisciplinary Optimization, vol. 45, p. 101–118. 64, 92

[83] PICHENY, V. 2014, «A stepwise uncertainty reduction approach to constrained global optimization», dans Proceedings of the 17th Conference on Artificial Intelligence and Statistics, vol. 33, p. 787–795. 21, 24, 25, 36, 38, 87, 111, 132, 157

[84] PICHENY, V., R. B. GRAMACY, S. WILD et S. LE DIGABEL. 2016, «Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian», dans Advances in Neural Information Processing Systems, p. 1435–1443. 36, 39, 40, 87, 94, 96, 105, 111, 132, 157

[85] PIPERNI, P., M. ABDO et F. KAFYEKE. 2004, «The application of multi-disciplinary optimization technologies to the design of a business jet», dans Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, p. 4370. 133

[86] PIPERNI, P., M. ABDO, F. KAFYEKE et A. T. ISIKVEREN. 2007, «Preliminary aerostructural optimization of a large business jet», Journal of Aircraft, vol. 44, p. 1422–1438. 135

[87] PIPERNI, P. et J. BOUDREAU. 2003, «The evolution of structured grid generation at Bombardier Aerospace», dans Proceedings of the 9th Aerodynamics Symposium of the Canadian Aeronautics and Space Institute. 135

[88] PIPERNI, P., A. DEBLOIS et R. HENDERSON. 2013, «Development of a multilevel multidisciplinary-optimization capability for an industrial environment», AIAA Journal, vol. 51, p. 2335–2352. 133, 134, 135

[89] POURBAGIAN, M., B. TALGORN, W. G. HABASHI, M. KOKKOLARAS et S. LE DIGABEL. 2015, «Constrained problem formulations for power optimization of aircraft electro-thermal anti-icing systems», Optimization and Engineering, vol. 16, p. 663–693. 37

[90] POWELL, M. J. D. 1998, «Direct search algorithms for optimization calculations», Acta numerica, vol. 7, p. 287–336. 12, 111, 126, 157

[91] PRIEM, R., N. BARTOLI et Y. DIOUANE. 2019, «On the Use of Upper Trust Bounds in Constrained Bayesian Optimization Infill Criteria», dans Proceedings of the AIAA AVIATION 2019 FORUM, p. 2986. 4, 5

[92] PRIEM, R., N. BARTOLI, Y. DIOUANE, S. DUBREUIL et T. LEFEBVRE. 2018, «An adaptive feasibility approach for constrained bayesian optimization with application in aircraft design», dans Proceedings of the 6th International Conference on Engineering Optimization. 6, 121

[93] PRIEM, R., N. BARTOLI, Y. DIOUANE et H. GAGNON. 2019, «On a feasibility criterion for aircraft efficient global multidisciplinary optimization», dans Proceedings of the 1st MDOPhD Day. 5, 124

[94] PRIEM, R., N. BARTOLI, Y. DIOUANE, T. LEFEBVRE, S. DUBREUIL, M. SALAÜN et J. MORLIER. 2018, «SEGOMOE : Super Efficient Global Optimization with Mixture of Experts», dans Proceedings of the CIMI Optimization and Learning Workshop. 6, 121

[95] PRIEM, R., N. BARTOLI, Y. DIOUANE et A. SGUEGLIA. 2019, «An upper trust bound feasibility criterion for constrained bayesian optimization», dans Proceedings of the Optimization Days 2019. 5, 124

[96] PRIEM, R., N. BARTOLI, Y. DIOUANE et A. SGUEGLIA. 2020, «Upper trust bound feasibility criterion for mixed constrained Bayesian optimization with application to aircraft design», Aerospace Science and Technology, vol. 105, p. 105 980. 5, 121, 124

[97] PRIEM, R., H. GAGNON, I. CHITTICK, S. DUFRESNE, Y. DIOUANE et N. BARTOLI. 2020, «An efficient application of Bayesian optimization to an industrial MDO framework for aircraft design.», dans Proceedings of the AIAA AVIATION 2020 FORUM, VIRTUAL EVENT, p. 3152. 5, 124

[98] RASMUSSEN, C. E. et C. K. I. WILLIAMS. 2006, Gaussian Processes for Machine Learning, MIT Press. 3, 8, 9, 10, 11, 20

[99] RAYMER, D. P. 2006, Aircraft Design : A Conceptual Approach, AIAA Education Series, AIAA (American Institute of Aeronautics & Ast. 2, 125, 126, 155

[100] REGIS, R. G. 2014, «Evolutionary Programming for High-Dimensional Constrained Expensive Black-Box Optimization Using Radial Basis Functions», IEEE Transactions on Evolutionary Computation, vol. 18, p. 326–347. 105

[101] REGIS, R. G. 2016, «Trust regions in Kriging-Based optimization with expected improvement», Engineering optimization, vol. 48, p. 1037–1059. 33

[102] REIST, T. A., D. KOO, D. W. ZINGG, P. BOCHUD, P. CASTONGUAY et D. LEBLOND. 2019, «Cross-validation of high-fidelity aerodynamic shape optimization methodologies for aircraft wing-body optimization», AIAA Journal. 135

[103] ROLLAND, P., J. SCARLETT, I. BOGUNOVIC et V. CEVHER. 2018, «High-Dimensional Bayesian Optimization via Additive Models with Overlapping Groups», dans Proceedings of the 21st International Conference on Artificial Intelligence and Statistics, p. 298–307. 26

[104] ROSKAM, J. 2005, Airplane design part I : preliminary sizing of airplanes, 4e éd., DAR Corporation. 124, 125, 126, 129

[105] ROUSTANT, O., D. GINSBOURGER et Y. DEVILLE. 2012, «DiceKriging, DiceOptim : Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization», Journal of Statistical Software, vol. 51. 12, 38, 111

[106] ROUSTANT, O., E. PADONOU, Y. DEVILLE, A. CLÉMENT, G. PERRIN, J. GIORLA et H. WYNN. 2020, «Group kernels for Gaussian process metamodels with categorical inputs», SIAM/ASA Journal on Uncertainty Quantification, vol. 8, p. 775–806. 160

[107] RUNARSSON, T. P. et X. YAO. 2005, «Search biases in constrained evolutionary optimization», IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 35, p. 233–243. 64, 92

[108] SASENA, M. J., P. PAPALAMBROS et P. GOOVAERTS. 2002, «Exploration of metamodeling sampling criteria for constrained global optimization», Engineering optimization, vol. 34, p. 263–278. 36, 42, 64, 87, 93, 111, 126, 134, 156

[109] SCHITTKOWSKI, K. 1986, «NLPQL : A FORTRAN subroutine solving constrained nonlinear programming problems», Annals of operations research, vol. 5, p. 485–500. 140

[110] SCHMOLLGRUBER, P., J. BEDOUET, A. SGUEGLIA, S. DEFOORT, R. LAFAGE, N. BARTOLI, Y. GOURINAT et E. BENARD. 2017, «Use of a certification constraints module for aircraft design activities», dans Proceedings of the 17th AIAA Aviation Technology, Integration, and Operations Conference, p. 3762. 124, 158

[111] SCHONLAU, M. 1997, Computer Experiments and Global Optimization, thèse de doctorat, University of Waterloo, Canada. 36

[112] SCHONLAU, M., W. J. WELCH et D. R. JONES. 1998, «Global versus local search in constrained optimization of computer models», Lecture Notes-Monograph Series, vol. 34, p. 11–25. 36, 87, 111, 132, 157

[113] SCHWEFEL, H. P. 1975, Evolutionsstrategie Und Numerische Optimierung [Evolution Strategy and Numerical Optimization], PhD Thesis, Ph. D. thesis, Fachbereich Verfahrenstechnik, Technische Universität Berlin 140

[114] SERMEUS, K., K. MOHAMED, E. LAURENDEAU et S. NADARAJAH. 2010, «Development of an industrial aerodynamic shape optimization strategy using a discrete adjoint method», dans Proceedings of the 18th Annual Conference of the CFD Society of Canada, p. 6. 135

[115] SGUEGLIA, A. 2019, Exploration Du Dimensionnement et Priorités d'optimisation Dans La Conception Avion Avec Application à Une Aile Volante Avec Propulsion Électrique Distribuée, thèse de doctorat, ISAE. 124, 125

[116] SGUEGLIA, A., P. SCHMOLLGRUBER, N. BARTOLI, E. BENARD, J. MORLIER, J. JASA, J. R. R. A. MARTINS, J. T. HWANG et J. S. GRAY. 2020, «Multidisciplinary design optimization framework with coupled derivative computation for hybrid aircraft», Journal of Aircraft, p. 1–15. xiv, 124, 129

[117] SHAHRIARI, B., K. SWERSKY, Z. WANG, R. P. ADAMS et N. DE FREITAS. 2016, «Taking the Human Out of the Loop : A Review of Bayesian Optimization», Proceedings of the IEEE, vol. 104, p. 148–175. 3, 20, 21, 23, 36, 155

[118] SHILTON, A., S. GUPTA, S. RANA et S. VENKATESH. 2020, «Sequential Subspace Search for Functional Bayesian Optimization Incorporating Experimenter Intuition», arXiv :2009.03543. 84, 157, 159

[119] SILVER, D., T. HUBERT, J. SCHRITTWIESER, I. ANTONOGLOU, M. LAI, A. GUEZ, M. LANCTOT, L. SIFRE, D. KUMARAN et T. GRAEPEL. 2018, «A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play», Science, vol. 362, p. 1140–1144. 25

[120] SNELSON, E. et Z. GHAHRAMANI. 2006, «Variable noise and dimensionality reduction for sparse Gaussian processes», dans Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence, p. 461–468. 12

[121] SNOEK, J., H. LAROCHELLE et R. P. ADAMS. 2012, «Practical bayesian optimization of machine learning algorithms», dans Advances in Neural Information Processing Systems, p. 2951–2959. 22

[122] SNOEK, J., K. SWERSKY, R. ZEMEL et R. ADAMS. 2014, «Input warping for Bayesian optimization of non-stationary functions», dans Proceedings of the 31st International Conference on Machine Learning, p. 1674–1682. 12

[123] SPAGNOL, A., R. LE RICHE et S. DA VEIGA. 2019, «Global Sensitivity Analysis for Optimization with Variable Selection», SIAM/ASA Journal on Uncertainty Quantification, vol. 7, p. 417–443. 159

[124] SRINIVAS, N., A. KRAUSE, S. M. KAKADE et M. SEEGER. 2010, «Gaussian process optimization in the bandit setting : No regret and experimental design», dans Proceedings of the 27th International Conference on Machine Learning, p. 1015–1022. 21

[125] TENENBAUM, J. B., V. DE SILVA et J. C. LANGFORD. 2000, «A global geometric framework for nonlinear dimensionality reduction», Science, vol. 290, p. 2319–2323. 26, 159

[126] THOMPSON, W. R. 1933, «On the likelihood that one unknown probability exceeds another in view of the evidence of two samples», Biometrika, vol. 25, p. 285–294. 21

[127] TOAL, D. J. J., N. W. BRESSLOFF, A. J. KEANE et C. M. E. HOLDEN. 2011, «The development of a hybridized particle swarm for kriging hyperparameter tuning», Engineering optimization, vol. 43, p. 675–699. 12

[128] TOAL, D. J. J., A. I. J. FORRESTER, N. W. BRESSLOFF, A. J. KEANE et C. HOLDEN. 2009, «An adjoint for likelihood maximization», Proceedings of the Royal Society A : Mathematical, Physical and Engineering Sciences, vol. 465, p. 3267–3287. 12

[129] TRAN, A., S. MCCANN, J. M. FURLAN, K. V. PAGALTHIVARTHI, R. J. VISINTAINER et T. WILDEY. 2020, «aphBO-2GP-3B : A budgeted asynchronously-parallel multi-acquisition for known/unknown constrained Bayesian optimization on high-performing computing architecture», arXiv :2003.09436. 20, 159

[130] TREMBLAY, O. et L. A. DESSAINT. 2009, «Experimental validation of a battery dynamic model for ev applications», World Electric Vehicle Journal, vol. 3, p. 289–298. 126

[131] VALAREZO, W. et V. CHIN. 1994, «Method for the prediction of wing maximum lift», Journal of Aircraft, vol. 31, p. 103–109. 135

[132] VAN DER VELDEN, A. et P. KOCH. 2010, «Isight design optimization methodologies», ASM handbook, vol. 22, p. 79. 135, 140, 158

[133] VILLEMONTEIX, J., E. VAZQUEZ et E. WALTER. 2009, «An informational approach to the global optimization of expensive-to-evaluate functions», Journal of Global Optimization, vol. 44, p. 509–534. 21

[134] WANG, Z., F. HUTTER, M. ZOGHI, D. MATHESON et N. DE FEITAS. 2016, «Bayesian optimization in a billion dimensions via random embeddings», Journal of Artificial Intelligence Research, vol. 55, p. 361–387. xi, 26, 27, 28, 29, 31, 32, 45, 54, 61, 65, 156

[135] WANG, Z. et S. JEGELKA. 2017, «Max-value entropy search for efficient Bayesian optimization», dans Proceedings of the 34th International Conference on Machine Learning, vol. 70, p. 3627–3635. 20, 21, 26

[136] WANG, Z., C. LI, S. JEGELKA et P. KOHLI. 2018, «Batched High-dimensional Bayesian Optimization via Structural Kernel Learning», dans Proceedings of the 21st International Conference on Artificial Intelligence and Statistics, vol. 84, p. 745–754. 26, 66, 156, 159

[137] WATSON, A. G. et R. J. BARNES. 1995, «Infill sampling criteria to locate extremes», Mathematical Geology, vol. 27, p. 589–608. 21, 22

[138] WOLD, S., K. ESBENSEN et P. GELADI. 1987, «Principal component analysis», Chemometrics and intelligent laboratory systems, vol. 2, p. 37–52. 26

[139] ZHAN, D. et H. XING. 2020, «Expected improvement for expensive optimization : A review», Journal of Global Optimization. 21

[140] ZHANG, M., H. LI et S. SU. 2019, «High dimensional Bayesian optimization via supervised dimension reduction», dans Proceedings of the 28th International Joint Conference on Artificial Intelligence, p. 4292–4298. 159

[141] ZHU, C., R. H. BYRD, P. LU et J. NOCEDAL. 1997, «Algorithm 778 : L-BFGS-B : Fortran subroutines for large-scale bound-constrained optimization», ACM Transactions on Mathematical Software (TOMS), vol. 23, p. 550–560. 12

Annexe A

Contenus additionnels

A.1 Détails sur les approximations nécessaires aux MGP

Dans cette annexe, on détaille les calculs nécessaires à l'approximation de $\mathbb{P}\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{x}\right)$ de la Section 2.1.4.1.

On commence par une marginalisation sur $\boldsymbol{\theta}_s^{(l)}$ qui permet d'écrire la fonction de densité $p\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{x}\right)$ comme suit :

$$p\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{x}\right)=\int p\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)p\left(\boldsymbol{\theta}_s^{(l)}|\mathcal{D}_s^{(l)}\right)\mathrm{d}\boldsymbol{\theta}_s^{(l)}, \tag{A.1}$$

où $p\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)$ est la fonction de densité du GP de s en $\boldsymbol{x}$ construit avec les hyper-paramètres $\boldsymbol{\theta}_s^{(l)}$. On note la loi de ce GP en $\boldsymbol{x}$ comme suit :

$$\mathbb{P}\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)=\mathcal{GP}\left(\mu,k|\mathcal{D}_s^{(l)},\boldsymbol{x},\boldsymbol{\theta}_s^{(l)}\right)=\mathcal{N}\left(\hat{\mu}^{(l)}_{s,\boldsymbol{\theta}_s^{(l)}}(\boldsymbol{x}),\hat{\nu}^{(l)}_{s,\boldsymbol{\theta}_s^{(l)}}(\boldsymbol{x})\right), \tag{A.2}$$

où $\hat{\mu}^{(l)}_{s,\boldsymbol{\theta}_s^{(l)}}$ et $\hat{\nu}^{(l)}_{s,\boldsymbol{\theta}_s^{(l)}}(\boldsymbol{x})=\left[\hat{\sigma}^{(l)}_{s,\boldsymbol{\theta}_s^{(l)}}\right]^2$ sont respectivement la fonction de moyenne et de variance de ce GP.

Pour avoir une expression analytique de $p\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{x}\right)$, on cherche une approximation linéaire de $\mathbb{P}\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)$ donnée par $\mathbb{Q}\left(y_s,\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)$ de la forme :

$$\begin{aligned}\mathbb{P}\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)&=\mathcal{N}\left(\hat{\mu}^{(l)}_{s,\boldsymbol{\theta}_s^{(l)}}(\boldsymbol{x}),\hat{\nu}^{(l)}_{s,\boldsymbol{\theta}_s^{(l)}}(\boldsymbol{x})\right)\\&\approx\mathbb{Q}\left(y_s,\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)=\mathcal{N}\left(\boldsymbol{a}^\top\boldsymbol{\theta}_s^{(l)}+b,\nu\right),\end{aligned} \tag{A.3}$$

en utilisant le vecteur $\boldsymbol{a}\in\mathbb{R}^{d_e\cdot d}$, $b\in\mathbb{R}$ et $\nu\in\mathbb{R}$ comme paramètres de l'approximation. On note $q\left(y_s,\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)$ la densité de $\mathbb{Q}\left(y_s,\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)$. Associée à l'approximation gaussienne de la loi des hyper-paramètres (2.14), on obtient une expression analytique de (A.1) telle que :

$$\mathbb{P}\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{x}\right)\approx\mathcal{N}\left(\boldsymbol{a}^\top\hat{\boldsymbol{\theta}}_s^{(l)}+b,\nu+\boldsymbol{a}^\top\hat{\boldsymbol{\Sigma}}\boldsymbol{a}\right), \tag{A.4}$$

où $\hat{\boldsymbol{\theta}}_s^{(l)}=\mathrm{vect}\left(\hat{\mathbf{A}}\right)$. Ensuite, on trouve $\boldsymbol{a}$, b et ν en faisant correspondre une expansion locale de $q\left(y_s,\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)$ à $p\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)$. Cette expansion est réalisée en $\boldsymbol{\theta}_s^{(l)}=\hat{\boldsymbol{\theta}}_s^{(l)}$ et en $y_s=\hat{y}_s$. Le choix de $\hat{y}_s$ est expliqué plus loin dans le document. Explicitement, cette expansion donne les équations

suivantes :

$$q\left(y_s,\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)\Big|_{\hat{\boldsymbol{\theta}}_s^{(l)},\hat{y}_s} = p\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)\Big|_{\hat{\boldsymbol{\theta}}_s^{(l)},\hat{y}_s}, \tag{A.5}$$

$$\frac{\partial q\left(y_s,\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)}{\partial y_s}\Bigg|_{\hat{\boldsymbol{\theta}}_s^{(l)},\hat{y}_s} = \frac{\partial p\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)}{\partial y_s}\Bigg|_{\hat{\boldsymbol{\theta}}_s^{(l)},\hat{y}_s}, \tag{A.6}$$

$$\frac{\partial q\left(y_s,\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)}{\partial \theta_{s,i}^{(l)}}\Bigg|_{\hat{\boldsymbol{\theta}}_s^{(l)},\hat{y}_s} = \frac{\partial p\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)}{\partial \theta_{s,i}^{(l)}}\Bigg|_{\hat{\boldsymbol{\theta}}_s^{(l)},\hat{y}_s}, \tag{A.7}$$

$$\frac{\partial^2 q\left(y_s,\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)}{\partial y_s \partial \theta_{s,i}^{(l)}}\Bigg|_{\hat{\boldsymbol{\theta}}_s^{(l)},\hat{y}_s} = \frac{\partial^2 p\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)}{\partial y_s \partial \theta_{s,i}^{(l)}}\Bigg|_{\hat{\boldsymbol{\theta}}_s^{(l)},\hat{y}_s}, \tag{A.8}$$

$$\frac{\partial^2 q\left(y_s,\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)}{\partial y_s \partial y_s}\Bigg|_{\hat{\boldsymbol{\theta}}_s^{(l)},\hat{y}_s} = \frac{\partial^2 p\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{\theta}_s^{(l)},\boldsymbol{x}\right)}{\partial y_s \partial y_s}\Bigg|_{\hat{\boldsymbol{\theta}}_s^{(l)},\hat{y}_s}. \tag{A.9}$$

On remarque que l'on ne prend pas en compte les dérivées secondes en $\boldsymbol{\theta}_s^{(l)}$ car elles sont trop nombreuses et engendreraient un coût de calcul trop important.

A partir de (A.5), (A.6), (A.7), (A.8) et (A.9), on cherche maintenant à trouver les vecteurs $\boldsymbol{a}$, b et ν.

— Premièrement, avec (A.5), (A.6) et (A.9), on a

$$\boldsymbol{a}^\top \hat{\boldsymbol{\theta}}_s^{(l)} + b = \hat{\mu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}, \tag{A.10}$$

$$\nu = \hat{\nu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}. \tag{A.11}$$

— Deuxièmement, avec (A.7) et (A.8), on a :

$$2a_i = \frac{\partial \hat{\nu}_{y_s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}}{\partial \theta_{s,i}^{(l)}}\left(\frac{1}{\hat{y}_s - \hat{\mu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}} - \frac{\hat{y}_s - \hat{\mu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}}{\hat{\nu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}}\right) + 2\frac{\partial \hat{\mu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}}{\partial \theta_{s,i}^{(l)}}, \tag{A.12}$$

$$2a_i = 2\frac{\partial \hat{\nu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}}{\partial \theta_{s,i}^{(l)}}\left(\frac{\hat{y}_s - \hat{\mu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}}{\hat{\nu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}}\right) + 2\frac{\partial \hat{\mu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}}{\partial \theta_{s,i}^{(l)}}, \tag{A.13}$$

où a_i est la i$^{\text{e}}$ composante de $\boldsymbol{a}$, $i \in \{1,\dots,d_e \cdot d\}$ et

$$\frac{\partial \hat{\mu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}}{\partial \theta_{s,i}^{(l)}} = \frac{\partial \hat{\mu}_{s,\boldsymbol{\theta}_s^{(l)}}^{(l)}}{\partial \theta_{s,i}^{(l)}}\Bigg|_{\boldsymbol{\theta}_s^{(l)}=\hat{\boldsymbol{\theta}}_s^{(l)}}, \quad \frac{\partial \hat{\nu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}}{\partial \theta_{s,i}^{(l)}} = \frac{\partial \hat{\nu}_{s,\boldsymbol{\theta}_s^{(l)}}^{(l)}}{\partial \theta_{s,i}^{(l)}}\Bigg|_{\boldsymbol{\theta}_s^{(l)}=\hat{\boldsymbol{\theta}}_s^{(l)}}. \tag{A.14}$$

Pour finir, (A.12) et (A.13) ne peuvent être résolues que pour deux valeurs de $\hat{y}_s$ donnant deux valeurs pour les composantes de $\boldsymbol{a}$:

$$a_i = \pm\frac{1}{\sqrt{3\hat{\nu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}}}\frac{\partial \hat{\nu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}}{\partial \theta_{s,i}^{(l)}} + \frac{\partial \hat{\mu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}}{\partial \theta_{s,i}^{(l)}}, \quad \hat{y}_s = \hat{\mu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)} \pm \sqrt{\frac{\hat{\nu}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}^{(l)}}{3}}, \quad i \in \{1,\dots,d_e \cdot d\}. \tag{A.15}$$

On choisit ces deux valeurs particulières de $\hat{y}_s$ car le choix intuitif $\hat{y}_s = \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}^{(l)}_s}$ donne des contraintes non consistantes dans (A.7).

Ainsi, on est capable de définir une approximation de $\mathbb{P}\left(y_s|\mathcal{D}^{(l)}_s, \boldsymbol{x}\right)$ pour chacune des valeurs possibles de $\boldsymbol{a}$ et $\hat{y}_s$:

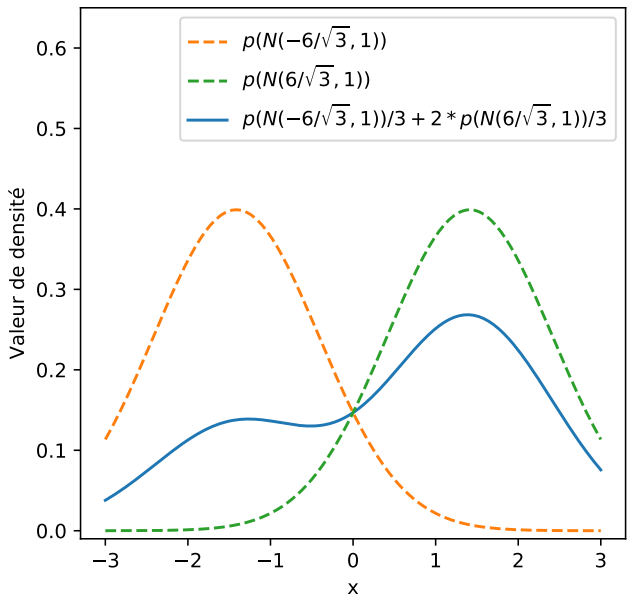
$$\mathbb{P}\left(y_s|\mathcal{D}^{(l)}_s, \boldsymbol{x}\right) \approx \mathcal{N}\left(\boldsymbol{a}^\top \hat{\boldsymbol{\theta}}^{(l)}_s + b \pm \sqrt{\frac{\hat{\nu}^{(l)}_{s,\hat{\boldsymbol{\theta}}^{(l)}_s}}{3}}, \nu + \boldsymbol{a}^\top \hat{\boldsymbol{\Sigma}} \boldsymbol{a}\right). \tag{A.16}$$

On moyenne ensuite les densités de ces deux lois pour obtenir une unique approximation de $\mathbb{P}\left(y_s|\mathcal{D}^{(l)}_s, \boldsymbol{x}\right)$. Par ce biais, on produit un mélange de densités gaussiennes. La théorie sur le mélange de densités gaussiennes nécessaire à la compréhension du document est présentée ci-dessous.

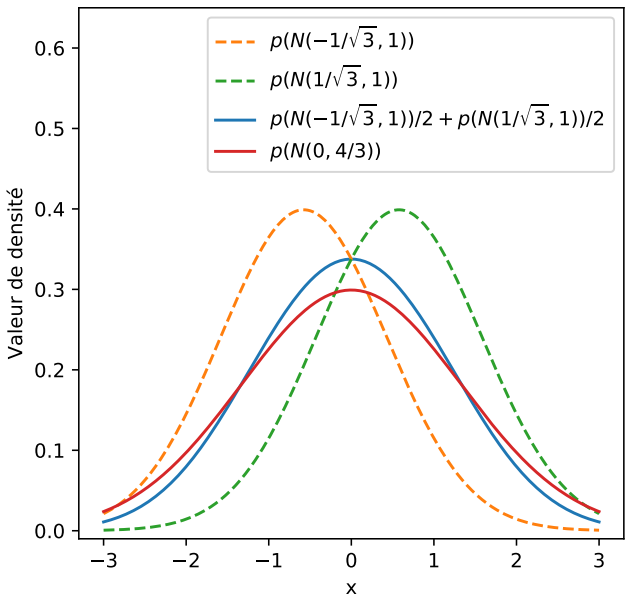
En général, un mélange de densités gaussiennes s'écrit comme suit :

$$p(\mathrm{X}) = \sum_{i=1}^{\mathrm{N}} \alpha_i p_i(\mathrm{X}), \tag{A.17}$$

où $\alpha_i \in [0,1]$, $\sum_{i=1}^{\mathrm{N}} \alpha_i = 1$, $p_i(\mathrm{X})$ la densité $\mathcal{N}(\mu_i, \sigma_i^2)$ et $\mathrm{N} \in \mathbb{N}^+$ le nombre de densités dans le mélange. Par exemple, la Figure A.1a montre un mélange de deux densités gaussiennes. On remarque les deux modes qui correspondent respectivement à chacune des densités gaussiennes du mélange. De plus, on est capable de calculer analytiquement la moyenne et la variance liée à la fonc-



(a) Exemple d'un mélange de densités gaussiennes.



(b) Exemple de l'approximation réalisée pour la construction d'un MGP.

FIGURE A.1 – Exemple d'un mélange de densités gaussiennes et de l'approximation réalisée pour la construction d'un MGP.

tion de densité de $p(\mathrm{X})$. Elles sont données par

$$\mathbb{E}_p[\mathrm{X}] = \sum_{i=1}^{\mathrm{N}} \alpha_i \mathbb{E}_{p_i}[\mathrm{X}] = \sum_{i=1}^{\mathrm{N}} \alpha_i \mu_i, \tag{A.18}$$

$$\begin{aligned} \mathbb{VAR}_p[\mathrm{X}] &= \mathbb{E}_p[\mathrm{X}^2] - \mathbb{E}_p[\mathrm{X}]^2 \\ &= \sum_{i=1}^{\mathrm{N}} \alpha_i \mathbb{E}_{p_i}(\mathrm{X}^2) - \left(\sum_{i=1}^{\mathrm{N}} \alpha_i \mathbb{E}_{p_i}[\mathrm{X}] \right)^2 \\ &= \sum_{i=1}^{\mathrm{N}} \alpha_i (\sigma_i + \mu_i^2) - \left(\sum_{i=1}^{\mathrm{N}} \alpha_i \mu_i \right)^2. \end{aligned} \tag{A.19}$$

Grâce à l'estimation (A.16), on donne une approximation gaussienne $\mathcal{N}\left(\tilde{\mu}_s^{(l)}, \left[\tilde{\sigma}_s^{(l)}\right]^2\right)$ de la loi à mélange de densités gaussiennes $\mathbb{P}\left(y_s | \mathcal{D}_s^{(l)}, \boldsymbol{\theta}_s^{(l)}, \boldsymbol{x}\right)$ en identifiant la moyenne $\tilde{\mu}_s^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}$ et l'écart type $\tilde{\sigma}_s^{(l)} : \mathbb{R}^d \mapsto \mathbb{R}$ de l'approximation avec la moyenne (A.18) et l'écart type (A.19). On rappelle que les deux lois gaussiennes utilisées sont

$$\mathcal{N}_1\left(\boldsymbol{a}^\top \hat{\boldsymbol{\theta}}_s^{(l)} + b + \sqrt{\frac{\hat{\mathrm{v}}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}{3}}, \mathrm{v} + \boldsymbol{a}^\top \hat{\boldsymbol{\Sigma}} \boldsymbol{a}\right), \quad \mathcal{N}_2\left(\boldsymbol{a}^\top \hat{\boldsymbol{\theta}}_s^{(l)} + b - \sqrt{\frac{\hat{\mathrm{v}}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}{3}}, \mathrm{v} + \boldsymbol{a}^\top \hat{\boldsymbol{\Sigma}} \boldsymbol{a}\right). \tag{A.20}$$

En utilisant (A.15) , on obtient :

$$\boldsymbol{a}^\top \hat{\boldsymbol{\theta}}_s^{(l)} + b \pm \sqrt{\frac{\hat{\mathrm{v}}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}{3}} = \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} \pm \sqrt{\frac{\hat{\mathrm{v}}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}{3}}, \tag{A.21}$$

$$\mathrm{v} + \boldsymbol{a}^\top \hat{\boldsymbol{\Sigma}} \boldsymbol{a} = \hat{\mathrm{v}}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} + \left[\nabla \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}\right]^\top \hat{\boldsymbol{\Sigma}} \nabla \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} + \frac{1}{3\hat{\mathrm{v}}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}} \left[\nabla \hat{\mathrm{v}}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}\right]^\top \hat{\boldsymbol{\Sigma}} \nabla \hat{\mathrm{v}}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}. \tag{A.22}$$

Les fonctions de densité associées à ces lois sont ensuite combinées avec un facteur de $\alpha_i = 1/2$ pour $i = \{1,2\}$ pour créer la fonction de densité du mélange de gaussienne. Grâce à (A.18) et (A.19), on est capable de trouver la moyenne et la variance de cette lois à mélange de densité gaussienne. En utilisant $\alpha_i = 1/2$, (A.18) et (A.19) deviennent respectivement

$$\tilde{\mu}_s^{(l)} = 0.5\mu_1 + 0.5\mu_2, \tag{A.23}$$

$$\left[\tilde{\sigma}_s^{(l)}\right]^2 = 0.5\sigma_1 + 0.5\sigma_2 + 0.5\mu_1^2 + 0.5\mu_2^2 - (0.5\mu_1 + 0.5\mu_2)^2. \tag{A.24}$$

En remplaçant μ_1, μ_2, σ_1 et σ_2 par les valeurs données en (A.21) et (A.22), on obtient :

$$\tilde{\mu}_s^{(l)} = 0.5\left(\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} + \sqrt{\frac{\hat{\mathrm{v}}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}{3}}\right) + 0.5\left(\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} - \sqrt{\frac{\hat{\mathrm{v}}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}{3}}\right) = \boxed{\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}, \tag{A.25}$$

$$
\begin{aligned}
\left[\tilde{\sigma}_s^{(l)}\right]^2 &= \nu + \boldsymbol{a}^\top \hat{\boldsymbol{\Sigma}} \boldsymbol{a} + 0.5 \left(\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} + \sqrt{\frac{\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}{3}} \right)^2 + 0.5 \left(\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} - \sqrt{\frac{\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}{3}} \right)^2 - \left[\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} \right]^2 \\
&= \nu + \boldsymbol{a}^\top \hat{\boldsymbol{\Sigma}} \boldsymbol{a} + \cancel{\left[\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} \right]^2} + \frac{\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}{3} + \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} \cancel{\sqrt{\frac{\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}{3}}} - \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} \cancel{\sqrt{\frac{\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}{3}}} - \cancel{\left[\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} \right]^2} \\
&= \hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} + \left[\nabla \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} \right]^\top \hat{\boldsymbol{\Sigma}} \nabla \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} + \frac{1}{3\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}} \left[\nabla \hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} \right]^\top \hat{\boldsymbol{\Sigma}} \nabla \hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} + \frac{\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}{3} \\
&\boxed{= \frac{4}{3} \hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} + \left[\nabla \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} \right]^\top \hat{\boldsymbol{\Sigma}} \nabla \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} + \frac{1}{3\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}} \left[\nabla \hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} \right]^\top \hat{\boldsymbol{\Sigma}} \nabla \hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}.}
\end{aligned}
\tag{A.26}
$$

Pour finir on approche $\mathbb{P}\left(y_s | \mathcal{D}_s^{(l)}, \boldsymbol{x}\right)$ par une loi gaussienne $\mathcal{N}\left(\tilde{\mu}_s^{(l)}, \tilde{v}_s^{(l)}\right)$ dont la moyenne est donnée par :

$$
\tilde{\mu}_s^{(l)} = \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}, \tag{A.27}
$$

$$
\left[\tilde{\sigma}_s^{(l)}\right]^2 = \tilde{v}_s^{(l)} = \frac{4}{3} \hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} + \left[\nabla \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} \right]^\top \hat{\boldsymbol{\Sigma}} \nabla \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} + \frac{1}{3\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}} \left[\nabla \hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} \right]^\top \hat{\boldsymbol{\Sigma}} \nabla \hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}, \tag{A.28}
$$

où $\hat{\boldsymbol{\Sigma}}$ et $\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}$ sont respectivement données par (2.16) et (A.2). Les dérivées de la moyenne $\nabla \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}$ et de la variance $\nabla \hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}$ ont une expression analytique qui dépend du noyau de covariance et de la moyennes a priori choisis. La Figure A.1b montre un exemple de l'approximation réalisée par GARNETT et al. [40] pour passer d'un mélange de densités gaussiennes à une unique densité gaussienne. En vert et en orange sont représentées les deux densités de probabilité utilisées dans la mélange de gaussiennes. La densité de probabilité résultant du mélange de ces deux densités gaussiennes est tracée en bleu. En rouge, on représente l'approximation gaussienne de cette probabilité. La même approximation est réalisée lors de la construction des MGP. On remarque qu'une erreur est commise entre la densité rouge et la densité bleue mais que la forme générale de la densité bleue est conservée.

Pour finir, on rappelle les 5 grandes étapes nécessaires à la construction d'un MGP.

1. Choix de la moyenne a priori $\text{vect}\left(\mathbf{A}_p\right)$ et de la covariance $\boldsymbol{\Sigma}_p$ de la matrice vectorisée $\text{vect}\left(\mathbf{A}\right)$. Grandeurs obtenues : $\text{vect}\left(\mathbf{A}_p\right)$ et $\boldsymbol{\Sigma}_p$.
2. Maximisation de la densité de probabilité a posteriori de $\text{vect}\left(\mathbf{A}\right)$ en utilisant les dérivées du log vraisemblance (2.11) pour trouver la moyenne $\text{vect}\left(\hat{\mathbf{A}}\right)$ de la probabilité a posteriori de $\text{vect}\left(\mathbf{A}\right)$. Grandeur obtenue : $\text{vect}\left(\hat{\mathbf{A}}\right)$.
3. Calcul de la matrice de covariance $\hat{\boldsymbol{\Sigma}}$ de la probabilité a posteriori de $\text{vect}\left(\mathbf{A}\right)$ grâce à la Hessienne du log vraisemblance (2.17). Grandeur obtenue : $\hat{\boldsymbol{\Sigma}}$.
4. Construction d'un GP en imposant les hyper-paramètres à $\hat{\boldsymbol{\theta}}_s^{(l)} = \text{vect}\left(\hat{\mathbf{A}}\right)$. Grandeurs obtenues : $\hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}$ et $\hat{v}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}$.
5. Prise en compte de l'incertitude des hyper-paramètres sur la construction du MGP de fonction de densité $p\left(y_s | \mathcal{D}_s^{(l)}, \boldsymbol{x}\right)$ grâce à une marginalisation sur $\boldsymbol{\theta}_s^{(l)}$ (A.1). Grandeur estimée : $p\left(y_s | \mathcal{D}_s^{(l)}, \boldsymbol{x}\right)$.

(a) Approximation linéaire de la loi du GP de s en $\boldsymbol{x}$ construit avec les hyper-paramètres $\boldsymbol{\theta}_s^{(l)}$ (A.3). Grandeur estimée : $\mathbb{P}\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{x}\right) \approx \mathcal{N}\left(\boldsymbol{a}^\top \hat{\boldsymbol{\theta}}_s^{(l)} + b, \nu + \boldsymbol{a}^\top \hat{\boldsymbol{\Sigma}} \boldsymbol{a}\right)$.

i. Estimation des paramètres de cette loi par expansion autour de $\boldsymbol{\theta}_s^{(l)} = \hat{\boldsymbol{\theta}}_s^{(l)}$ où $\hat{\boldsymbol{\theta}}_s^{(l)} = \text{vect}\left(\hat{\mathbf{A}}\right)$ et $y_s = \hat{y}_s$ (A.5), (A.6), (A.7), (A.8) et (A.9). Grandeurs obtenues : $\boldsymbol{a}$, b, ν et $\hat{y}_s$.

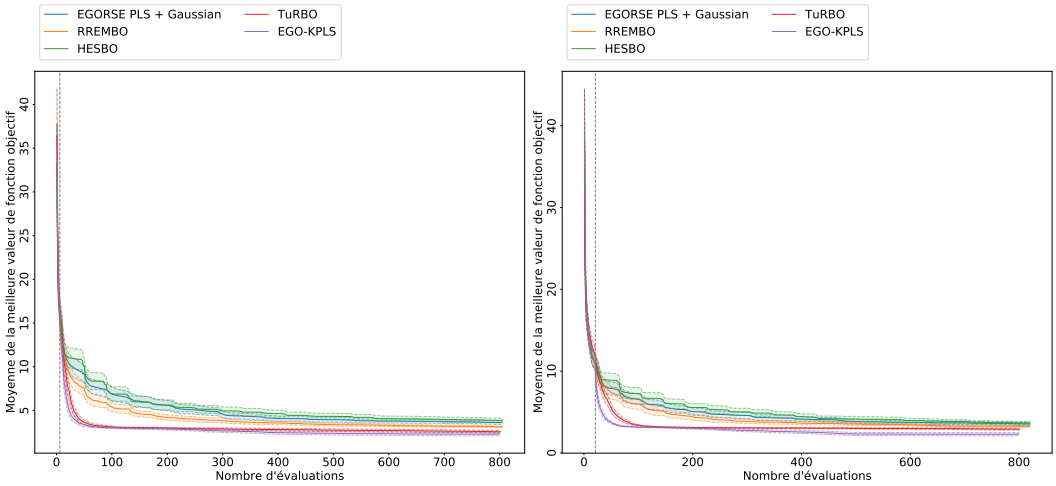
ii. Deux solutions possibles pour $p\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{x}\right)$ (A.16) à cause d'une expansion autour de $\hat{y}_s = \hat{\mu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}} \pm \sqrt{\frac{\hat{\nu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}{3}}$ (A.15). Grandeurs obtenues :

$$\mathbb{P}\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{x}\right) \approx \mathcal{N}\left(\boldsymbol{a}^\top \hat{\boldsymbol{\theta}}_s^{(l)} + b \pm \sqrt{\frac{\hat{\nu}^{(l)}_{s,\hat{\boldsymbol{\theta}}_s^{(l)}}}{3}}, \nu + \boldsymbol{a}^\top \hat{\boldsymbol{\Sigma}} \boldsymbol{a}\right).$$

(b) Obtention d'une unique fonction de densité par mélange des deux densités gaussiennes obtenues. Grandeur obtenue : $p\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{x}\right)$.

(c) Approximation par une loi gaussienne de cette loi de mélange de densités gaussiennes (A.28) et (A.27). Grandeur estimée : $p\left(y_s|\mathcal{D}_s^{(l)},\boldsymbol{x}\right)$.

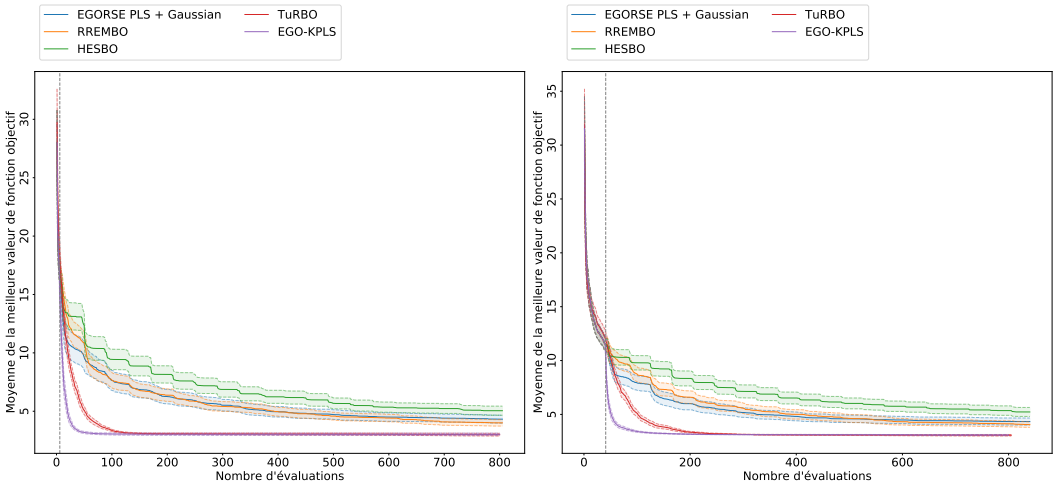
A.2 Courbes de convergence pour EGORSE, RREMBO, TuRBO, HESBO, SEGO-KPLS pour des DOE initiaux comprenant 5 et d points.



(a) DOE de 5 points.

(b) DOE de $2d$ points.

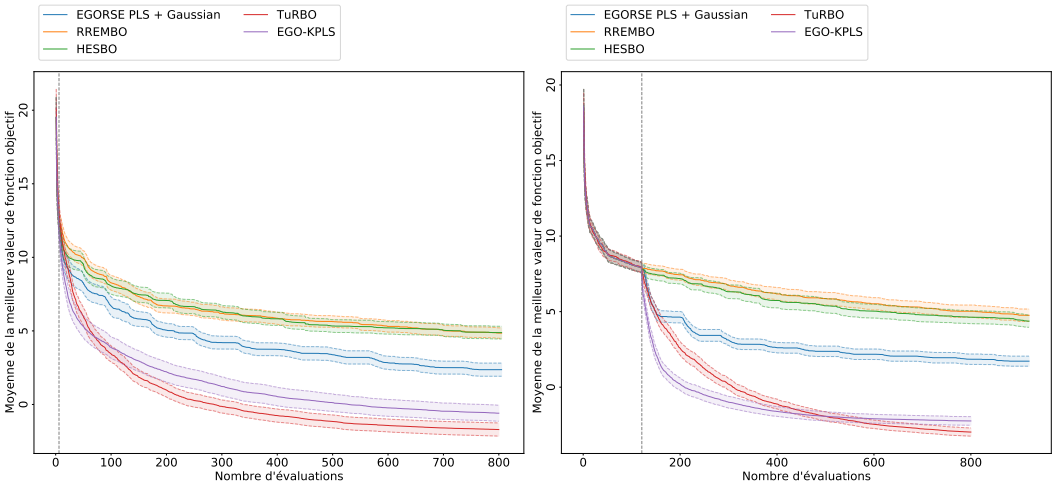
FIGURE A.2 – Courbes de convergence de HESBO, RREMBO, TuRBO, SEGO-KPLS et EGORSE pour le problème MB_10.



(a) DOE de 5 points.

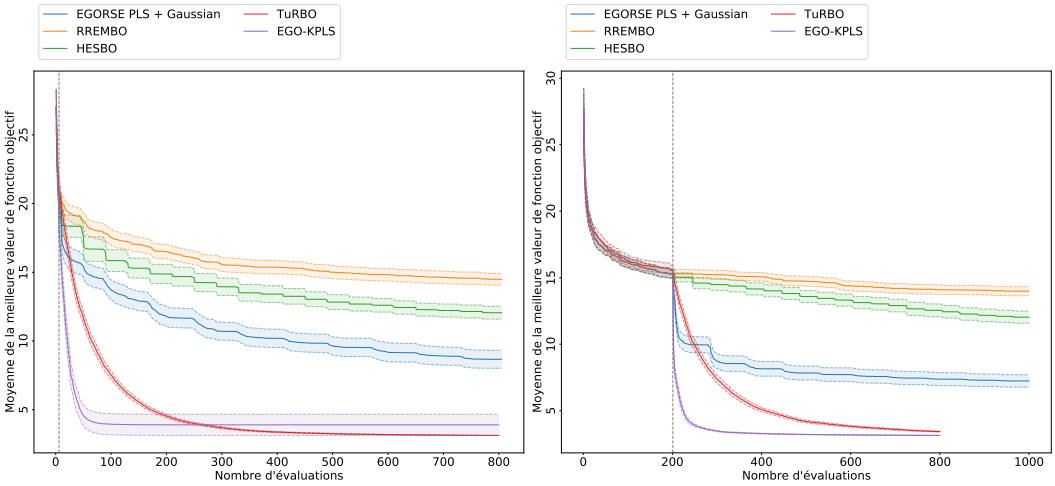
(b) DOE de $2d$ points.

FIGURE A.3 – Courbes de convergence de HESBO, RREMBO, TuRBO, SEGO-KPLS et EGORSE pour le problème MB_20.



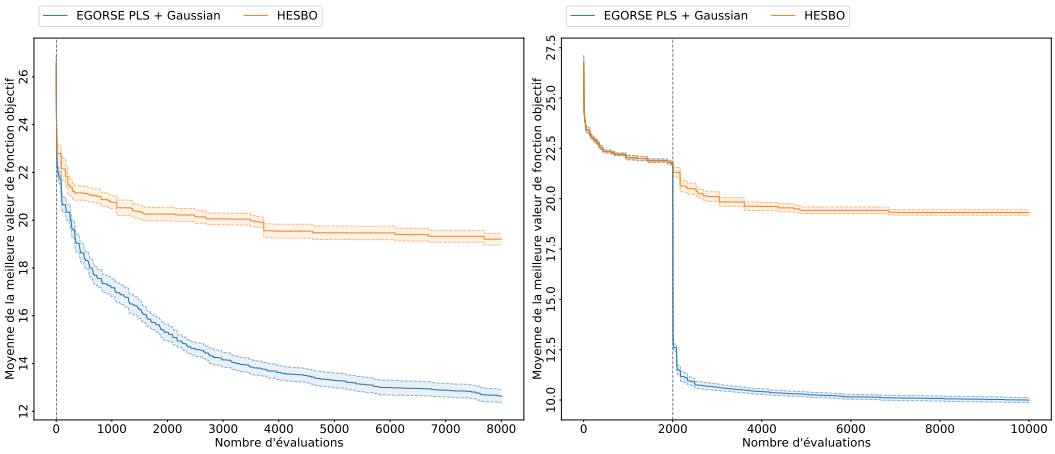
(a) DOE de 5 points. (b) DOE de $2d$ points.

FIGURE A.4 – Courbes de convergence de HESBO, RREMBO, TuRBO, SEGO-KPLS et EGORSE pour le problème Rover_60.



(a) DOE de 5 points. (b) DOE de $2d$ points.

FIGURE A.5 – Courbes de convergence de HESBO, RREMBO, TuRBO, SEGO-KPLS et EGORSE pour le problème MB_100.



(a) DOE de 5 points. (b) DOE de $2d$ points.

FIGURE A.6 – Courbes de convergence de HESBO et EGORSE pour le problème MB_1000.

A.3 L'ensemble des 29 problèmes de l'étude.

A.3.1 GSBP

$$\begin{cases} \min & \frac{\log\left[\left(1+a(4x_1+4x_2-3)^2\right)\left(30+b(8x_1-12x_2+2)^2\right)\right]-8.69}{2.43}, \\ \text{s.à} & g_1(\boldsymbol{x}) = 0.5\sin\left(2\pi\left(x_1^2-2x_2\right)\right)+x_1+2x_2-1.5 \geq 0, \\ & h_1(\boldsymbol{x}) = -x_1^2-x_2^2+1.5 = 0, \\ & h_2(\boldsymbol{x}) = 15-\left(15x_2-\frac{5}{4\pi^2}(15x_1-5)^2+\frac{5}{\pi}(15x_1-5)-6\right)^2 \\ & -10\left(1-\frac{1}{8\pi}\right)\cos(15x_1-5) = 0, \\ \text{avec} & a = 75-56(x_1+x_2)+3(4x_1-2)^2+6(4x_1-2)(4x_2-2)+3(4x_2-2)^2, \\ & b = -14-128x_1+12(4x_1-2)^2+192x_2-36(4x_1-2)(4x_2-2)+27(4x_2-2)^2, \\ & 0 \leq x_i \leq 1,\ \forall i \in \{0,1\} \end{cases} \tag{A.29}$$

A.3.2 GTCD

$$\begin{cases} \min & 8{,}61\cdot 10^5 x_1^{1/2}x_2x_3^{-2/3}x_4^{-1/2}+3{,}69\cdot 10^4 x_3+7{,}72\cdot 10^8 x_1^{-1}x_2^{0.219}-765{,}43\cdot 10^6 x_1^{-1}, \\ \text{s.à} & g_1(\boldsymbol{x}) = x_4x_2^{-2}+x_2^{-2}-1 \leq 0, \\ & 20 \leq x_1 \leq 50,\ 1 \leq x_2 \leq 10,\ 20 \leq x_3 \leq 50,\ 0.1 \leq x_2 \leq 60 \end{cases} \tag{A.30}$$

A.3.3 Hesse

$$\begin{cases} \min & -25(x_1-2)^2-(x_2-2)^2-(x_3-1)^2-(x_4-4)^2-(x_5-1^2)-(x_6-4)^2, \\ \text{s.à} & g_1(\boldsymbol{x}) = \frac{2-x_1-x_2}{2} \leq 0, \\ & g_2(\boldsymbol{x}) = \frac{x_1+x_2-6}{6} \leq 0, \\ & g_3(\boldsymbol{x}) = \frac{-x_1+x_2-2}{2} \leq 0, \\ & g_4(\boldsymbol{x}) = \frac{x_1-3x_2-2}{2} \leq 0, \\ & g_5(\boldsymbol{x}) = \frac{4-(x_3-3)^2-x_4}{4} \leq 0, \\ & g_6(\boldsymbol{x}) = \frac{4-(x_5-3)^2-x_6}{4} \leq 0, \\ & 0 \leq x_1 \leq 5,\ 0 \leq x_2 \leq 4,\ 1 \leq x_3 \leq 5,\ 0 \leq x_4 \leq 6,\ 1 \leq x_5 \leq 5,\ 0 \leq x_6 \leq 10 \end{cases} \tag{A.31}$$

A.3.4 LAH

$$\begin{cases} \min & x_1+x_2+x_3+x_4, \\ \text{s.à} & g_1(\boldsymbol{x}) = 20\exp\left(-0.2\sqrt{\frac{1}{4}\sum_{i=1}^{4}(3x_i-1)^2}\right)+\exp\left(\frac{1}{4}\sum_{i=1}^{4}\cos\left(2\pi(3x_i-1)\right)\right)\leq 0, \\ & h_1(\boldsymbol{x}) = \frac{1}{0.8387}\left[-1.1+\sum_{i=1}^{4}\mathrm{C}_i\exp\left(-\sum_{j=1}^{4}a_{ij}(x_j-p_{ij})^2\right)\right]=0, \\ \text{avec} & \boldsymbol{a} = \begin{bmatrix} 10.00 & 0.05 & 3.00 & 17.00 \\ 3.00 & 10.00 & 3.50 & 8.00 \\ 17.00 & 17.00 & 1.70 & 0.05 \\ 3.50 & 0.10 & 10.00 & 10.00 \end{bmatrix}, \quad \boldsymbol{p} = \begin{bmatrix} 0.131 & 0.232 & 0.234 & 0.404 \\ 0.169 & 0.413 & 0.145 & 0.882 \\ 0.556 & 0.830 & 0.352 & 0.873 \\ 0.012 & 0.373 & 0.288 & 0.574 \end{bmatrix}, \\ & \mathbf{C} = \begin{bmatrix} 1.0 & 1.2 & 3.0 & 3.2 \end{bmatrix}, 0\leq x_i\leq 1, \forall i\in\{1,2,3,4\} \end{cases} \tag{A.32}$$

A.3.5 LSQ

$$\begin{cases} \min & x_1+x_2, \\ \text{s.à} & g_1(\boldsymbol{x}) = 0.5\sin\left(2\pi\left(x_1^2-2x_2\right)\right)+x_1+2x_2-1.5\geq 0, \\ & g_2(\boldsymbol{x}) = -x_1^2-x_2^2+1.5\geq 0, \\ & 0\leq x_i\leq 1, \forall i\in\{1,2\} \end{cases} \tag{A.33}$$

A.3.6 MB

$$\begin{cases} \min & \left[\left(x_2-\frac{5.1x_1^2}{4\pi^2}+\frac{5x_1}{\pi}-6\right)^2+\left(10-\frac{10}{8\pi}\right)\cos(x_1)+1\right]+\frac{5x_1+25}{15}, \\ \text{s.à} & g_1(\boldsymbol{x}) = 6-\left(4-2.1\overline{x}_1^2+\frac{\overline{x}_1^4}{3}\right)\overline{x}_1^2-\overline{x}_1\overline{x}_2-\left(4\overline{x}_2^2-4\right)\overline{x}_2^2 \\ & -3\sin\left(6(1-\overline{x}_1)\right)-3\sin\left(6(1-\overline{x}_2)\right)\leq 0, \\ \text{avec} & \overline{x}_1 = {}^{(x_1-2.5)}/_{7.5} \text{ et } \overline{x}_2 = {}^{(x_2-7.5)}/_{7.5}, \\ & -5\leq x_1\leq 10, 0\leq x_1\leq 15 \end{cases} \tag{A.34}$$

A.3.7 MB_eq

$$\begin{cases} \min & \left[\left(x_2-\frac{5.1x_1^2}{4\pi^2}+\frac{5x_1}{\pi}-6\right)^2+\left(10-\frac{10}{8\pi}\right)\cos(x_1)+1\right]+\frac{5x_1+25}{15}, \\ \text{s.à} & h_1(\boldsymbol{x}) = 6-\left(4-2.1\overline{x}_1^2+\frac{\overline{x}_1^4}{3}\right)\overline{x}_1^2-\overline{x}_1\overline{x}_2-\left(4\overline{x}_2^2-4\right)\overline{x}_2^2 \\ & -3\sin\left(6(1-\overline{x}_1)\right)-3\sin\left(6(1-\overline{x}_2)\right)=0, \\ \text{avec} & \overline{x}_1 = {}^{(x_1-2.5)}/_{7.5} \text{ et } \overline{x}_2 = {}^{(x_2-7.5)}/_{7.5} \\ & -5\leq x_1\leq 10, 0\leq x_1\leq 15 \end{cases} \tag{A.35}$$

A.3.8 PVD4

$$\begin{cases} \min & 0{,}062\,24x_1x_3x_4 + 1{,}778\,1x_2x3^2 + 3{,}166\,1x_1^2x_4 + 19.84x_1^2x_3, \\ \text{s.à} & g_1(\boldsymbol{x}) = -x1 + 0{,}019\,3x_3 \le 0, \\ & g_2(\boldsymbol{x}) = -x2 + 0{,}009\,54x_3 \le 0, \\ & g_3(\boldsymbol{x}) = plog(-\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1\,296\,000) \le 0, \\ \text{avec} & \text{si } x \ge 0,\ plog(x) = \log(1+x)\,;\ \text{sinon } plog(x) = -\log(1-x), \\ & 0 \le x_1, x_2 \le 1,\ 0 \le x_3 \le 50,\ 0 \le x_4 \le 240 \end{cases} \tag{A.36}$$

A.3.9 SR7

$$\begin{cases} \min & 0.7854x_1x_2^2\mathrm{A} - 1.508x_1\mathrm{B} + 7.477\mathrm{C} + 0.7854\mathrm{D}, \\ \text{s.à} & g_1(\boldsymbol{x}) = \frac{27 - x_1x_2^2x_3}{27} \le 0, \\ & g_2(\boldsymbol{x}) = \frac{397.5 - x_1x_2^2x_3^2}{397.5} \le 0, \\ & g_3(\boldsymbol{x}) = \frac{1.93 - x_2x_6^4x_3x_4^{-3}}{1.93} \le 0, \\ & g_4(\boldsymbol{x}) = \frac{1.93 - x_2x_7^4x_3x_5^{-3}}{1.93} \le 0, \\ & g_5(\boldsymbol{x}) = \frac{\mathrm{A}_1\mathrm{B}_1^{-1} - 1\,100}{1\,100} \le 0, \\ & g_6(\boldsymbol{x}) = \frac{\mathrm{A}_2\mathrm{B}_2^{-1} - 850}{850} \le 0, \\ & g_7(\boldsymbol{x}) = \frac{x_2x_3 - 40}{40} \le 0, \\ & g_8(\boldsymbol{x}) = \frac{5 - x_1x_2^{-1}}{5} \le 0, \\ & g_9(\boldsymbol{x}) = \frac{x_1x_2^{-1} - 12}{12} \le 0, \\ & g_{10}(\boldsymbol{x}) = \frac{1.9 + 1.5x_6 - x_4}{1.9} \le 0, \\ & g_{11}(\boldsymbol{x}) = \frac{1.9 + 1.1x_7 - x_5}{1.9} \le 0, \\ \text{avec} & \mathrm{A} = 3.3333x_3^2 + 14.9334x_3 - 43.0934, \\ & \mathrm{B} = x_6^2 + x_7^2, \\ & \mathrm{C} = x_6^3 + x_7^3, \\ & \mathrm{D} = x_4x_6^2 + x_5x_7^2, \\ & \mathrm{A}_1 = \sqrt{(745x_4x_2^{-1}x_3^{-1})^2 + 16{,}91 \cdot 10^6}, \\ & \mathrm{A}_2 = \sqrt{(745x_5x_2^{-1}x_3^{-1})^2 + 157{,}5 \cdot 10^6}, \\ & \mathrm{B}_1 = 0.1x_6^3,\ \mathrm{B}_2 = 0.1x_7^3, \\ & 2.6 \le x_1 \le 3.6,\ 0.7 \le x_2 \le 0.8,\ 17 \le x_3 \le 28,\ 7.3 \le x_4, x_5 \le 8.3, \\ & 2.9 \le x_6 \le 3.9,\ 5 \le x_7 \le 5.5 \end{cases} \tag{A.37}$$

A.3.10 WB4

$$\begin{cases} \min & 1{,}104\,71 x_1^2 x_2 + 4{,}811 \cdot 10^{-2} x_3 x_4 (14 + x_2), \\ \text{s.à} & g_1(\boldsymbol{x}) = \frac{t - 1{,}36 \cdot 10^4}{1{,}36 \cdot 10^4} \leq 0,\ g_2(\boldsymbol{x}) = \frac{s - 3 \cdot 10^4}{3 \cdot 10^4} \leq 0, \\ & g_3(\boldsymbol{x}) = \frac{x_1 - x_4}{10} \leq 0,\ g_4(\boldsymbol{x}) = \frac{0{,}104\,71 x_1^2 + 0{,}048\,11 x_3 x_4 (14 + x_2) - 5}{5} \leq 0, \\ & g_5(\boldsymbol{x}) = \frac{d - 0.25}{0.25} \leq 0,\ g_6(\boldsymbol{x}) = \frac{6000 - \mathrm{P}_c}{6000} \leq 0, \\ \text{avec} & \mathrm{M} = 8{,}4 \cdot 10^4 + 3\,000 x_2,\ \mathrm{R} = \sqrt{0.25 (x_2^2 + (x_1 + x_3)^2)}, \\ & \mathrm{J} = \sqrt{2} x_1 x_2 \left(x_2^2 / 12 + 0.25 (x_1 + x_3)^2 \right),\ \mathrm{P}_c = 1{,}023\,724\,5 \cdot 10^5 x_3 x_4^3 (1 - 2{,}445\,2 \cdot 10^{-2} x_3), \\ & t_1 = \frac{6000}{\sqrt{2} x_1 x_2},\ t_2 = \mathrm{MRJ}^{-1},\ t = \sqrt{t_1^2 + t_1 t_2 \mathrm{R}^{-1} x_2 + t_2^2},\ s = \frac{5{,}04 \cdot 10^5}{x_4 x_3^2},\ d = \frac{2.1952}{x_4 x_3^3}, \\ & 0.125 \leq x_1 \leq 10,\ 0.1 \leq x_i \leq 10,\ \forall i \in \{2, 3, 4\} \end{cases} \tag{A.38}$$

A.3.11 G03

$$\begin{cases} \min & -\left(\sqrt{10}\right)^{10} \prod_{i=1}^{10} x_i., \\ \text{s.à} & h(\boldsymbol{x}) = \sum_{i=1}^{10} x_i^2 - 1 = 0, \\ & 0 \leq x_i \leq 1, \forall i \in \{1, \ldots, 10\} \end{cases} \tag{A.39}$$

A.3.12 G04

$$\begin{cases} \min & 5.3578547 x_3^2 + 0.8356891 x_1 x_5 + 37.293239 x_1 - 40792.141, \\ \text{s.à} & g_1(\boldsymbol{x}) = 85.334407 + 0.0056858 x_2 x_5 + 0.0006262 x_1 x_4 - 0.0022053 x_3 x_5 - 92 \leq 0, \\ & g_2(\boldsymbol{x}) = -85.334407 - 0.0056858 x_2 x_5 - 0.0006262 x_1 x_4 + 0.0022053 x_3 x_5 \leq 0, \\ & g_3(\boldsymbol{x}) = 80.51249 + 0.0071317 x_2 x_5 + 0.0029955 x_1 x_2 + 0.0021813 x_3^2 - 110 \leq 0, \\ & g_4(\boldsymbol{x}) = -80.51249 - 0.0071317 x_2 x_5 - 0.0029955 x_1 x_2 - 0.0021813 x_3^2 + 90 \leq 0, \\ & g_5(\boldsymbol{x}) = 9.300961 + 0.0047026 x_3 x_5 + 0.0012547 x_1 x_3 + 0.0019085 x_3 x_4 - 25 \leq 0, \\ & g_6(\boldsymbol{x}) = -9.300961 - 0.0047026 x_3 x_5 - 0.0012547 x_1 x_3 - 0.0019085 x_3 x_4 + 20 \leq 0, \\ & 78 \leq x_1 \leq 102,\ 33 \leq x_2 \leq 45,\ 27 \leq x_i \leq 45,\ \forall i \in \{3, 4, 5\} \end{cases} \tag{A.40}$$

A.3.13 G05

$$\begin{cases} \min & 3 x_1 + 0.000001 x_1^3 + 2 x_2 + \frac{0.000002}{3} x_2^3, \\ \text{s.à} & g_1(\boldsymbol{x}) = -x_4 + x_3 - 0.55 \leq 0, \\ & g_2(\boldsymbol{x}) = -x_3 + x_4 - 0.55 \leq 0, \\ & h_3(\boldsymbol{x}) = 1000 \sin - x_3 - 0.25 + + 1000 \sin - x_4 - 0.25 + + 894.8 - x_1 = 0, \\ & h_4(\boldsymbol{x}) = 1000 \sin x_3 - 0.25 + + 1000 \sin x_3 - x_4 - 0.25 + + 894.8 - x_2 = 0, \\ & h_5(\boldsymbol{x}) = 1000 \sin x_4 - 0.25 + + 1000 \sin x_4 - x_3 - 0.25 + + 1294.8 = 0, \\ & 0 \leq x_1 \leq 1\,200,\ 0 \leq x_2 \leq 1\,200,\ -0.55 \leq x_3 \leq 0.55,\ -0.55 \leq x_4 \leq 0.55 \end{cases} \tag{A.41}$$

A.3.14 G06

$$\begin{cases} \min & (x_1-10)^3+(x_2-20)^3, \\ \text{s.à} & g_1(\boldsymbol{x})=-(x_1-5)^2-(x_2-5)^2+100\leq 0, \\ & g_2(\boldsymbol{x})=(x_1-6)^2+(x_2-5)^2-82.81\leq 0, \\ & 13\leq x_1\leq 100,\ 0\leq x_2\leq 100 \end{cases} \tag{A.42}$$

A.3.15 G07

$$\begin{cases} \min & x_1^2+x_2^2+x_1x_2-14x_1-16x_2+(x_3-10)^2+4(x_4-5)^2+(x_5-3)^2+2(x_6-1)^2+5x_7^2 \\ & +7(x_8-11)^2+2(x_9-10)^2+(x_{10}-7)^2, \\ \text{s.à} & g_1(\boldsymbol{x})=-105+4x_1+5x_2-3x_7+9x_8\leq 0, \\ & g_2(\boldsymbol{x})=10x_1-8x_2-17x_7+2x_8\leq 0, \\ & g_3(\boldsymbol{x})=-8x_1+2x_2+5x_9-2x_{10}-12\leq 0, \\ & g_4(\boldsymbol{x})=3(x_1-2)2+4(x_2-3)2+2x_3^2-7x_4-120\leq 0, \\ & g_5(\boldsymbol{x})=5x_1^2+8x_2+(x_3-6)2-2x_4-40\leq 0, \\ & g_6(\boldsymbol{x})=x_1^2+2(x_2-2)2-2x_1x_2+14x_5-6x_6\leq 0, \\ & g_7(\boldsymbol{x})=0.5(x_1-8)2+2(x_2-4)2+3x_5^2-x_6-30\leq 0, \\ & g_8(\boldsymbol{x})=-3x_1+6x_2+12(x_9-8)2-7x_{10}\leq 0, \\ & -10\leq x_i\leq 10,\ \forall i\in\{1,\ldots,10\} \end{cases} \tag{A.43}$$

A.3.16 G08

$$\begin{cases} \min & -\frac{[\sin(2\pi x_1)]^3\sin(2\pi x_2)}{x_1^3(x_1+x_2)}, \\ \text{s.à} & g_1(\boldsymbol{x})=x_1^2-x_2+1\leq 0, \\ & g_2(\boldsymbol{x})=1-x_1+(x_2+4)^2\leq 0, \\ & 0\leq x_1\leq 10,\ 0\leq x_2\leq 10 \end{cases} \tag{A.44}$$

A.3.17 G09

$$\begin{cases} \min & (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2+10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7, \\ \text{s.à} & g_1(\boldsymbol{x})=-127+2x_1^2+3x_2^4+x3+4x_4^2+5x5\leq 0, \\ & g_2(\boldsymbol{x})=-282+7x1+3x2+10x_3^2+x4-x5\leq 0, \\ & g_3(\boldsymbol{x})=-282+7x1+3x2+10x_3^2+x4-x5\leq 0, \\ & g_4(\boldsymbol{x})=4x_1^2+x_2^2-3x1x2+2x_3^2+5x6-11x7\leq 0, \\ & -10\leq x_i\leq 10,\ \forall i\in\{1,\ldots,7\} \end{cases} \tag{A.45}$$

A.3.18 G10

$$\begin{cases} \min & x_1 + x_2 + x_3, \\ \text{s.à} & g_1(\boldsymbol{x}) = -1 + 0.0025(x_4 + x_6) \leq 0, \\ & g_2(\boldsymbol{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0, \\ & g_3(\boldsymbol{x}) = -1 + 0.01(x_8 - x_5) \leq 0, \\ & g_4(\boldsymbol{x}) = -x_1 x_6 + 833.33252 x_4 + 100 x_1 - 83333.333 \leq 0, \\ & g_5(\boldsymbol{x}) = -x_2 x_7 + 1250 x_5 + x_2 x_4 - 1250 x_4 \leq 0, \\ & g_6(\boldsymbol{x}) = -x_3 x_8 + 1250000 + x_3 x_5 - 2500 x_5 \leq 0, \\ & -10 \leq x_i \leq 10, \, \forall i \in \{1, \ldots, 10\} \end{cases} \tag{A.46}$$

A.3.19 G11

$$\begin{cases} \min & x_1^2 + (x_2 - 1)^2, \\ \text{s.à} & h_1(\boldsymbol{x}) = x_2 - x_1^2 = 0, \\ & -1 \leq x_i \leq 1, \, \forall i \in \{1, 2\} \end{cases} \tag{A.47}$$

A.3.20 G12

$$\begin{cases} \min & -\frac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100}, \\ \text{s.à} & g_1(\boldsymbol{x}) = \min\left\{(x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625, \{p, q, r\} \in \{1, \ldots, 9\}^3\right\} \leq 0, \\ & 0 \leq x_i \leq 10, \, \forall i \in \{1, 2, 3\} \end{cases} \tag{A.48}$$

A.3.21 G13

$$\begin{cases} \min & \exp x_1 x_2 x_3 x_4 \\ \text{s.à} & h_1(\boldsymbol{x}) = x_1^2 + x_2^2 + x - 3^2 + x_4^2 + x_5^2 - 10 = 0, \\ & h_2(\boldsymbol{x}) = x_2 x_3 - 5 x_4 x_5 = 0, \\ & h_3(\boldsymbol{x}) = x_1^3 + x_2^3 + 1 = 0, \\ & -2.3 \leq x_i \leq 2.3, \, \forall i \in \{1, 2\}, \; -3.2 \leq x_i \leq 3.2, \, \forall i \in \{3, 4, 5\} \end{cases} \tag{A.49}$$

A.3.22 G14

$$\begin{cases} \min & \sum_{i=1}^{10} x_i \left(c_i + \log\left(\frac{x_i}{\sum_{j=1}^{10} x_j} \right) \right) \\ \text{s.à} & h_1(\boldsymbol{x}) = x_1 + + x_2 + 2x_3 + x_6 + x_{10} - 2 = 0, \\ & h_2(\boldsymbol{x}) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0, \\ & h_3(\boldsymbol{x}) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0, \\ & 0 \leq x_i \leq 10, \, \forall i \in \{1, \ldots, 10\}, \; c_1 = -6.089, \, c_2 = -17.164, \, c_3 = -34.054, \\ & c_4 = -5.914, \, c_5 = -24.721, \, c_6 = -14.986, \, c7 = -24.1, \\ & c_8 = -10.708, \, c_9 = -26.662, \, c_{10} = -22.179 \end{cases} \tag{A.50}$$

A.3.23 G15

$$\begin{cases} \min & 1\,000 - x_1^2 - 2x_2^2 - x - 3^2 - x_1x_2 - x_1x_3, \\ \text{s.à} & h_1(\boldsymbol{x}) = x_1^2 + x_2^2 + x_3^2 - 25 = 0, \\ & h_2(\boldsymbol{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0, \\ & h_3(\boldsymbol{x}) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0, \\ & 0 \leq x_i \leq 10,\ \forall i \in \{1,\ldots,10\} \end{cases} \tag{A.51}$$

A.3.24 G17

$$\begin{cases} \min & f_1(x_1) + f_2(x_2) \\ \text{s.à} & h_1(\boldsymbol{x}) = -x_1 + 300 - \frac{x_3x_4}{131.078}\cos(1{,}484\,77 - x_6) + \frac{0{,}907\,98x_3^2}{131.078}\cos(1{,}475\,88) = 0, \\ & h_2(\boldsymbol{x}) = -x_2 - \frac{x_3x_4}{131.078}\cos(1{,}484\,77 + x_6) + \frac{0{,}907\,98x_4^2}{131.078}\cos(1{,}475\,88) = 0, \\ & h_3(\boldsymbol{x}) = -x_5 - \frac{x_3x_4}{131.078}\sin(1{,}484\,77 + x_6) + \frac{0{,}907\,98x_4^2}{131.078}\sin(1{,}475\,88) = 0, \\ & h_4(\boldsymbol{x}) = 200 - \frac{x_3x_4}{131.078}\sin(1{,}484\,77 - x_6) + \frac{0{,}907\,98x_4^2}{131.078}\sin(1{,}475\,88) = 0, \\ \text{avec} & \text{si } x_1 < 300,\ f_1x_1 = 30x_1\,;\ \text{sinon } f_1x_1 = 31x_1 \\ & \text{si } x_2 < 100,\ f_2x_2 = 28x_2\,;\ \text{si } 100 \leq x_2 < 100,\ f_2x_2 = 29x_2\,;\ \text{sinon } f_1x_1 = 30x_2 \\ & 0 \leq x_1 \leq 400,\ 0 \leq x_2 \leq 1000,\ 340 \leq x_3 \leq 420,\ 340 \leq x_4 \leq 420, \\ & -1000 \leq x_5 \leq 1000,\ 0 \leq x_6 \leq 0.5236 \end{cases} \tag{A.52}$$

A.3.25 G18

$$\begin{cases} \min & -0.5(x_1x_4 - x_2x_3 + x_3x_9 - x_5x_9 + x_5x_8 - x_6x_7) \\ \text{s.à} & g_1(\boldsymbol{x}) = x_3^2 + x_4^2 - 1 \leq 0, \\ & g_2(\boldsymbol{x}) = x_9^2 - 1 \leq 0, \\ & g_3(\boldsymbol{x}) = x_5^2 + x_6^2 - 1 \leq 0, \\ & g_4(\boldsymbol{x}) = x_1^2 + (x_2 - x_9)^2 - 1 \leq 0, \\ & g_5(\boldsymbol{x}) = (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1 \leq 0, \\ & g_6(\boldsymbol{x}) = (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \leq 0, \\ & g_7(\boldsymbol{x}) = (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1 \leq 0, \\ & g_8(\boldsymbol{x}) = (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1 \leq 0, \\ & g_9(\boldsymbol{x}) = x_7^2 + (x_8 - x_9)^2 - 1 \leq 0, \\ & g_{10}(\boldsymbol{x}) = x_2x_3 - x_1x_4 \leq 0, \\ & g_{11}(\boldsymbol{x}) = -x_3x_9 \leq 0, \\ & g_{12}(\boldsymbol{x}) = x_5x_9 \leq 0, \\ & g_{13}(\boldsymbol{x}) = x_6x_7 - x_5x_9 \leq 0, \\ & -10 \leq x_i \leq 10,\ \forall i \in \{1,\ldots,8\},\ 0 \leq x_9 \leq 20 \end{cases} \tag{A.53}$$

A.3.26 G16

$$
\left\{
\begin{array}{ll}
\min & 0{,}000\,117\,y_{14} + 0{,}136\,5 + 0{,}000\,023\,58\,y_{13} + 0{,}000\,001\,502\,y_{16} + 0{,}032\,1\,y_{12} \\
& +0{,}004\,324\,y_5 + 0{,}000\,1\frac{c_{15}}{c_{16}} + 37.48\frac{y_2}{c_{12}} - 0{,}000\,000\,584\,3\,y_{17}, \\
\text{s.à} & g_1(\boldsymbol{x}) = \frac{0.28}{0.72}y_5 - y_4 \leq 0,\ g_2(\boldsymbol{x}) = x_3 - 1.5x_2 \leq 0, \\
& g_3(\boldsymbol{x}) = 3\,496\frac{y_2}{c_1 2} - 21 \leq 0,\ g_4(\boldsymbol{x}) = 110.6 + y_1 - \frac{62\,212}{c_{17}} \leq 0, \\
& g_5(\boldsymbol{x}) = 213.1 - y_1 \leq 0,\ g_6(\boldsymbol{x}) = y_1 - 405.23 \leq 0, \\
& g_7(\boldsymbol{x}) = 17.505 - y_2 \leq 0,\ g_1 8(\boldsymbol{x}) = y_2 - 1\,053{,}666\,7 \leq 0, \\
& g_9(\boldsymbol{x}) = 11.275 - y_3 \leq 0,\ g_{10}(\boldsymbol{x}) = y_3 - 35.03 \leq 0, \\
& g_{11}(\boldsymbol{x}) = 214.228 - y_4 \leq 0,\ g_{12}(\boldsymbol{x}) = y_4 - 665.585 \leq 0, \\
& g_{13}(\boldsymbol{x}) = 7.458 - y_5 \leq 0,\ g_{14}(\boldsymbol{x}) = y_5 - 584.463 \leq 0, \\
& g_{15}(\boldsymbol{x}) = 0.961 - y_6 \leq 0,\ g_{16}(\boldsymbol{x}) = y_6 - 265.916 \leq 0, \\
& g_{17}(\boldsymbol{x}) = 1.612 - y_7 \leq 0,\ g_{18}(\boldsymbol{x}) = y_7 - 7.046 \leq 0, \\
& g_{19}(\boldsymbol{x}) = 0.146 - y_8 \leq 0,\ g_{20}(\boldsymbol{x}) = y_8 - 0.222 \leq 0, \\
& g_{21}(\boldsymbol{x}) = 107.99 - y_9 \leq 0,\ g_{22}(\boldsymbol{x}) = y_9 - 273.366 \leq 0, \\
& g_{23}(\boldsymbol{x}) = 922.693 - y_{10} \leq 0,\ g_{24}(\boldsymbol{x}) = y_{10} - 1\,286{,}105 \leq 0, \\
& g_{25}(\boldsymbol{x}) = y_{11} - 1\,444{,}046 \leq 0,\ g_{26}(\boldsymbol{x}) = 926.32 - y_{11} \leq 0, \\
& g_{27}(\boldsymbol{x}) = 18.766 - y_{12} \leq 0,\ g_{28}(\boldsymbol{x}) = y_{12} - 537.141 \leq 0, \\
& g_{29}(\boldsymbol{x}) = 1\,072{,}163 - y_{13} \leq 0,\ g_{30}(\boldsymbol{x}) = y_{13} - 3\,247{,}039 \leq 0, \\
& g_{31}(\boldsymbol{x}) = 8\,961{,}448 - y_{14} \leq 0,\ g_{32}(\boldsymbol{x}) = y_{14} - 26\,844{,}086 \leq 0, \\
& g_{33}(\boldsymbol{x}) = 0.063 - y_{15} \leq 0,\ g_{34}(\boldsymbol{x}) = y_{15} - 0.386 \leq 0, \\
& g_{35}(\boldsymbol{x}) = 71\,084{,}33 - y_{16} \leq 0,\ g_{36}(\boldsymbol{x}) = y_{16} - 140\,000 \leq 0, \\
& g_{37}(\boldsymbol{x}) = 2\,802\,713 - y_{17} \leq 0,\ g_{38}(\boldsymbol{x}) = y_{17} - 12\,146\,108 \leq 0, \\
\text{avec} & y_1 = x_2 + x_3 + 41.6,\ c_1 = 0.024x_4 - 4.62, \\
& y_2 = \frac{12.5}{c_1} + 12,\ c_2 = 0{,}000\,353\,5x_1^2 + 0{,}531\,1x_1 + 0{,}087\,05y_2x_1, \\
& c_3 = 0.052x_1 + 78 + 0{,}002\,377y_2x_1,\ y_3 = \frac{c_2}{c_3}, \\
& y_4 = 19y_3,\ c_4 = 0{,}047\,82(x_1 - y_3) + \frac{0{,}195\,6(x_1 - y_3)^2}{x_2} + 0{,}637\,9y_4 + 1.594y_3, \\
& c_5 = 100x_2,\ c_6 = x_1 - y_3 - y_4, \\
& c7 = 0.950 - \frac{c_4}{c_5},\ y_5 = c_6c_7, \\
& y_6 = x_1 - y_5 - y_4 - y_3,\ c_8 = 0.995(y_5 + y_4), \\
& y_7 = \frac{c_8}{y_1},\ y_8 = \frac{c_8}{3\,798}, \\
& c_9 = y_7 - \frac{0{,}066\,3y_7}{y_8} - 0{,}315\,3,\ y_9 = \frac{96.82}{c_9} + 0.321y_1, \\
& y_{10} = 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6,\ y_{11} = 1.71x_1 - 0.452y_4 + 0.580y_3, \\
& c_{10} = \frac{12.3}{752.3},\ c_{11} = c_{10}x_1 + \frac{c_{11}}{c_{12}}, \\
& c_{12} = 0.995y_{10} + 1\,998,\ y_{12} = c_{10}x_1 + \frac{c_{11}}{c_{12}}, \\
& y_{13} = c_{12} + 1.75y_2,\ y_{14} = 3\,623 + 64.4x_2 + 58.4x_3 + \frac{146\,312}{y_9 + x_5}, \\
& c_{13} = 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5\,095,\ y_{15} = \frac{y_{13}}{c_{13}}, \\
& y_{16} = 148\,000 - 331\,000y_{15} + 40y_{13} - 61y_{15}y_{13},\ c_{14} = 2\,324y_{10} - 28\,740\,000y_2, \\
& y_{17} = 14\,130\,000 - 1\,328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}},\ c_{15} = \frac{y_{13}}{y_{15}} - \frac{y13}{0.52}, \\
& c_{16} = 1.104 - 0.72y_{15},\ c_{17} = y_9 + x_5, \\
& 704.4148 \leq x_1 \leq 906.3855,\ 68.6 \leq x_2 \leq 288.88,\ 0 \leq x_3 \leq 134.75, \\
& 193 \leq x_4 \leq 287.0966,\ 25 \leq x_5 \leq 54.1988
\end{array}
\right. \tag{A.54}
$$

A.3.27 G21

$$\begin{cases} \min & x_1, \\ \text{s.à} & g_1(\boldsymbol{x}) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leq 0, \\ & h_1(\boldsymbol{x}) = -300x_3 + 7\,500x_5 - 7\,500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0, \\ & h_2(\boldsymbol{x}) = 100x_2 + 155.365x_4 + 2\,500x_7 - x_2x_4 - 24x_4x_7 - 15\,536{,}5 = 0, \\ & h_3(\boldsymbol{x}) = -x_5 + \log(-x_4 + 900) = 0, \\ & h_4(\boldsymbol{x}) = -x_6 + \log(x_4 + 300) = 0, \\ & h_5(\boldsymbol{x}) = -x_7 + \log(-2x_4 + 700) = 0, \\ & 0 \leq x_1 \leq 1\,000, 0 \leq x_2, x_3 \leq 40, 100 \leq x_4 \leq 300, \\ & 6.3 \leq x_5 \leq 6.7, 5.9 \leq x_6 \leq 6.4, 4.5 \leq x_7 \leq 6.25 \end{cases} \tag{A.55}$$

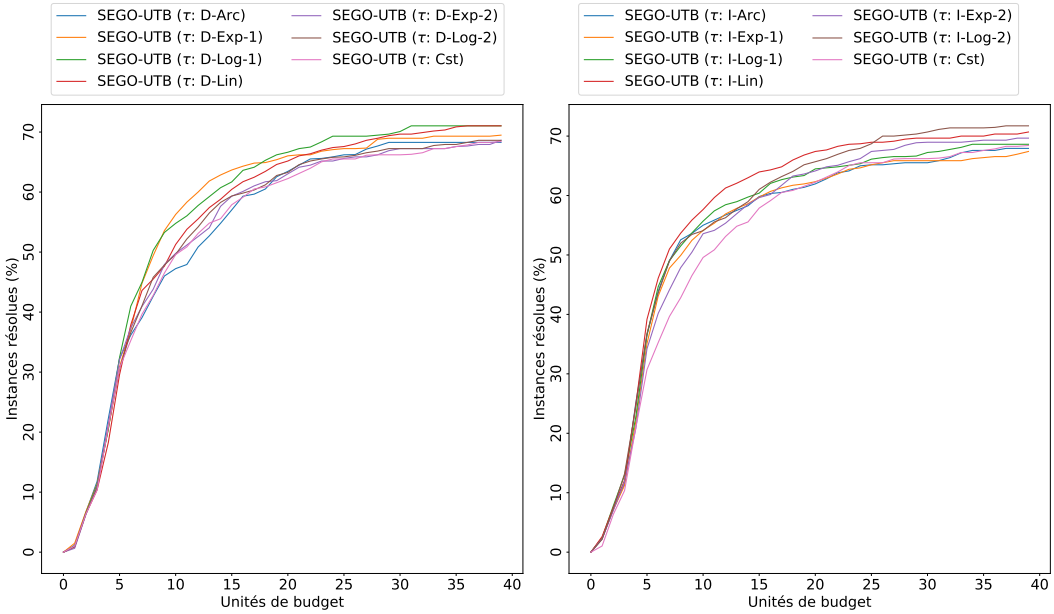
A.3.28 G23

$$\begin{cases} \min & -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7), \\ \text{s.à} & g_1(\boldsymbol{x}) = x_9x_3 + 0.02x_6 - 0.025x_5 \leq 0, \\ & g_2(\boldsymbol{x}) = x_9x_4 + 0.02x_7 - 0.015x_8 \leq 0, \\ & h_1(\boldsymbol{x}) = x_1 + x_2 - x_3 - x_4 = 0, \\ & h_2(\boldsymbol{x}) = 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0, \\ & h_3(\boldsymbol{x}) = x_3 + x_6 - x_5 = 0, \\ & h_4(\boldsymbol{x}) = x_4 + x_7 - x_8 = 0, \\ & 0 \leq x_1, x2, x6 \leq 300, 0 \leq x_3, x_5, x_7 \leq 100, 0 \leq x_4, x_8 \leq 200, 0.01 \leq x_9 \leq 0.03 \end{cases} \tag{A.56}$$

A.3.29 G24

$$\begin{cases} \min & -x_1 - x2, \\ \text{s.à} & g_1(\boldsymbol{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0, \\ & g_2(\boldsymbol{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0, \\ & 0 \leq x_1 \leq 3, 0 \leq x_2 \leq 4 \end{cases} \tag{A.57}$$

A.4 Data Profiles pour les différents seuil de doute avec une violation des contraintes $\epsilon_c = 10^{-2}$ avec noyau de covariance gaussien.

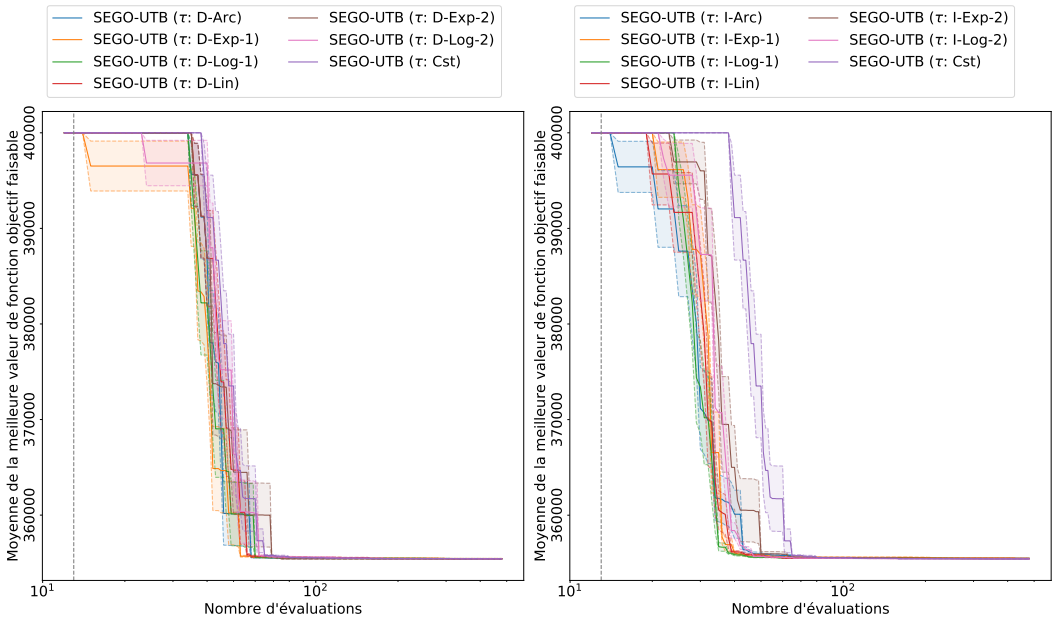


(a) Seuil de doute décroissants.

(b) Seuil de doute croissants.

FIGURE A.7 – Data profile pour 29 problèmes pour un noyaux de covariance gaussien et différentes évolutions du seuil de doute avec une violation des contraintes de $\epsilon_c = 10^{-2}$.

A.5 Courbe de convergence du concept hybride pour les différentes mises à jour du seuil de doute.

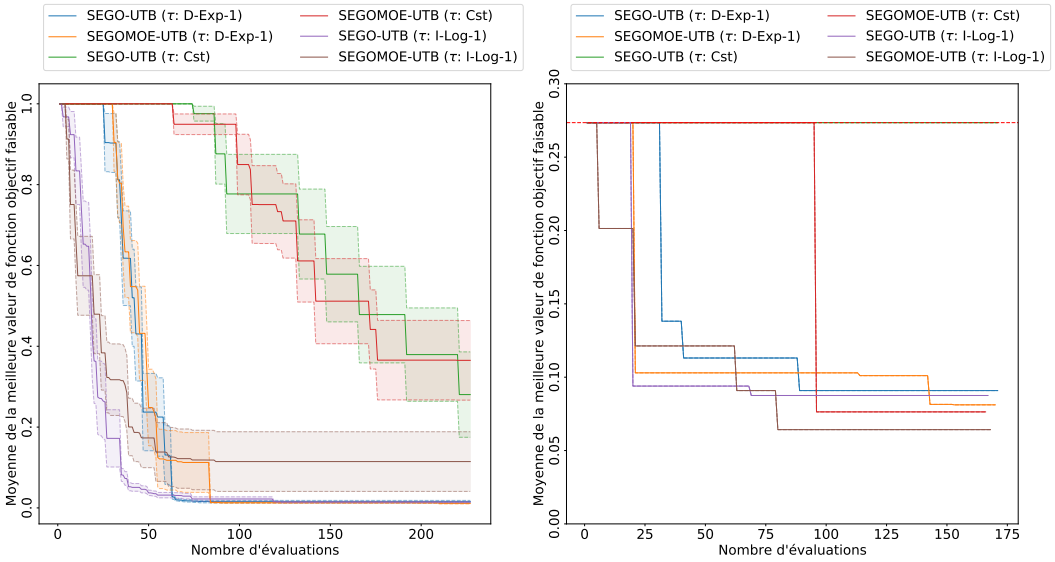


(a) seuil de doute décroissants.

(b) seuil de doute croissants.

FIGURE A.8 – Courbes de convergence du problème d'optimisation du concept hybride pour SEGO-UTB pour différentes mises à jour du seuil de doute. La ligne en pointillé grise verticale indique le nombre de points d'échantillonage dans le DoE initial.

A.6 Courbes de convergence en évaluations de la CMDO et de la PMDO de la BRAC pour les différentes mises à jour du seuil de doute pour SEGO-UTB et SEGOMOE-UTB.



(a) Courbes de convergence en évaluations de la CMDO de la BRAC.

(b) Courbes de convergence en temps de la PMDO de la BRAC.

FIGURE A.9 – Courbes de convergence en nombre d'évaluations pour la CMDO et la PMDO de la BRAC pour différents seuil de doute de SEGO-UTB et SEGOMOE-UTB. La ligne rouge horizontale indique la valeur de l'optimum de la CMDO de la BRAC dans le processus PMDO.

Annexe B

Liste des acronymes

AGILE Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts. 4

AI intelligence artificielle, ou artificial intelligence. 8, 25, 160

AL Lagrangien augmenté, ou augmented Lagrangian. 39, 40, 111

ALBO Lagrangien augmenté pour l'optimisation Bayésienne, ou augmented Lagrangian for Bayesian optimization. 39–41, 47–49, 87, 93, 111–114, 117–122, 132, 152, 157

AWSOM Automatic Wing Structural Optimization Module. 135

BALD Bayesian active learning by disagreement. 15

BBO optimisation boite noire, ou black box optimization. 2, 3

BFL longueur de piste équivalente, ou balanced field length. 135, 138

BO optimisation Bayésienne, ou Bayesian optimization. 3, 8, 19–23, 26, 27, 32–36, 39, 44, 47–49, 54, 56, 62, 76, 77, 83, 84, 93, 155, 156, 160

BOOST méthode d'optimisation reposant sur l'adjoint développé à Bombardier. 135, 136

BRAC configuration avion de recherche de Bombardier, ou Bombardier research aircraft configuration. 134, 136–139, 141–143, 146, 148, 151–153

CATIA v5 conception assistée tridimensionnelle interactive appliquée version 5. 135

CBO optimisation Bayésienne sous contraintes, ou constrained Bayesian optimization. 4, 5, 36, 39–41, 44, 46–49, 58, 59, 61, 62, 64, 86–90, 93, 111, 112, 117, 119, 120, 126, 139, 141, 152, 156–158

CMDO optimisation de conception multi-disciplinaire conceptuelle, ou conceptual multi-disciplinary design optimization. 133–139, 141, 143, 146–148, 151, 152

COBYLA constrained optimization by linear approximation. 12, 86, 111–114, 117–119, 121, 122, 126–129, 132, 133, 152, 153, 157, 158

CPU unité centrale de traitement, ou central processing unit. 68, 77, 80–84, 117, 120, 121, 126, 132–134, 136, 156

DFO optimisation sans dérivées ou derivative-free optimization. 2

DISC département d'ingénierie des systèmes complexes. 1

DMDO optimisation de conception multi-disciplinaire détaillée, ou detailed multi-disciplinary design optimization. 133

DoE plan d'expériences, ou design of experiments. 8, 9, 12, 15–27, 31, 33–35, 37–42, 47, 48, 58, 60, 62–68, 73, 74, 76, 77, 80–84, 90, 92, 93, 96, 97, 105, 110, 111, 127, 141, 146, 157–159

DP profil de données, ou data profile. 105, 107, 108, 117, 118, 122

DTIS département traitement de l'information et systèmes. 1, 4

ED-AA école doctorale aéronautique et astronautique. 1

EFI amélioration faisable espérée, ou expected feasible improvement. 36–41, 47–49, 87, 93, 111–114, 117–122, 132, 152, 157

EGO efficient global optimization. 20

EGO-KPLS efficient global optimization enhanced by a KPLS surrogate model. 27, 35, 48, 49, 54, 62, 76, 77, 80–84, 156, 157, 159

EGORSE efficient global optimisation coupled with random and supervised embeddings. 53, 61–68, 73, 74, 76, 77, 80–84, 156–159

EI amélioration espérée, ou expected improvement. 21–25, 31, 33, 36, 37, 40, 48, 56, 60, 64, 77, 159

EM maximisation des attentes, ou expectation-maximization. 17

EV violation espérée, ou expected violation. 43

Evol algorithme d'optimisation évolutionnaire, ou evolutionary optimization algorithm. 140–143, 146–148, 151–153, 158

FANSC full aircraft Navier-Stokes code. 135

FAST outil de conception pour configuration avion à voilure fixe, ou fixed-wing aircraft sizing tool. 124, 125, 132, 158

FAST-OAD outil de conception pour configuration avion du futur - conception d'avion complet, ou future aircraft sizing tool - overall aircraft design. 125

GMM modèle de mélange gaussien, ou Gaussian mixture model. 17, 18

GP processus gaussien, ou Gaussian process. I, V, VI, 3, 8, 9, 11–13, 15–20, 22–28, 30, 31, 33–36, 38–49, 54, 60, 62–64, 76, 84, 87–93, 111, 119, 120, 122, 147, 148, 152, 156, 157, 160

GSBP Goldstein-Price-Sinusoidal-Branin-Parr problem. 95, 99, 101, 103, 114, 117, 120, 121

HDBO optimisation Bayésienne en grande dimension, ou high dimensionnal Bayesian optimization. 4, 5, 20, 27, 35, 36, 45, 46, 48, 49, 53, 54, 62, 63, 73, 76, 81, 82, 156, 159

HDCBO optimisation Bayésienne sous-contraintes en grande dimension, ou high dimensionnal constrained Bayesian optimization. 8, 45–47, 49

HeSBO hashing-enhanced subspace Bayesian optimization. 27, 28, 30, 32, 35, 45, 48, 49, 54, 63, 65, 66, 76, 77, 80–84, 156, 157, 159

ICA altitude initiale de croisière, ou initial cruise altitude. 135, 138

ISAE-SUPAERO Institut Supérieur de l'Aéronautique et de l'Espace - SUPAERO. 1, 4, 64, 124, 134, 151, 152, 158, 160

ISRES improved stochastic ranking evolution strategy. 64, 92

KPLS Krigeage combiné aux moindres carrés partiels, ou kriging coupled with partial least squares. 14, 16, 17, 19, 27, 45, 47, 49, 54, 76, 139

L-BFGS-B limited memory Broyden-Fletcher-Goldfarb-Shanno method subject to bounds. 12

LAH Linear-Hartman-Ackley problem. 94, 101, 103, 114, 117

LHS échantillonnage par hypercube latin, latin hypercube sampling. 23, 96, 105, 141, 146

LSQ Linear-Sinusoidal-Quadratic problem. 95–97, 99, 103, 117

M2CI méthodes multidisciplinaires, concepts intégrés. 1, 4

MA mathématiques appliquées. 1

MADS mesh adaptive direct search algorithm. 111

MB problème de Branin modifié, ou modified Branin problem. 42, 65, 96, 97, 99, 103, 113, 114, 117, 120, 121, 156

MBGRID générateur de maillage reposant sur CATIA développé à Bombardier. 135

MDO optimisation de conception multi-disciplinaire, ou multi-disciplinary design optimization. 2, 8, 125, 126, 132–136, 151, 152, 160

MGP processus gaussien marginal, ou marginal Gaussian process. V, 13–17, 19, 47, 49, 57, 60, 61, 65, 67, 68, 73, 74, 83, 84, 156

ML apprentissage automatique, ou machine learning. 8, 22, 25, 26, 160

MOE mélange d'experts, ou mixture of experts. 17–20, 45, 47

MOPSO optimisation multi-objectifs par essaims particulaires, ou multi-objective particle swarm optimization. 140

MOST outil de système d'optimisation multi-fonctions, ou multi-function optimization system tool. 140

MTOW masse maximale au décollage, ou maximum take-off weight. 135, 138, 146, 148, 151, 152

NLPQL programmation non-linéaire par Lagrangien quadratique, ou non-linear programming by quadratic Lagrangian. 140

NOMAD nonlinear optimization with the MADS algorithm. 86, 111–114, 117–119, 121, 122, 126–129, 132, 152, 153, 157, 158

NOMAD-VNS NOMAD-VNS. 111, 113, 114, 117, 118, 126

OML ligne de moule externe, ou outer mold lines. 135

ONERA Office National d'Etudes et de Recherches Aérospatiales. 1, 4, 64, 124, 133, 134, 151, 152, 158, 160

PEB barrière progressive et extrême, ou progressive and extreme barrier. 111

PES recherche entropique prédictive, ou predictive entropy search. 25, 38, 48, 118

PESC recherche entropique prédictive avec contraintes, ou predictive entropy search with constraints. 39, 41, 47–49, 87, 93, 111, 112, 114, 117, 119–122, 132, 152, 157

PLS moindres carrés partiels, ou partial least squares. 16, 17, 19, 57, 60, 61, 65, 67, 68, 73, 82–84, 156, 159

PMDO optimisation de conception multi-disciplinaire préliminaire, ou preliminary multi-disciplinary design optimization. 133–136, 138, 139, 141, 142, 146, 148, 151, 152

Pointer-2 variante de l'algorithme du "chien de détection", ou variant of the Pointerdog algorithm. 140–143, 146–148, 151–153, 158

REMBO random embedding Bayesian optimization. 27–30, 32, 34, 35, 45, 46, 48, 49, 54, 61, 63, 80, 156, 159

RREMBO R random embedding Bayesian optimization. 27–30, 32, 34, 35, 45, 48, 49, 53–57, 59–63, 65, 66, 76, 77, 80–84, 156, 157, 159

S^3-BFO sequential-subspace-search bayesian functional optimisation. 84, 157, 159

SCBO scalable constrained Bayesian optimization. 45, 46, 49

SEGO super efficient global optimization. 42–49, 59, 60, 64, 87, 93, 111, 113, 114, 117–122, 126–129, 132–134, 139, 141–143, 146–148, 151–153, 156–158

SEGO-EV super efficient global optimization using the expected violation. 43, 44, 48, 49, 87, 157

SEGO-KPLS super efficient global optimization enhanced by a KPLS surrogate model. 45, 46, 49

SEGO-UTB super efficient global optimization using the upper trust bound. 44, 45, 47–49, 86, 87, 89–91, 94, 96, 97, 99, 101, 103, 105, 108–114, 117–121, 123–129, 132–134, 139, 141–143, 147, 148, 151–153, 157–159

SEGOMOE super efficient global optimization coupled with mixture of experts. 42–48, 64, 76, 111, 123, 124, 134, 139, 141–143, 146–148, 151–153, 157–160

SEGOMOE-UTB SEGOMOE utilisant UTB. 134, 139, 141–143, 147, 148, 151–153, 158

SMT surrogate modeling toolbox. 12, 64, 65, 91

SNOPT sparse nonlinear optimizer. 64, 92

SoC état de charge, ou state of charge. 126, 127

SUR réduction progressive de l'incertitude, ou stepwise uncertainty reduction. 24, 25, 38, 39, 41, 47–49, 87, 111–114, 117, 119–122, 132, 152, 157

TEC quantité totale d'énergie consommée, ou total energy consumption. 125, 127, 152

TOFL longueur de piste au décollage, ou takeoff field length. 126, 127

TRIKE trust region implementation in kriging-based optimization with expected improvement. 33

TS échantillonage de Thomson, ou Thomson sampling. 23, 34, 48

TuRBO trust region Bayesian optimization. 27, 33–35, 45, 46, 48, 49, 54, 62, 76, 77, 80–84, 156, 157

UTB borne supérieure de confiance, ou upper trust bound. 43, 44, 87–90, 143, 148, 157, 158

VNS recherche en voisinage, ou variable neighborhood search. 111, 112

WB2 Watson Barnes 2. 22–25, 48

WB2S Watson Barnes 2 normalisée, ou scaled Watson Barnes. 23–25, 48, 92, 139