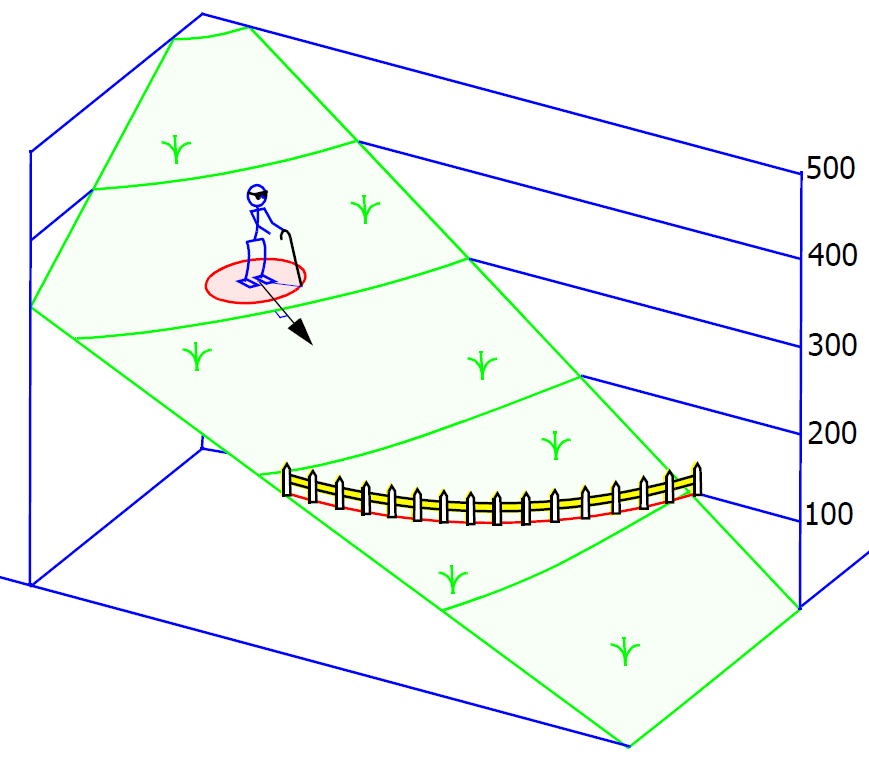
****

|  |
| --- |
| Mémoire de PFE |
| Optimisation paramétrique en simulation numérique |
|  |
| **Quentin BOUNIOL** |
| **28/11/2016** |

|  |
| --- |
| Tuteur industriel : M. RUIZ Guillaume  Tuteur académique : M. NAIT ALI Azdine |
| *Du 02/05/2016 au 28/10/2016* |



# Résumé/Abstract

Ce rapport traite de la compréhension et de la mise en place d’une méthode d’optimisation paramétrique générale en simulation numérique. Après avoir réalisé un état de l’art pour comprendre le fonctionnement des algorithmes d’optimisation, un cas test est réalisé sur un modèle thermomécanique avec le logiciel Optimus combiné avec Marc/Mentat afin de tester la mise en place de boucle d’optimisation et de déterminer les performances de chaque algorithme pour en déduire une classification. Un deuxième cas industriel permet cette fois d’allier optimisation paramétrique et topologique pour respecter les performances fixées par le cahier des charges client d’un bouchon doseur de lessive. L’étude traite de l’écoulement de la lessive dans le bouchon par effet siphon grâce à la différence de pression avec, dans un premier temps, la plate-forme ANSYS Workbench utilisant le logiciel Fluent puis, dans un second temps, avec un code Python couplé avec Fluent aux meilleures performances. Ce cas permet d’étudier la validité des méthodes de résolution numérique proposées au cas précédent et de pouvoir ainsi les appliquer à un cas industriel.

This report is about the understanding and the setting up of general parametric optimization methods in numerical simulation. After studying the principles of several optimization algorithms, a first case is realized on a thermomechanical model on Optimus combined with Marc/Mentat in order to test the setting up of an optimization loop and to establish the performance of each algorithm to build then a classification. A second industrial case allows the engineer to ally parametric and topologic optimization to respect the performance fixed by client’s specifications. The study is about a detergent flow in a plug modeled with, in a first time, ANSYS Workbench/Fluent and, in a second time, with Python/Workbench/Fluent, which gives better results. This case permits to study the validity of the numerical resolution methods proposed in the first case.

# Remerciements

Tout d’abord, je tiens à remercier Guillaume Ruiz, mon tuteur de stage pendant ces 6 mois. Investi et impliqué, il a su m’expliquer les principes de base de l’optimisation et me donner des pistes pour poursuivre l’étude. Il a toujours été à l’écoute et a fait preuve de patience et sérieux.

Je remercie Matthieu Puyo, responsable du service de simulation numérique, pour l’opportunité qu’il m’a offerte en me permettant d’effectuer mon stage chez Ingeliance. J’ai acquis aussi bien des compétences techniques que professionnelles et un épanouissement personnel.

Je souhaiterais également remercier Delphine Depeyras, responsable de l’équipe CFD, pour son écoute et la disponibilité dont elle a fait preuve pour me donner des conseils fructueux au cours de réunions et relire mon mémoire.

Je remercie tout particulièrement François Plessier, Christophe Ars et Gilles Vogt pour l’aide qu’ils m’ont apportée au cours des cas industriels que j’ai pu rencontrer. Leurs expériences dans les domaines du calcul thermomécanique, de la simulation d’écoulements fluide et de la programmation m’ont apporté une aide précieuse pour remplir les objectifs du stage.

Enfin, je remercie l’ensemble des membres de l’agence de Bordeaux pour leurs conseils sur certains logiciels, leur convivialité et leur bonne humeur.

# Présentation de l’entreprise

Le groupe INGELIANCE est une société de prestation en ingénierie depuis 2001 suite à la fusion des entreprises ALTEP Ingénierie et AXS Analyse de Structures. Il compte plus de 500 collaborateurs répartis à travers toute la France et à l’international avec des agences à Singapour et en Allemagne afin de répondre au mieux aux besoins de ses clients. La société, qui représente un chiffre d’affaires de 30 M d’euros, couvre un large panel de secteurs industriels clés de l’ingénierie comme présenté sur le diagramme suivant.

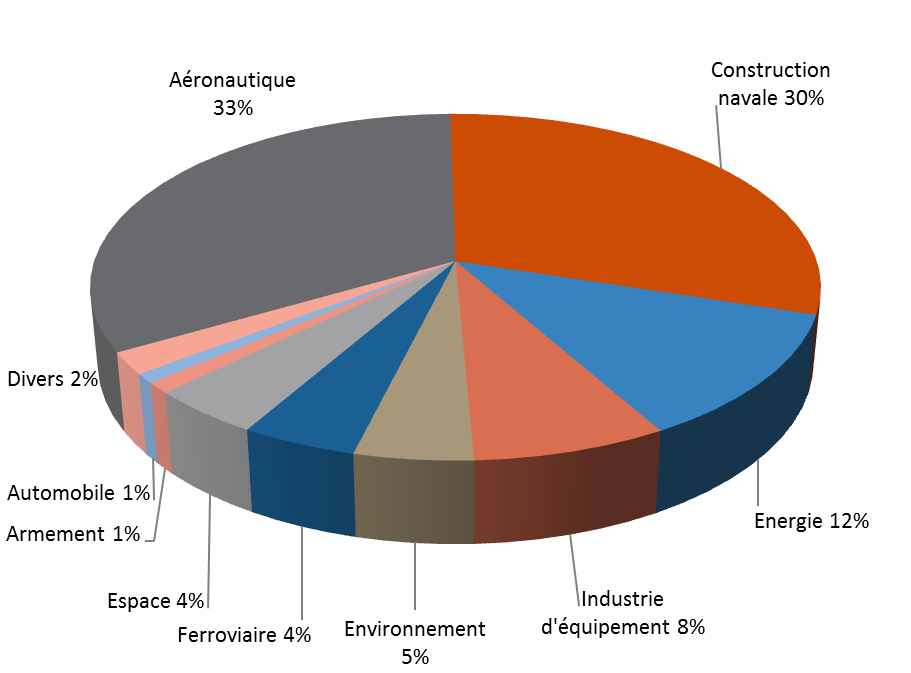


Figure : Secteurs d'activité du groupe Ingéliance

Le stage s’est réalisé à l’agence de Mérignac (33) dont le bureau de calcul regroupe une trentaine d’ingénieurs spécialisés dans le domaine de la simulation numérique et répartis en deux équipes dirigées par M. Matthieu PUYO : le pôle Structure/Architecture et le pôle CFD (Computational Fluid Dynamics)/Couplage multiphysiques. Le stage s’est déroulé dans le pôle Structure/Architecture sous la direction et encadré par Mr Guillaume RUIZ.

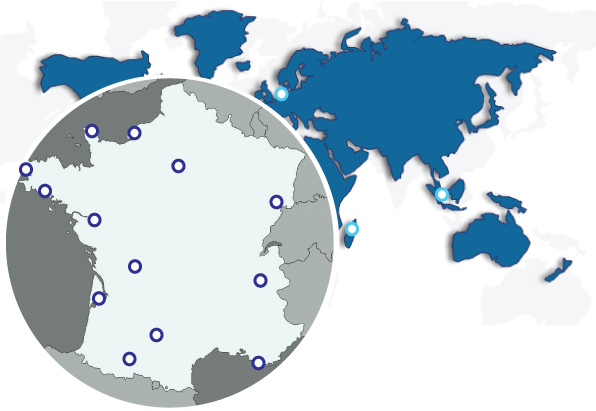


Figure : Implantation du groupe en France et à l'international

# Contexte du stage au sein d’Ingeliance

L’agence de Bordeaux a pour vocation au sein du groupe d’être d’avantage multidisciplinaire et polyvalente (sur les différents métiers, mais aussi sur divers secteurs d’activité et donc de produits industriels). De ce fait, elle souhaite étendre son expertise afin de capter de nouveaux marchés, tel que celui de l’optimisation dans un cadre industriel. L’industrie aéronautique est constamment en recherche de nouvelles conceptions de produit et est amenée à rencontrer des problèmes d’ordre technologique : réduction de coût et de masse, recalage d’un modèle par rapport à des données expérimentales… Ces problématiques peuvent s’exprimer sous la forme d’une fonction objectif à minimiser. Par exemple, dans le célèbre problème du voyageur de commerce, la longueur de la tournée d’un “voyageur de commerce”, qui doit visiter un certain nombre de villes, avant de retourner à la ville de départ, doit être minimisée. La définition du problème d’optimisation est souvent complétée par la donnée de *contraintes* (conditions initiales, paramètres à maximiser, minimiser…)sur les paramètres de la solution proposée, afin que celle-ci soit réalisable. Afin de minimiser cette fonction objectif, de nombreux algorithmes d’optimisation sont disponibles au sein de logiciels tels que Optimus, ANSYS ou directement sous forme de librairies dans pour des langages de programmation tel que Python.

Ingeliance est une entreprise de conseil en ingénierie et technologie dépendante du climat industriel. Il est donc important pour une société de ce type de se diversifier, d’être toujours innovants et en capacité de faire de réelles propositions aux clients, parfois même avant qu’ils en aient exprimé le besoin. C’est la raison pour laquelle Ingeliance souhaite élargir ses compétences en modélisation numérique, en proposant des solutions d’optimisation applicables à des cas industriels dans des délais de mise en œuvre raisonnables. C’est dans ce contexte que se positionne le stage afin de développer un nouvel axe de compétence au sein de la société répondant aux thématiques actuelles de l’industrie.

Il existe de nombreuses méthodes d’optimisation “classiques” pour résoudre de tels problèmes, applicables lorsque certaines conditions mathématiques sont satisfaites. Ainsi, la *programmation linéaire* traite efficacement le cas où la fonction objectif, ainsi que les contraintes, s’expriment linéairement en fonction des variables de décision. Malheureusement, les cas industriels comportent souvent une ou plusieurs complexités introduisant des non linéarités dans la fonction objectif, et donc le besoin de recourir à d’autres approches.

Ce stage est à dominante « recherche appliquée » et a deux objectifs distincts. Il s’agit tout d’abord d’étudier de manière approfondie le fonctionnement des différents algorithmes présents dans la littérature. Dans un second temps, il s’agit de vérifier de manière qualitative la performance de ces algorithmes sur des cas industriels concrets afin d’en déterminer une méthode aboutie pour l’optimisation paramétrique générale.

# Table des matières

[Résumé/Abstract 2](#_Toc465348407)

[Remerciements 3](#_Toc465348408)

[Présentation de l’entreprise 4](#_Toc465348409)

[Contexte du stage au sein d’Ingeliance 5](#_Toc465348410)

[Table des matières 6](#_Toc465348411)

[Table des illustrations 8](#_Toc465348412)

[Table des annotations 9](#_Toc465348413)

[Etat de l’art 10](#_Toc465348414)

[I. Notion de préférence et problème multi-objectifs 11](#_Toc465348415)

[II. Méthodes de sélection multi-objectifs 12](#_Toc465348416)

[A. Méthodes de résolution 12](#_Toc465348417)

[B. Méthodes de classement multi-objectif 12](#_Toc465348418)

[III. Les méthodes d’optimisation 14](#_Toc465348419)

[A. Les méthodes déterministes 15](#_Toc465348420)

[B. Les méthodes stochastiques 23](#_Toc465348421)

[IV. Les problèmes à grand nombre de variables en conception 28](#_Toc465348422)

[A. La notion de parcimonie 28](#_Toc465348423)

[B. Réduction et approximation de modèles 28](#_Toc465348424)

[C. La gestion par lots 29](#_Toc465348425)

[Première optimisation sur un modèle thermique 30](#_Toc465348426)

[I. Contexte 30](#_Toc465348427)

[II. Déroulement 31](#_Toc465348428)

[III. Traitement des résultats 32](#_Toc465348429)

[IV. Présentation des résultats 34](#_Toc465348430)

[Optimisation paramétrique/topologique d’un bouchon de lessive 40](#_Toc465348431)

[I. Contexte 40](#_Toc465348432)

[II. Traitement d’un cas test : conduite coudée 40](#_Toc465348433)

[III. Application au bouchon de lessive 43](#_Toc465348434)

[A. Simplification de modèle 43](#_Toc465348435)

[B. Modèles monophasiques 44](#_Toc465348436)

[C. Modèles diphasiques 47](#_Toc465348437)

[Conclusion 50](#_Toc465348438)

[Références 51](#_Toc465348439)

[Annexes 53](#_Toc465348440)

[I. Formalisation des préférences sur les variables d’observation 53](#_Toc465348441)

[A. Le paradoxe de St Pétersbourg 53](#_Toc465348442)

[B. Les fonctions de satisfaction 53](#_Toc465348443)

[C. Les fonctions d’appartenance 53](#_Toc465348444)

[D. Les fonctions d’utilités 53](#_Toc465348445)

[E. Les fonctions de désirabilité 54](#_Toc465348446)

[II. Le système d’aide à la décision 56](#_Toc465348447)

[A. Identification des objectifs de conception 56](#_Toc465348448)

[B. Etablissement des priorités 56](#_Toc465348449)

[C. La notion d’arc-élasticité 56](#_Toc465348450)

[III. Code Python pour l’optimisation 59](#_Toc465348451)

# Table des illustrations

[Figure 1 : Secteurs d'activité du groupe Ingéliance 4](#_Toc465344404)

[Figure 2 : Implantation du groupe en France et à l'international 4](#_Toc465344405)

[Figure 3 : Front de Pareto 11](#_Toc465344406)

[Figure 4 : Echelle de comparaison et matrice de jugement 13](#_Toc465344407)

[Figure 5 : Indices aléatoires proposés par Saaty 14](#_Toc465344408)

[Figure 6 : Classification des méthodes d'optimisation 23](#_Toc465344409)

[Figure 7 : Génération d'un individu enfant 24](#_Toc465344410)

[Figure 8 : Algorithme de l'essaim particulaire 26](#_Toc465344411)

[Figure 9 : Recherche du meilleur chemin pour les fourmis 27](#_Toc465344412)

[Figure 10 : Configuration de l'essai thermomécanique de la rotule EP4 30](#_Toc465344413)

[Figure 11 : Disposition des capteurs de température 30](#_Toc465344414)

[Figure 12 : Première méthode d'optimisation proposée 33](#_Toc465344415)

[Figure 13 : Seconde méthode d'optimisation proposée 34](#_Toc465344416)

[Figure 14 : Scores des différents algorithmes pour l'Erreur Quadratique Moyenne 35](#_Toc465344417)

[Figure 15 : Scores des différents algorithmes pour le Coefficient de Détermination 35](#_Toc465344418)

[Figure 16 : Scores des différents algorithmes pour la fonction objectif 35](#_Toc465344419)

[Figure 17 : Scores des différents algorithmes pour le temps de calcul 36](#_Toc465344420)

[Figure 18 : Scores totaux des différents algorithmes 36](#_Toc465344421)

[Figure 19 : Nouveaux scores des différents algorithmes pour l'Erreur Quadratique Moyenne 37](#_Toc465344422)

[Figure 20 : Nouveaux scores des différents algorithmes pour le Coefficient de Détermination 37](#_Toc465344423)

[Figure 21 : Nouveaux scores des différents algorithmes pour la fonction objectif 38](#_Toc465344424)

[Figure 22 : Nouveaux scores des différents algorithmes pour le temps de calcul 38](#_Toc465344425)

[Figure 23 : Nouveaux scores totaux pour les algorithmes utilisés 38](#_Toc465344426)

[Figure 24 : Méthode d'optimisation finale 40](#_Toc465344427)

[Figure 25 : Géométrie du bouchon de lessive 41](#_Toc465344428)

[Figure 26 : Géométrie de la conduite coudée réalisée sous ANSYS 42](#_Toc465344429)

[Figure 27 : Surface de réponse générée lors de l'optimisation de la conduite coudée 42](#_Toc465344430)

[Figure 28 : Evolution de la géométrie du coude avant et après optimisation 43](#_Toc465344431)

[Figure 29 : Visualisation sur la géométrie du bouchon des paramètres d'entrée de l'optimisation 44](#_Toc465344432)

[Figure 30 : Surface de réponse générée après une première optimisation sur le bouchon de lessive 45](#_Toc465344433)

[Figure 31: Méthode d'échantillonnage pour rechercher une valeur cible 46](#_Toc465344434)

[Figure 32 : Convergence de la solution vers la valeur cible 46](#_Toc465344435)

[Figure 33 : Lignes de courant dans le bouchon de lessive 47](#_Toc465344436)

[Figure 34 : Lignes de courant dans le bouchon de lessive avec une condition sur la prise d'air 47](#_Toc465344437)

[Figure 35 : Configuration avant-après optimisation du bouchon 49](#_Toc465344438)

[Figure 36 : Valeurs les plus basses de la fonction objectif en fonction du SG et de la longueur du tube 50](#_Toc465344439)

[Figure 37 : Fonctions de désirabilité de Harrington 55](#_Toc465344440)

[Figure 38 : Fonctions de désirabilité de Derringer 56](#_Toc465344441)

# Table des annotations

|  |  |
| --- | --- |
| **Symbole** | **Signification** |
| Vobs | Variable d'observation |
| f | Fonction objectif |
| d | Direction de descente |
| VSA | Variable de sortie agrégée |
| Nabla de f | Gradient de f |
| Nabla carré de f | Hessien de f |
| α | Pas de direction de descente |
| P | Projection d'une fonction |
| Par | Parcimonie d'un problème |
| Vco | Variable de conception |
| T | Statistique de Student |
| SG | Siphon gap [m] |
| VP | Venting pipe [m] |

# Etat de l’art

La conception d’un produit dans l’industrie dépend de ses caractéristiques architecturales telles que la masse, les dimensions, le choix des composants … Dans la plupart des cas, de nombreuses solutions potentielles s’offrent au concepteur qui devra faire un choix et pouvoir le justifier. Ce choix doit intégrer la présence d’une solution de référence [Collignan 2011] issue de projets ou d’études préliminaires.

La conception d’un tel produit dépend non seulement de la connaissance physique du problème mais aussi de connaissances subjectives liées aux préférences, au savoir-faire, aux habitudes, aux règles métiers et aux expériences des concepteurs en charge du projet. Le but sera de mettre en forme le problème avec le moins de variables possibles, de formaliser les paramètres subjectifs du problème tout en prenant compte qu’une solution initiale est peut-être déjà en place : la nouvelle solution devra être proche de la solution initiale afin de ne pas engager trop de frais par exemple. Cependant, l’optimisation peut également être utilisée afin de trouver des solutions complètement différentes innovantes par rapport aux habitudes actuelles. Dès lors, deux types de raisonnement s’offrent au concepteur [Saridakis 2008] :

* Le raisonnement à partir de modèles (MBR) : présente une description précise du problème et des solutions potentielles pour les simuler et identifier la plus pertinente. La modélisation constituant indéniablement la phase la plus longue et fastidieuse, il est important de souligner les différents types de modélisation en conception :
  + Les « boîtes blanches » dont le contenu est connu du concepteur :
    - Les modèles numériques (CAO, éléments finis …)
    - Les modèles analytiques basés sur une représentation des modèles physiques et sur une traduction mathématique
  + Les « boîtes noires » permettant de modéliser les interactions d’un système sans en comprendre le fonctionnement et sans pouvoir accéder aux éléments qui la définissent.
* Le raisonnement à partir de cas (CBR) : vise à analyser le problème de conception et le résoudre au moyen d’une solution utilisée dans un problème passé. Un tel raisonnement n’est efficace que si le nombre de cas rencontrés dans la base de connaissance est suffisant [Amen, 2001].

Compte tenu des activités de l’entreprise, un raisonnement à base de modèles sera privilégié afin de formaliser [Collignan, 2011]:

* Le comportement physique du produit
* Le processus de décision visant à évaluer la performance globale
* La méthode de recherche d’une solution satisfaisante
* La tendance à rester proche d’une éventuelle solution initiale

La recherche d’une solution satisfaisante introduit la notion d’optimisation qui est le sujet du stage et qui vise à déterminer le meilleur ensemble de variables qui répond au cahier des charges. Mais qu’est-ce qui pousse à choisir une solution plutôt qu’une autre ? Au cours du XXème siècle, plusieurs économistes ont posé les bases de l’aide à la décision. Par exemple, la théorie de l’utilité [Neumann, 1944] stipule que chaque alternative doit être associée à une variable d’utilité : il s’agit ici de formaliser la notion de préférence entre les différentes solutions. Sélectionner la solution préférée revient à maximiser la variable d’utilité.

## Notion de préférence et problème multi-objectifs

Soit une solution d’un ensemble de solutions d’un problème. Une solution de conception correspond à une configuration de produit et prend la forme d’un jeu de variables. Les solutions ne sont que très rarement comparables directement entre elles. La plupart du temps, elles sont caractérisées par des variables d’observation VObs aussi appelées attributs ou critères [Collignan, 2011]. Les VObs sont utilisées par le concepteur pour observer le comportement d’un produit (masse, volume, coût …). Ainsi, des VObs de même nature sont comparables et ordonnables. Dans la majorité des cas, chaque solution présente plusieurs variables d’observations : le problème est qualifié de multi-objectifs. Lorsque plusieurs variables d’observations doivent être prises en compte, plusieurs problèmes sont rencontrés [Jones, 2002]:

* Chaque variable d’observation vise à satisfaire un but ou répondre à une contrainte donnée. L’objectif est d’avoir une valeur satisfaisante, c’est-à-dire proche du but escompté
* Les variables d’observations sont souvent antinomiques : si l’une est satisfaisante, les autres s’éloignent de leurs buts respectifs

Il faut cependant bien dissocier les buts des contraintes dans un problème multi-objectifs [Billionnet, 2006] :

* But : s’approcher d’une valeur cible pour une VObs
* Contrainte : restrictions (égalité(s) ou inégalité(s)) sur la variable. Une solution qui viole une contrainte est inacceptable.

Il est également possible de représenter des solutions dans l’espace des variables d’observation via un graphique dont les axes correspondent à chaque VObs. Sur ce graphe, les solutions sont assimilées à des points dont les VObs sont les coordonnées. Une solution est préférée à une autre si toutes ses variables d’observation ont une valeur également satisfaisante ou plus satisfaisante et si l’une des VObs est strictement plus satisfaisante : c’est la dominance. Si une solution s’avère n’avoir qu’une partie de ses VObs qui soit également satisfaisante ou plus satisfaisante que celle d’une autre solution, ces deux solutions sont équivalentes.

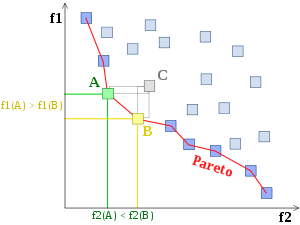


Figure : Front de Pareto

Il existe un certain nombre de solutions qui ne sont pas dominées. Cet ensemble est nommé Front de Pareto. Cependant, identifier l’ensemble des solutions non dominées ne permet pas de sélectionner une solution particulière. Une fois le front obtenu, le concepteur doit encore faire un choix non formalisé parmi les solutions.

## Méthodes de sélection multi-objectifs

### Méthodes de résolution

Il s’agit ici de passer en revue plusieurs méthodes permettant de sélectionner une solution dans un ensemble de solutions.

#### Equilibre de Nash

L’équilibre de Nash fut proposé dans le cadre de la théorie des jeux [Nash, 1951]. Le théorème de Nash stipule qu’il existe un point d’équilibre où chaque joueur ne pourra plus améliorer sa position sans dégrader celle des autres : c’est l’équilibre de Nash. Dans un problème multi-objectifs, une VObs est vue comme un joueur et « améliorer sa position » consiste à approcher son but. Chaque VObs est traitée une à une afin d’atteindre son but sans éloigner les autres VObs de leur but respectif. L’équilibre de Nash indique qu’une solution peut être atteinte : c’est une position d’équilibre. Le principal inconvénient de l’équilibre de Nash est que la solution d’équilibre n’est pas la meilleure pour chaque VObs (il n’y a pas unicité de la solution d’équilibre). Toutefois, cette théorie est intéressante car elle suggère une résolution par coopération et non par confrontation.

#### Méthode des ε-contraintes

La méthode des ε-contraintes proposées par [Haimes, 1971] est le contraire de la méthode précédente. Seul un but est satisfait, les autres buts sont alors transformés en contraintes (transformation en un problème de minimisation). Chaque nouvelle contrainte obtenue se doit d’être inférieure à un. La difficulté de cette méthode résulte dans le choix des . Il faut qu’ils soient les plus petits possibles afin de refléter au mieux les problèmes initiaux tout en étant suffisamment grands pour garantir l’existence de solutions au problème et ne pas le sur-contraindre. Enfin, le principal inconvénient de cette méthode est que la conversion de buts en contraintes implique une perte d’informations qualitatives : les préférences du concepteur ne sont plus exprimées sous forme de buts.

### Méthodes de classement multi-objectif

Les méthodes présentées ci-dessous visent à établir un classement qui va tenir compte des préférences des concepteurs en vue de la sélection d’une solution.

#### Système de pointage simple

Le système de pointage simple [LeBel, 2009] réalise une somme pondérée des notes affectées à chaque attribut. Cette méthode est divisée en trois étapes :

1. Etablissement des poids pour chaque attribut
2. Etablissement du niveau de satisfaction de la valeur des attributs de chaque solution
3. Calcul du pointage de chaque solution

Les poids affectés sont choisis sur une échelle de valeur, allant de 1 à 5. De même, le niveau de satisfaction de chaque attribut pour chaque solution est évalué de 1 (très bas) à 10 (très élevé). Le pointage est réalisé en effectuant la somme de chaque niveau de satisfaction multiplié par le poids de l’attribut correspondant.

#### Processus de hiérarchie analytique (AHP)

Le processus de hiérarchie analytique est une méthode de hiérarchie des préférences pour effectuer un classement cardinal entre solutions. L’AHP fut proposé par Saaty à la fin du XXème siècle et est adapté à la décision dans une étude multi-objectifs [Saaty, 1990]. Pour chaque niveau hiérarchique constitué d’entités, il faut réaliser une matrice de jugement M en comparant chaque paire d’entités.

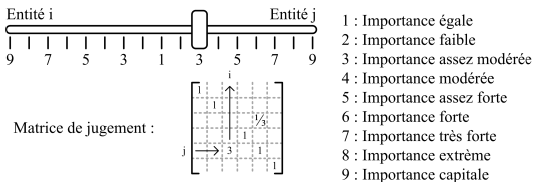


Figure : Echelle de comparaison et matrice de jugement

Pour chaque paire d’entités, une préférence est exprimée en termes d’importance de l’une par rapport à l’autre. Lorsqu’une entité j est préférée à une entité i, la valeur correspondante dans l’échelle de Saaty est reportée dans la matrice de jugement à la ligne de l’entité préférée et à la colonne de l’entité non préférée. La matrice possède une diagonale de 1 puisque chaque entité est également préférée à elle-même. Enfin, chaque élément est inversé par rapport à la diagonale afin d’obtenir une matrice inverse symétrique.

Une fois la matrice de jugement établie, il est possible de calculer les priorités de chaque entité par une méthode de normalisation de la matrice [LeBel, 2009].

Chaque colonne est adimensionnée par la somme de ses éléments. Sur la matrice normalisée résultante, la moyenne des éléments est calculée pour chaque ligne. Le vecteur obtenu est enfin lui-même adimensionné par rapport à la somme de ses éléments. Le vecteur final est constitué des priorités de chaque entité.

**for** colonne

M\_norm(colonne) ← M(colonne)/somme(M(colonne))

**end** **for**

**for** ligne

V ← moyenne(M\_norm(ligne))

**end** **for**

V\_priorites ← V/somme(V)

Cependant, la matrice de jugement est soumise aux préférences du concepteur et celui-ci peut proposer des préférences incohérentes ou mal utiliser l’échelle de comparaison. Le calcul du ratio de cohérence permet de s’assurer que la matrice de jugement respecte un niveau de cohérence minimum :



Figure : Indices aléatoires proposés par Saaty

RC correspond en fait à la probabilité que l’utilisateur ait rempli la matrice de jugement de manière totalement aléatoire. Le seuil proposé par Saaty est de 10% au-dessous duquel doit se situer le ratio pour considérer que la matrice de jugement est cohérente. Cependant, Saaty souligne que cette solution ne convient qu’à un problème de 7 entités ou moins. Au-delà, la matrice de jugement M est de moins en moins cohérente. Il apparait nécessaire d’utiliser d’autres méthodes pour comparer plus de sept solutions simultanément.

## Les méthodes d’optimisation

Une fois le problème posé, il faut maintenant évoquer la résolution numérique afin d’établir en pratique une solution préférée. Deux notions sont à distinguer :

* Le choix qui correspond à la sélection d’une solution parmi un nombre fini de candidats
* L’optimisation qui génère des solutions candidates et en sélectionne une, préférée par rapport aux autres. L’optimisation est une activité de choix en conception portant sur un très grand nombre de solutions.

En application du raisonnement à partir de modèles (MBR), les variables d’observations se trouvent en sortie du modèle dont les variables d’entrées sont les variables de conception (dimensions, type de composant, flux, énergie …). D’après les paragraphes précédents, il était possible de modéliser une solution candidate par des variables d’observation ou des variables d’interprétation auxquelles s’ajoute maintenant l’espace des variables de conception. Réaliser une optimisation de produit en conception revient donc à identifier un vecteur de variables de conception permettant d’obtenir des variables d’observation satisfaisantes au problème d’aide à la décision. Chaque variable de conception est associée à un domaine de valeurs qui représente l’étendue et l’incertitude. L’ensemble des domaines de valeur constitue l’espace de conception dans lequel chaque point est une solution candidate. La méthode d’optimisation procède par opérations mathématiques et/ou logiques pour identifier l’optimum d’une fonction-objectif. Lorsque cet optimum est atteint, c’est la convergence. Les qualités d’une méthode d’optimisation sont les suivantes :

* Capacité à traiter des fonctions objectifs non linéaires, non dérivables et multi-modales
* Facilité d’utilisation
* Bon taux de convergence

En optimisation, la notion d’échelle est importante. Deux types d’optimisation sont alors distingués :

* L’optimisation locale qui identifie un optimum proche d’un point connu de la fonction objectif
* L’optimisation globale qui identifie l’optimum à l’échelle de l’ensemble de l’espace de conception

Les principales méthodes d’optimisation peuvent être classées en trois catégories [Cabodevilla, ]:

1. Les méthodes déterministes qui évaluent le point de calcul suivant en fonction des caractéristiques de la fonction au point considéré
2. Les méthodes stochastiques qui intègrent une part de hasard dans le choix du point de calcul suivant
3. L’énumération qui consiste à discrétiser l’espace engendré par le vecteur à optimiser et à calculer l’ensemble des possibilités

Les méthodes d’optimisation peuvent être évaluées et comparées selon certains critères :

* La robustesse qui est l’aptitude de la méthode à converger vers un minimum global sans être perturbée par des minima locaux
* La vitesse, mesurée en nombre d’itérations sur la fonction objectif
* La globalité du minimum

### Les méthodes déterministes

Les méthodes déterministes usuelles nécessitent la définition d’une fonction objectif : le modèle du produit incluant éventuellement des préférences formalisées. Ce modèle admet des variables de conception en entrée et une variable en sortie qui correspond à l’agrégation des variables d’interprétation entre elles, notée VSA.

Ces méthodes usent d’outils mathématiques, en particulier la continuité et la dérivabilité au premier et/ou second ordre pour trouver l’optimum de la fonction objectif. Ainsi est générée la solution de conception préférée. Si ces méthodes sont relativement rapides en temps de calcul, elles sont limitées à certaines fonctions objectifs, en particulier des fonctions linéaires, dérivables et convexes. Elles nécessitent généralement une solution de référence pour débuter l’optimisation et il arrive que la solution optimale générée soit proche de la solution de référence sans pour autant être la meilleure solution possible : c’est l’optimisation locale puisque l’algorithme d’optimisation n’a convergé que dans une zone locale de la fonction objectif.

Les méthodes déterministes les plus utilisées seront décrites ci-dessous (liste non exhaustive)

#### Sans contraintes

Soit le problème d’optimisation sans contraintes suivant :

Avec f une fonction continue, le gradient de f, le Hessien de f, d une direction de descente. Pour tout α réel positif suffisamment petit :

Les méthodes à directions de descentes suivent ce principe et construisent une suite d’itérés approchant la solution du problème de la façon suivante :

Un algorithme à directions de descente est donc déterminé par les paramètres d et α : la façon dont la direction de descente d est calculée donne son nom à l’algorithme et la façon dont le pas α est déterminé est appelée recherche linéaire et peut se déterminer de différentes façons.

##### Gradient et quasi-Newton

L’ensemble des méthodes présentées ci-dessous convergent si la fonction est convexe. Le principal inconvénient de ces méthodes est de devoir calculer le gradient de la fonction à chaque itération.

###### Recherche linéaire (Bergman, 2010)

La recherche linéaire consiste à déterminer le pas αk à effectuer le long d’une direction de descente dk. Des valeurs de ce pas peuvent être obtenues par différentes méthodes : par recherche linéaire exacte, par la méthode d’Armijo ou de Goldstein et par la méthode de Wolfe.

Le pas αk est souvent choisi afin de vérifier les deux conditions suivantes :

* la fonction f doit décroître suffisamment le long de la direction de descente
* le pas αk ne doit pas être trop petit

Soit donc

Recherche linéaire exacte

La recherche linéaire exacte consiste à déterminer le pas optimal, c’est à dire le pas qui minimise la fonction f le long de la direction de descente dk. Le pas αk est donc solution du problème :

Le pas optimal ainsi déterminé est le pas de Cauchy. Le problème ci-dessus est appelé problème de Cauchy. Le pas optimal obtenu correspond donc au minimum global de la fonction f. Il est parfois préférable de choisir le premier minimum local atteint par la fonction f. Le pas ainsi déterminé est appelé pas de Curry. Ce dernier n’est pas optimal d’un point de vue global mais rentre tout de même dans la classe des méthodes de recherche linéaire exacte.

Ces méthodes de recherche ne sont cependant applicables que lorsque le Hessien de la fonction f est défini positif. De plus, ces méthodes demandent en général beaucoup de temps de calcul sans permettre d’améliorer grandement la convergence de l’algorithme. D’autres méthodes de calcul du pas peuvent être utilisées.

Méthode d’Armijo et de Goldstein

La règle d’Armijo se base sur le choix d’un paramètre m compris entre 0 et 1 et détermine une valeur approchée de αk par la condition :

Le risque de cette méthode est de favoriser les valeurs trop petites de . Par conséquent, elle n’est jamais utilisée seule et est combinée avec la règle de Goldstein.

Cette dernière se base sur le choix de deux paramètres m1 et m2, tous deux compris entre 0 et 1, et détermine les valeurs approchés de αk par deux conditions :

Méthode de Wolfe

La règle de Wolfe est assez similaire à la méthode d’Armijo ou de Goldstein. Elle est basée également sur le choix de deux paramètres m1 et m2, tous deux compris entre 0 et 1, et détermine les valeurs approchés de par deux conditions :

Deux valeurs usuelles des paramètres sont m1=0.1 et m2=0.7.

###### Méthode du gradient (Cabodevilla)

Cette méthode utilise pour direction de descente l’opposée du gradient de la fonction f au point courant x, soit :

où est un vecteur colonne

La fonction f à minimiser peut alors être approchée autour d’un point x0 :

Le point de calcul suivant sera déterminé par :

Pour des raisons de commodités, on notera : .

###### Méthode de gradient conjugué non-linéaire (Bergman, 2010)

Cette méthode est basée sur l’inégalité suivante :

Autour d’un point x0, f0=f(x0) et sont déterminés. L’algorithme est initialisé par une étape de gradient simple donc d0=. Tant qu’un critère de convergence n’est pas vérifié :

1. Détermination d’un pas αk par la méthode de recherche linéaire de son choix et calcul d’un nouvel itéré
2. Evaluation d’un nouveau gradient
3. Calcul du paramètre βk+1 par la méthode de son choix (voir ci-dessous)
4. Construction d’une nouvelle direction de descente
5. Incrémentation : k=k+1

Pour calculer le terme de l’étape 3, 3 méthodes différentes sont décrites ci-dessous :

**Fletcher-Reeves**

**Polack-Ribière**

**Hestenes-Stiefel**

Il est à noter que la méthode de Hestenes –Stiefel est particulièrement efficace dans le cas où la fonction f est quadratique et où la recherche linéaire est effectuée de manière exacte.

###### Méthode de Newton (Cabodevilla)

Cette méthode est en fait l’extension à l’ordre 2 de la méthode précédente. Ainsi, la fonction f à minimiser est approchée par :

Le point où la fonction approchée est minimum est telle que la dérivée de la fonction est nulle soit l’équation :

Le pas s’exprime alors :

En pratique, cette méthode est peu utilisée car la détermination du Hessien est longue et coûteuse en temps de calcul. D’autres méthodes lui sont préférées, fondées sur l’approximation du Hessien.

###### Méthode de Newton-Raphson (Cabodevilla)

Le pas est calculé de la même manière que pour la méthode de Newton. Cependant, au lieu de calculer le Hessien et son inverse, le pas est approximé tel que :

Cette fois, à chaque itération, un système linéaire est résolu à la place d’un calcul d’inverse.

###### Méthode de Levenberg-Markardt (Cabodevilla)

Les deux méthodes précédentes présentent l’inconvénient de ne pas converger lorsque le point initial est loin du minimum recherché. L’algorithme de Levenberg-Markardt propose un compromis entre la méthode du gradient (robuste mais lente à l’approche du minimum) et celle de Newton (peu robuste loin du minimum mais très efficace près). Le pas Δxi s’obtient en résolvant le système suivant :

Ainsi, lorsque λi est grand devant le Hessien, l’algorithme tend vers l’algorithme du gradient. Sinon, l’algorithme converge aussi vite que celui de Newton.

###### Algorithme de Gauss-Newton (A.Rondepierre, P.Weiss, 2013)

Cette méthode, bien que très efficace, est totalement spécifique à la minimisation d’une somme de fonctions au carré. Elle est trop réductrice pour la présente étude ; c’est la raison pour laquelle elle ne sera pas détaillée ici mais répertoriée dans la liste des différentes méthodes qui existent.

###### L’algorithme NLPQL ou BFGS (Cabodevilla)

BFGS sont les initiales de Brayden-Fletcher-Goldfarb-Shanno. Cet algorithme est l’un des plus utilisés dans la littérature. La méthode est la suivante :

A partir d’un point initial x0 et une première approximation du Hessien H0:

1. Calculer le pas par :
2. Déterminer αi optimal par une méthode de recherche linéaire de son choix (décrites ci-dessus) et calculer :
3. Calculer la variation du gradient :
4. Puis la nouvelle estimation du Hessien est donnée par :

###### L’algorithme DFP (X.Antoine, P.Dreyfuss, Y.Privat, 2006-2007)

DFP sont les initiales de Davidon-Fletcher-Powell. C’est une variante de la méthode BFGS. Seule l’étape de mise à jour du Hessien change :

##### Simplex de Nelder & Mead (Cabodevilla)

Parmi les algorithmes d'ordre 0, la méthode d'optimisation Simplex de Nelder & Mead est l'une des plus anciennes connues et est encore aujourd'hui souvent utilisée (fminsearch de matlab). Un Simplex est un tableau de n + 1 vecteurs de n paramètres. La méthode procède à une série de :

* Réflexions (le volume du Simplex reste constant) pour déplacer le barycentre du Simplex en direction du point de critère le plus faible.
* Expansions (le volume augmente) qui étendent le Simplex en direction du point de critère le plus faible.
* Contractions

La méthode de l’algorithme est décrite ci-dessous :

Le vecteur donnant le critère le plus grand à la kème itération xkmax est projeté à travers le barycentre de l’ensemble des autres vecteurs du simplex  :

Le point xr remplace donc le point xkmax pour former un nouveau simplexe. Souvent, le point réfléchi est le symétrique du point xkmax par rapport au barycentre, c’est la raison pour laquelle α=1.

Si l’étape de réflexion donne un point correspondant à une valeur de la fonction inférieure aux valeurs des autres points du simplexe, cette direction est privilégiée et le sommet subit une expansion par la relation suivante :

où γ est laissé au choix de l’utilisateur et varie généralement entre 1 et 2 suivant la convexité de la fonction f.

Si, le point xrk est remplacé par le point xexpank dans le nouveau simplexe. Dans le cas contraire, le point xrk est conservé.

Au contraire, si l’étape de réflexion donne encore un point correspondant à la valeur maximum de f sur le simplexe, le mouvement est contracté :

où β est laissé au choix de l’utilisateur et varie entre 0.2 et 1.

Si, le point xrk est remplacé par le point xcompk dans le nouveau simplexe. Dans le cas contraire, un mouvement de réduction de tous les vecteurs du simplexe en direction du vecteur donnant le critère le plus faible xkmin est effectué afin d’adapter le simplexe à la topologie du problème étudié.

Si, le simplexe initial subit une réduction : les étapes 1, 2 et 3 sont ignorées.

Si, le simplexe réfléchi subit une réduction : les étapes 2 et 3 sont ignorées.

Le choix des constantes α et β est très important dans le déroulement du processus de résolution. Pour ne pas prendre trop de risques, un choix de ces paramètres autour de 1 peut souvent être judicieux.

Cette méthode fonctionne bien à priori, mais aucune preuve mathématique n’assure la convergence pour une fonction dont la dimension de l’intervalle de départ est supérieure à 2.

##### Méthode de recherche multi-directionnelle (Bergman, 2010)

Ce type de méthodes consiste à effectuer des recherches unidirectionnelles suivant des directions orthogonales, formant une base de l’espace considéré, idéalement choisies. La méthode la plus évidente pour un problème de recherche multi-directionnelle est de choisir pour directions de recherche la base canonique de. Une minimisation unidirectionnelle doit donc être réalisée dans chaque direction successive, et ceci jusqu’à convergence. Ce type d’algorithme présente le défaut majeur d’avoir un comportement oscillant (recherche suivant des directions orthogonales) menant au minimum.

#### Avec contraintes

Soit le problème d’optimisation sous contraintes suivant :

##### Les méthodes de projection

L’une des façons de résoudre un problème d’optimisation sous contraintes est d’adapter un algorithme de descente de façon à respecter, à chaque pas de l’itération, les contraintes. Pour respecter ces contraintes, il est possible d’appliquer une projection sur l’espace entier afin de rester dans le bon domaine. Comme K est un ensemble convexe et fermé, il existe un unique vecteur y de K appelé « projection de x sur K » et défini comme solution du problème de minimisation :

Ce vecteur sera noté.

Ainsi, si une approximation xk de K est supposée connue, comme pour un algorithme de descente classique, la direction de descente dk et αk sont déterminés. L’approximation suivante est ensuite définie par une projection :

###### Méthode du gradient projeté (X.Antoine, P.Dreyfuss, Y.Privat, 2006-2007)

Soit. A partir d’un point initial x0, tant que le critère de convergence n’est pas respecté :

1. Calculer le pas αk par la méthode de son choix
2. Estimer le nouvel itéré
3. Itération sur k : k=k+1

###### Méthode de Newton projetée (X.Antoine, P.Dreyfuss, Y.Privat, 2006-2007)

Cette méthode est utilisée lorsque le concepteur rencontre un problème avec des contraintes de bornes.

Elle ressemble très fortement à la méthode de Newton décrite ci-dessus sauf qu’au moment du calcul du nouvel itéré, il faut appliquer une projection sur l’ensemble des contraintes. Ainsi :

##### Méthode de Lagrange Newton (X.Antoine, P.Dreyfuss, Y.Privat, 2006-2007)

Cette méthode est généralement utilisée pour des problèmes à contraintes en égalité. Soit le problème suivant :

Le Lagrangien du problème s’écrit :

Utiliser la méthode de Lagrange Newton revient à résoudre le système suivant par la méthode de Newton :

##### Méthode de pénalisation (X.Antoine, P.Dreyfuss, Y.Privat, 2006-2007)

Les méthodes de pénalisation sont très utilisées en pratique car elles sont relativement simples d’utilisation. Elles partent du principe suivant : un problème avec contraintes peut être remplacé par un problème sans contraintes. Par exemple, le problème sans contraintes suivant :

peut être remplacé par le problème sans contraintes ci-après :

où α est une fonction de pénalisation des contraintes et ε>0. Le but est alors de trouver des fonctions α telles que les deux problèmes présentés précédemment soient équivalents, c’est-à-dire qu’ils aient les mêmes solutions. Dans ce cas, la pénalisation est dite exacte.

En général, ce cas est très rare car il faut que la fonction de pénalisation soit dérivable afin d’appliquer les techniques de résolution sans contraintes. Ainsi, la pénalisation dite inexacte est employée. En fait, l’ensemble des solutions du problème sans contrainte ne couvrira pas celui du problème contraint. Néanmoins, des fonctions α dérivables peuvent être trouvées, ce qui permet d’utiliser des résultats de minimisation sans contraintes.

Par exemple, si la contrainte imposée est x≤0, alors α est définie telle que :

##### Méthode d’Uzawa (X.Antoine, P.Dreyfuss, Y.Privat, 2006-2007)

La méthode d’Uzawa repose sur l’utilisation du Lagrangien. En effet, la fonction Lagrangienne permet d’englober la fonction f du problème ainsi que les contraintes du problème, nommées g et h. Le Lagrangien de f est défini tel que :

Par la suite, la notion de point selle doit être définie. Un point selle de L sur RnxRpxR+m est un triplet (x\*,μ\*,λ\*) de RnxRpxR+m qui vérifie l’équation :

Ainsi, pour (μ\*,λ\*) fixé, le minimum sans contraintes de la fonction est recherché et pour x\* fixé, le maximum de la fonction est recherché. La suite des x trouvée est assurée de converger vers une solution.

### Les méthodes stochastiques

Dans le cas où la fonction objectif n’est pas dérivable et/ou continue, les méthodes stochastiques sont tout à fait appropriées [Collignan, 2011]. Ces méthodes se basent sur la génération de nombres aléatoires et reproduisent des comportements issus de la nature pour identifier une solution optimale. Cependant, elles ne garantissent pas d’atteindre un minimum global mais permettent de s’en approcher. Ces méthodes considèrent les solutions comme des individus en confrontation ou en coopération qui échangent des informations ou des caractéristiques au moyen d’opérateurs reproduisant des comportements naturels. De tels processus itératifs nécessitent néanmoins des temps de calcul longs. Ces méthodes sont également efficaces pour des modèles de type « boîte noire » dont le contenu est ignoré.

Les méthodes stochastiques les plus utilisées seront décrites ci-dessous (liste non exhaustive).

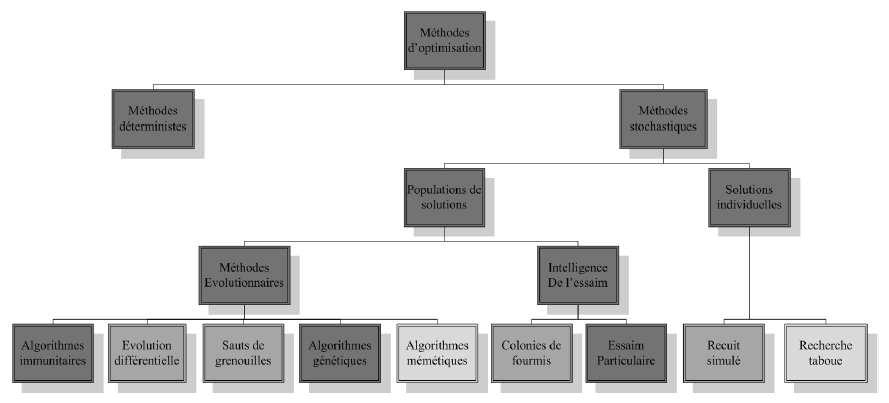


Figure : Classification des méthodes d'optimisation

#### Populations de solutions

##### Algorithmes évolutionnaires

Ce type d’algorithme s’inspire de la théorie Darwinienne de l’évolution des espèces pour résoudre des problèmes divers. Ils font évoluer un ensemble de solutions à un problème donné dans l’optique de trouver les meilleurs résultats Bien que ces algorithmes soient performants lors d’optimisation dynamiques, leur temps de convergence peut être long. Malgré le fait que leur nombre soit conséquent, seuls les algorithmes les plus utilisés dans la littérature seront répertoriés.

###### Algorithmes génétiques (Collignan, 2011)

Le principe des algorithmes génétiques est de générer une population de solutions candidates et de la faire évoluer, imitant la sélection naturelle ainsi que des processus génétiques. Ici, les solutions candidates sont vues comme des individus tandis que les variables de conception constituent le génome d’un individu.

L’algorithme sélectionne d’abord des individus à partir d’une population évaluée. L’opérateur sélectionne les individus en favorisant les meilleurs tout en laissant la possibilité de sélectionner des individus moins bons. Ces individus sont ensuite rassemblés en couple puis deux nouveaux individus (les « enfants ») sont générés par croisement à partir des deux anciens (les « parents »). Ce croisement simule la reproduction donc les enfants sont constitués aléatoirement de gènes de leurs parents. L’opérateur de croisement s’active moyennant une probabilité pour chaque couple notée probC. Cette activation est gérée via la génération d’un nombre aléatoire puis par comparaison à la valeur de probC. En cas d’activation, une double coupure est effectuée : ces deux coupures générées aléatoirement encadrent une séquence de gènes dans le génome des deux parents.

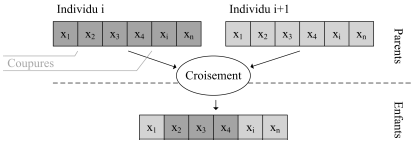


Figure : Génération d'un individu enfant

Cette opération est répétée une nouvelle fois pour obtenir les deux « enfants » qui remplacent leurs parents dans la population d’individus. Enfin, la mutation modifie aléatoirement un gène afin d’introduire de nouveaux individus dans le processus. Cependant, cette mutation reste généralement marginale. En effet, l’opérateur de mutation ne s’active que moyennant une certaine probabilité pour chaque individu notée probM. L’opérateur de mutation sélectionne aléatoirement un gène parmi ceux de l’individu considéré et une nouvelle valeur pour ce gène est tirée de manière aléatoire dans le domaine de valeurs considérées. 4 paramètres assurent donc le pilotage de l’algorithme génétique :

* Le nombre d’individus
* La pression de sélection
* La probabilité de croisement
* La probabilité de mutation

###### Algorithmes immunitaires (Collignan, 2011)

Ce type d’algorithme est basé sur une analogie au système immunitaire. Ce dernier est composé de cellules immunitaires nommées anticorps. Elles sont différenciées par les différents récepteurs protéiques situés en surface. Lorsque des cellules indésirables nommées « antigènes » sont détectées, l’organisme génère un grand nombre d’anticorps différents. Leur efficacité est proportionnelle à leur affinité aux antigènes. Cette affinité dépend de la compatibilité entre les récepteurs protéiques des antigènes et des anticorps. L’organisme identifie ensuite les anticorps ayant la meilleure affinité avec la menace. Ils sont ensuite clonés et maturés afin de modifier leurs marqueurs en espérant obtenir des anticorps encore plus performants tout en générant de nouveaux anticorps. Au fil du temps, les anticorps correspondent de plus en plus aux antigènes jusqu’à leur élimination. Enfin, les meilleurs anticorps sont gardés en mémoire en cas d’agression ultérieure.

Ici, un anticorps correspond à une solution candidate, les variables de conception sont ses gènes, directement assimilés à ses récepteurs protéiques. Cet algorithme dépend de 5 opérateurs détaillés ci-après :

* La sélection clonale : cet opérateur est chargé de sélectionner une partie des anticorps après classement en fonction de leur affinité avec l’antigène. Seuls les n meilleurs sont sélectionnés.
* La multiplication clonale : cet opérateur dépend d’un taux de clonage noté Tc et qui duplique Tc fois les n anticorps sélectionnés.
* La maturation : Pour chaque clone obtenu, l’opérateur de maturation sélectionne une séquence de gènes et en modifie la valeur.
* La mémorisation : cet opérateur ne garde qu’une partie du grand nombre de clones maturés obtenus au travers des opérateurs précédents.
* La génération : cet opérateur crée un grand nombre de nouveaux anticorps afin de compléter la nouvelle population tout en gardant un nombre total constant.

###### L’évolution différentielle (Lepagnot, 2011)

L'évolution différentielle est inspirée par les algorithmes génétiques et les stratégies évolutionnistes combinées avec une technique géométrique de recherche. Les algorithmes génétiques changent la structure des individus en utilisant la mutation et le croisement, alors que les stratégies évolutionnistes consistent à biaiser un opérateur de mutation, appliqué à un individu, en fonction des différences calculées entre d’autres individus sélectionnés aléatoirement. Ces idées ont été mises en œuvre par K. Price et R. Storn [Storn & Price, 1995]. Depuis, l'évolution différentielle est devenue une méthode utilisée pour une grande quantité de problèmes.

##### Intelligence de l’essaim

Chez les insectes sociaux, le comportement collectif qui émerge des comportements simples des individus est nommé intelligence en essaim. Certains principes présents dans la nature ont inspiré des méthodes d’optimisation décrites ci-après.

###### Essaim particulaire (OEP) (Collignan, 2011 etLepagnot, 2011)

L’OEP s’inspire du comportement social des animaux évoluant en essaim (nids d’oiseaux, bancs de poissons …). Un individu de l’essaim ne dispose que d’une connaissance locale de sa situation dans l’essaim. Il utilise cette information ainsi que sa mémoire pour décider de son déplacement. Des règles simples telles que « aller dans une même direction » ou « rester proche de ses voisins » suffisent à maintenir la cohésion de l’essaim.

L’OEP considère en fait un essaim de particules, qui sont des solutions potentielles au problème d’optimisation, qui « survole » l’espace de recherche pour trouver un optimum global. Le déplacement d’une particule est influencé par trois composantes :

* L’inertie : la particule suit sa direction courante de déplacement
* La cognition : la particule tend à se fier à sa propre expérience et se dirige vers le meilleur site par lequel elle est déjà passée
* La sociabilité : la particule se fie à l’expérience de ses congénères et tend à se diriger vers le meilleur site déjà atteint collectivement par l’essaim

Dans l’espace de recherche, la particule i est modélisée par un vecteur position xi et un vecteur vitesse vi. La qualité de sa position est déterminée par la valeur de la fonction objectif en ce point. Cette particule garde en mémoire la meilleure position par laquelle elle est déjà passée, notée pi. La meilleure position atteinte par ses particules voisines est également répertoriée, notée gi. A chaque instant t, le vecteur vitesse est calculé tel que :

où w est en général une constante appelée coefficient d’inertie, c1 et c2 sont deux constantes, appelées coefficients d’accélération et r1 et r2 sont deux nombres aléatoires entre 0 et 1 tirés uniformément à chaque itération.

On peut ensuite en déduire la nouvelle position de la particule :

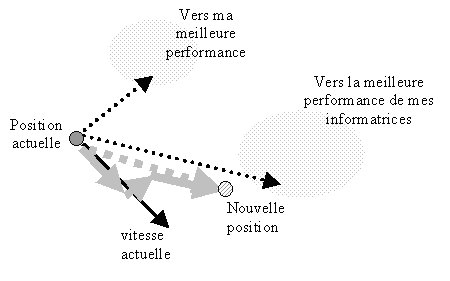


Figure : Algorithme de l'essaim particulaire

Cet algorithme est performant mais présente les mêmes défauts que les méthodes évolutionnaires : le temps de convergence peut être long.

###### Colonie de fourmis (ACO) (Lepagnot, 2011)

Comme l’OEP, l’ACO est une approche utilisant l’intelligence en essaim et découlant des comportements sociaux d’une fourmilière. En effet, les fourmis ont la capacité de communiquer entre elles par le dépôt de substances chimiques sur le sol : les phéromones. Si un obstacle est disposé entre le nid et la fourmilière, le chemin le plus court sera désigné par les fourmis qui sont retournées le plus rapidement au nid en passant par la source de nourriture. Il en découle que la quantité de phéromones déposées par unité de temps sur ce trajet est plus importante que les autres. Sachant qu’une fourmi est d’autant plus attirée à un endroit que le taux de phéromones y est important, toutes les fourmis finiront par emprunter le chemin le plus court pour accéder à la source de nourriture.

Pour chaque fourmi k, le trajet d’une ville i à une ville j dépend de :

* La liste des déplacements possibles
* L’inverse de la distance entre les points appelée visibilité, ηij=1/dij
* La quantité de phéromones déposées entre ces deux points, appelée intensité de la piste τij. Cette quantité dépend de la qualité de la solution trouvée.
* L’évaporation des phéromones

La règle de déplacement est la suivante :

où α et β sont deux paramètres contrôlant l’importance relative de l’intensité de la piste et de la visibilité.

Après un tour complet de l’obstacle, chaque fourmi dépose une certaine quantité de phéromone sur l’ensemble de son parcours. Cette quantité dépend de la qualité de la solution trouvée et est définie par :

où est le trajet effectué par la fourmi k à l’itération t, est la longueur de et Q est un paramètre de réglage.

Enfin, le processus d’évaporation des phéromones (pour éviter de trouver des optima locaux) peut être modélisé de la façon suivante :

où , m le nombre de fourmis et ρ est un paramètre de réglage.

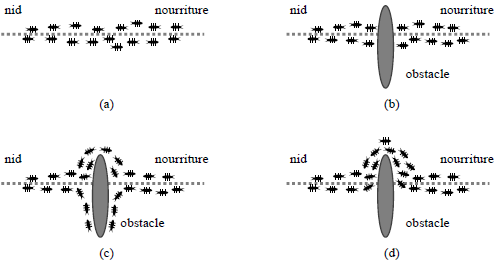


Figure : Recherche du meilleur chemin pour les fourmis

##### Recuit simulé (Cabodevilla)

La méthode du recuit simulé (Simulated Annealing) repose sur un principe de la thermodynamique : la loi de refroidissement des métaux. Un atome donné est animé par un mouvement brownien dont l'amplitude dépend de la température. Lorsque la température décroît ce mouvement s'estompe et l'atome tend à se positionner dans un état d'énergie minimale.

Le problème posé est la détermination d'un vecteur x qui minimise le critère λ(x). Après une initialisation aléatoire du premier point de calcul x0, l'algorithme génère, à chaque itération, un pas aléatoire δx suivant une loi de probabilité de type gaussienne. Le nouveau point de calcul est obtenu tel que :

Deux cas peuvent alors se présenter :

* Le nouveau point est meilleur :

Le point est accepté et le processus recommence

* Le nouveau point est moins bon :

Le nouveau point peut être accepté suivant une probabilité

A chaque itération, la température T(k) diminue suivant la loi :

Au fil du temps, l'algorithme finit par générer des pas de faible amplitude et n'accepter pratiquement plus de pas qui font augmenter le critère, les pas qui le font diminuer étant toujours acceptés. Cette méthode permet statistiquement de trouver un minimum global mais demande un très grand nombre d’itérations.

#### Approche hybride

Certains auteurs ont proposé des algorithmes hybrides OEP/Algorithmes Evolutionnaires [Lepagnot, 2011]. Cependant, les performances de ces algorithmes ne diffèrent pas significativement de celles des algorithmes basés sur l’OEP ou sur les principes des algorithmes évolutionnaires.

## Les problèmes à grand nombre de variables en conception

La plupart des problèmes étudiés comportent un grand nombre de variables de conception. L’optimisation étant incapable de traiter une centaine de variables à la fois, il faut réduire le nombre de VCo.

### La notion de parcimonie

La problématique d’un grand nombre de VCo peut être reliée à la parcimonie. La parcimonie, telle que définit par Vernat [Vernat, 2004] reflète la simplicité du problème ainsi que sa facilité d’utilisation et d’exécution. Elle est souhaitée aussi haute que possible. Vernat propose d’estimer la parcimonie à partir du nombre de relations et du nombre de variables du modèle :

Dans cette expression, le nombre de variables a autant d’impact que le nombre de relations. Cette définition n’est donc pas vraiment adaptée aux problèmes d’optimisation. En effet, augmenter le nombre de variables fait croître de manière conséquente l’espace des solutions candidates, contrairement au nombre de relations. Ainsi, dans un problème d’optimisation, il faudrait que la parcimonie soit inversement proportionnelle au nombre de variables. Mais cette notion constitue tout de même un indicateur qu’il faut parfois prendre en compte.

### Réduction et approximation de modèles

Le but recherché ici est de remplacer un modèle initial par un autre modèle plus rapide (c’est-à-dire plus parcimonieux car le nombre de variables et de relations auront diminué) et approché (la précision et l’exactitude diminueront). Pour cela, cette méthode nécessite généralement un ensemble de données (VCo et/ou VObs) issues de simulation du modèle initial que l’on veut réduire ou d’expérimentation de laquelle il faut extraire une modélisation. Dauvergne distingue deux familles [Dauvergne, 2008] :

* Les techniques internes : les variables les plus influentes sont conservées tandis que les moins influentes sont négligées
* Les techniques externes utilisent le principe des boîtes noires : les liens entre les variables d’entrée et de sortie du modèle ne sont pas connus

La réduction de modèle vise avant tout à gagner du temps mais cela n’implique pas forcément de réduire le nombre de variables de conception en entrée du modèle. Un modèle réduit a une parcimonie plus importante qui est compensée par une baisse de son exactitude. Une approche de gestion par lots peut également être envisagée.

### La gestion par lots

Le principe de la gestion par lots est de procéder à un regroupement de variables entre elles. Pour cela, plusieurs techniques existent.

#### L’approche par interpolation

L’approche par interpolation propose de reconstituer une information globale à partir d’un faible échantillon d’informations. En général, l’interpolation polynomiale de Lagrange [Collignan, 2011] est la plus utilisée. Une fois la fonction construite, il est possible de connaître l’emplacement de points qui n’étaient pas dans l’échantillon initial.

Ainsi, en utiliser une méthode par interpolation permet d’instancier automatiquement l’ensemble des VCo du lot en n’instanciant initialement qu’une partie d’entre elles. Toutefois, il faut choisir judicieusement les VCo à instancier.

#### L’approche par composition

Cette méthode proposée par Ledoux [Ledoux 2011] permet de définir les déformations d’une surface par ses modes propres. Plusieurs déformations sont ensuite sommées pour simuler le défaut de forme de la surface. Une grande variété de courbes peut être obtenue par composition mais cette méthode est assez complexe et ne sera pas détaillée ici.

# Première optimisation sur un modèle thermique

## Contexte

a déjà fait l’objet d’études lors de travaux précédents. Cette pièce a été mise en place sur banc d’essai afin d’orienter le jet de gaz chaud en sortie de tuyère d’un moteur à propergol solide. Afin de vérifier sa tenue thermomécanique, des cartouches ont été placées selon la configuration de la figure suivante afin de simuler ce jet pour que, à terme, la simulation remplace les essais afin de gagner du temps.

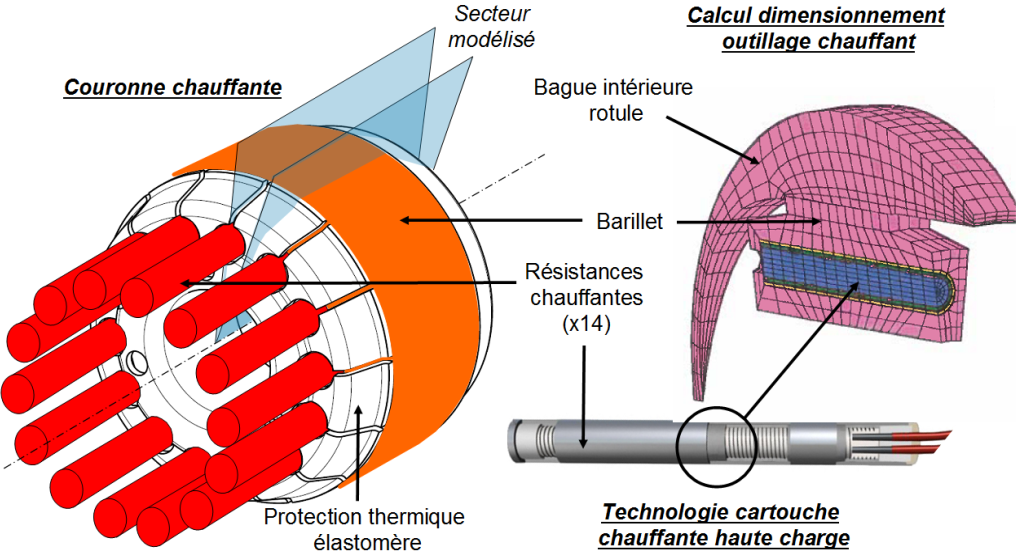


Figure : Configuration de l'essai thermomécanique de la

La justification de cette pièce a été réalisée *via* une étude numérique précédente. Il s’agit ici de reprendre ces travaux dans le cadre d’une campagne expérimentale pour comparer ces résultats avec ceux obtenus numériquement et recaler si besoin les modèles de calcul thermique et thermomécanique. Les logiciels utilisés pour réaliser cette étude sont Mentat 2011, Marc 2011 et Noesis Optimus 10.10.

Pour cela, plusieurs capteurs de température sont disposés en différents points d’intérêt de la pièce selon la figure suivante.

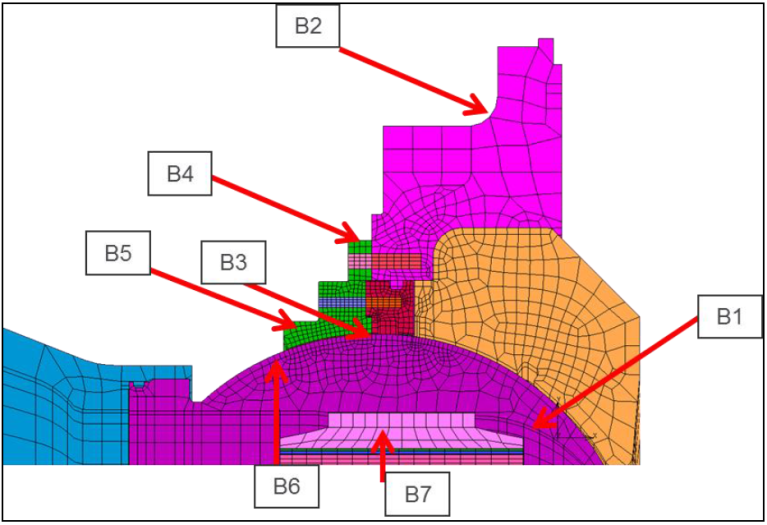


Figure : Disposition des capteurs de température

Une puissance de chauffe de est appliquée. Au cours de l’essai, la pièce est tout d’abord statique puis sollicitée mécaniquement avec des angles de plus ou moins degrés. L’objectif est de recaler le calcul thermique sur les résultats de la chauffe de kW puis, à long terme, d’établir une méthode générale d’optimisation paramétrique en simulation numérique.

Afin de simplifier le problème, le modèle a été rendu approché : l’étude se fera en 2D axisymétrique pour des raisons pratiques d’itération et les températures aux thermocouples seront considérées comme ponctuelles.

Le but de cette première optimisation est de comprendre le fonctionnement des algorithmes afin de mieux les utiliser et de voir le(s)quel(s) s’adapte(nt) le mieux au problème. En effet, pour un calculateur, le but sera d’être le plus proche de la solution réalisable tout en limitant le temps de mise en œuvre de la boucle d’optimisation et en minimisant le temps de calcul.

## Déroulement

Le modèle numérique est tout d’abord implémenté sous Optimus. Ce modèle dépend de 8 paramètres distincts :

* La conductivité de la magnésie, isolant thermique
* La conductivité du Sarflon, revêtement appliqué sur la surface de la bague intérieure pour limiter les frottements sans utiliser de lubrifiant
* La conductivité du contact Barillet-Cartouche
* La conductivité du contact Barillet-Rotule
* Le coefficient de convection naturelle au niveau du barillet
* Le coefficient de convection naturelle au niveau de l’espace confiné
* Le coefficient de convection naturelle dans le reste du système
* Le coefficient d’émissivité

Il s’agit ici de cibler les valeurs données par l’essai afin de recaler le modèle numérique par rapport à la réalité. Plusieurs algorithmes d’optimisation seront utilisés, le but étant de faire un comparatif des résultats d’optimisation afin de déterminer quelle méthodologie est la plus pertinente. Les bornes définies pour les variables d’optimisation sont fixées à :

* pour les variables de conduction : la valeur initiale
* pour les variables de convection :
* pour la variable d’émissivité :

De telles bornes sont choisies afin de balayer un grand espace de solution. En effet, l’ingénieur ne sait pas dans quelle direction aller donc préfère viser large. De plus, des simplifications de modèle ont été opérées (2D axisymétrique, températures de chauffe ponctuelles…) ce qui peut éloigner le calcul de la réalité. Viser large permet de ne pas se tromper et de ne pas relancer d’autres calculs derrière.

L’algorithme sélectionné peut ensuite être lancé, les paramètres de ce dernier étant précieusement notés. Optimus fournit l’optimum, c’est-à-dire le point qui minimise la fonction objectif (cette fonction objectif étant inconnue pour l’utilisateur car il utilise un modèle « boîte noire » pour l’optimisation). A partir de là, des traitements statistiques sont effectués sur le modèle optimisé afin de le valider par rapport aux essais ou non.

## Traitement des résultats

Le traitement des résultats est exclusivement porté sur les températures optimisées puisque c’est le modèle ciblé par le concepteur. Il est cependant important de connaître les données en amont puis d’analyser de manière critique les fichiers de sortie. Plusieurs méthodes d’évaluation des donnés existent pour la validation des modèles amonts et avals suivant les différents algorithmes d’optimisation qui ont tourné. La fonction objectif donne un premier aperçu : plus elle est faible, plus le modèle se rapproche de la réalité. Cependant, elle ne suffit pas à conclure. Un rapide test de Student [Tahan, 2012] permet de comparer deux distributions de variables numériques : l’une empirique correspondant au modèle cible (résultats d’essai) et l’autre correspondant au modèle calculé. Le test consiste à évaluer la différence δi entre les deux distributions. Il convient ensuite de calculer la moyenne empirique de ces différences puis ensuite d’estimer l’écart type de cette distribution de différences à l’aide de la formule :

Et enfin d’effectuer le test en calculant la statistique T définie par :

Le modèle est invalidé (rejet de l’hypothèse nulle H0) au seuil de signification α (ici, 95%) si :

Où est la valeur lue pour le seuil de signification dans la table de Student à n-1 degrés de liberté.

Si les données sont centrées normées, l’erreur quadratique moyenne entre le modèle optimisé et le modèle ciblé peut être calculée : elle doit tendre vers 0 pour que le modèle soit validé.

Le test de Xhi² n’est pas retenu car il n’est pas pertinent dans tous les cas. En effet, certaines valeurs relevées par les capteurs sont inférieures à 10. Or dans une table 8X2 ou 6X2 de Chi², le nombre de degré de liberté est toujours égal à 1.

Enfin, il est possible d’afficher un pourcentage de vraisemblance entre les deux modèles en traçant le modèle cible par rapport au modèle optimisé : c’est le diagramme de parité. Il suffit ensuite de calculer le coefficient de détermination R² qui doit tendre vers 1 pour un modèle qui correspond à la réalité. Une fois cette régression effectuée, une analyse des résidus peut être envisagée. Si le modèle est bon, les écarts constatés (c’est-à-dire les résidus) sont imputables à des erreurs de mesure. Les résidus doivent donc posséder les propriétés classiques d’une distribution normale. Ainsi, plusieurs tests statistiques peuvent être envisagés :

* L’histogramme des résidus doit suivre une loi normale
* Le test statistique de Durbin Watson doit être proche de 2 (aucune autocorrélation des résidus)
* La distribution des valeurs résiduelles en fonction des valeurs ajustées doit présenter une dispersion homogène et doit être centrée autour de 0
* Les résidus doivent s’aligner selon la droite de Henry
* Le test de Kolmogorov Smirnov (ou de Shapiro Wilk) s’assure de la normalité des résidus

Pour déterminer la meilleure solution, il faut établir un système de classement entre les différents algorithmes d’optimisation. Pour cela un système de classement par points a été utilisé (basé sur la matrice de jugement [Saaty, 1990]). Chaque facteur ou test statistique est appliqué à tous les modèles de chaque algorithme. Leurs valeurs sont ensuite recodées entre 0 et 1. La meilleure valeur se voit attribuée la note de 10 et la moins bonne, 0 (pour coder ce système automatiquement, il suffit de prendre la partie entière de 10 fois la valeur normée entre 0 et 1). Une fois toutes les notes attribuées pour chaque distribution et chaque algorithme, la note totale est calculée. L’algorithme qui a la meilleure note est celui le plus adapté au problème.

Ainsi, deux méthodes se détachent des autres :

* L’algorithme de descente classique NLPQL : il est très précis et surtout rapide à fournir des résultats. Sa faiblesse réside seulement dans le fait qu’il risque de trouver des minima locaux et non globaux
* L’algorithme génétique qui constitue la meilleure méthode stochastique pour ce problème : il est également très précis et trouve systématiquement le minimum global. Cependant, son utilisation peut être très longue (plus de deux jours). Un post traitement des résultats par un NLPQL doit être effectué pour déterminer le vrai minimum global.

Par conséquent, une première stratégie peut être mise en place :

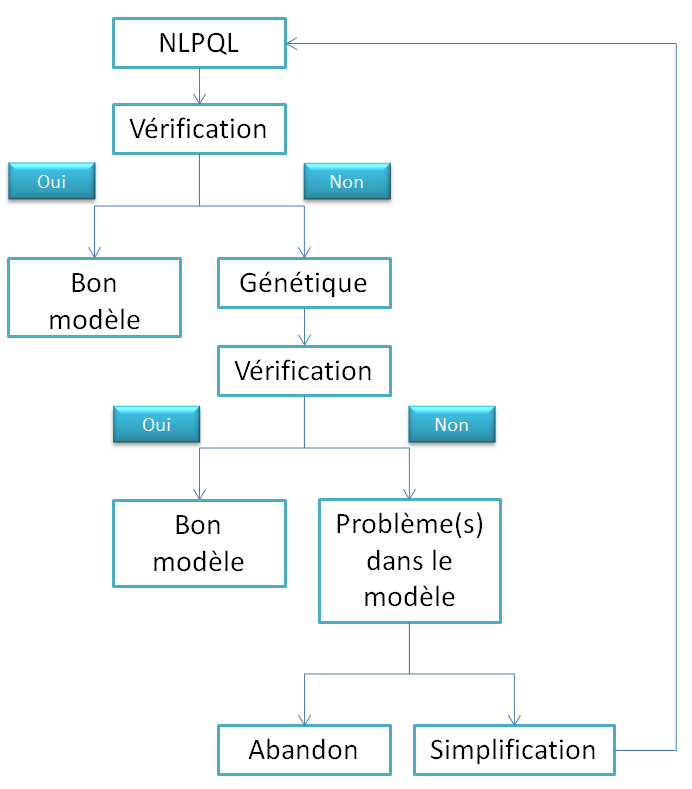


Figure : Première méthode d'optimisation proposée

Un NLPQL est lancé au début dans l’éventualité où le concepteur est déjà proche d’une solution optimisée. Si ce n’est pas le cas, il peut lancer un algorithme génétique, certes long, qui aura plus de chances de tomber sur le minimum global. Les tests statistiques détermineront alors si le modèle correspond aux attentes ou non. Si non, il faudra envisager une simplification ou une réduction de modèle. Pour cela, le concepteur a la possibilité de faire un plan d’expérience (*Design Of Experiment* en anglais) afin de mieux connaître la forme de la surface de réponse puis une analyse de variance pour déterminer quel(s) paramètre(s) d’optimisation est/sont le(s) plus influent(s). Il faut ensuite relancer le modèle thermique en optimisant seulement ce(s) paramètre(s). Avec cette technique, il est possible d’avoir de bons résultats tout en diminuant les temps d’optimisation.

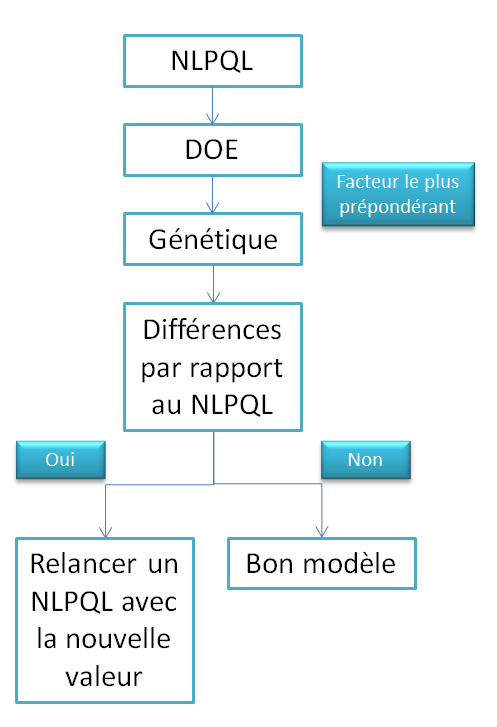


Figure : Seconde méthode d'optimisation proposée

Pour en revenir au système rotulé, une fois le modèle déterminé pour chaque capteur de température, il s’agit de le valider. La difficulté dans ce cas, est que chaque capteur n’a relevé au maximum que 8 températures (6 dans certains cas). En ne gardant qu’une seule valeur pour la validation et les autres pour l’apprentissage, le concepteur peut procéder à la technique du *Leave one out* :

* Apprentissage sur n-1 observations
* Validation sur la nième observation
* L’opération est répétée n fois
* Le score de chaque modèle est calculé par l’erreur quadratique moyenne
* Le modèle qui présente le meilleur score est sélectionné

Cependant, les résultats sont décevants : Par exemple, le meilleur modèle optimisé de température B6 proposé par l’algorithme SQP n’est pas satisfaisant : le coefficient de détermination par rapport à l’objectif n’est que de 60%, ce qui est trop faible. Cela est dû au fait du trop faible nombre de températures relevées. Par la suite, toutes les données seront utilisées pour faire l’apprentissage du modèle.

Egalement, un test statistique (Student) a mis en avant une incohérence dans les valeurs de température relevées par rapport au modèle thermique proposé. En effet, pour le capteur B3, le test est quasiment toujours faux pour n’importe quel algorithme d’optimisation. Cela peut provenir d’un mauvais respect du protocole d’essai. De plus, le capteur 3 occupe une position sensible et a alors relevé des valeurs erronées. Il a donc été décidé par la suite de ne pas tenir compte des mesures du capteur, l’optimisation du modèle se faisant sur les 5 autres. La partie suivante présente, dans un premier temps, les résultats d’optimisations faites avec les températures du capteur B3, puis, dans un deuxième temps, sans les valeurs du capteur B3.

## Présentation des résultats

Les résultats du calcul thermomécanique doivent correspondre au mieux aux températures relevées par les capteurs au cours du temps. L’algorithme d’optimisation va donc modifier les paramètres d’entrée du modèle pour adapter les résultats à ceux de la campagne expérimentale. Une première optimisation, prenant en compte le modèle thermique complet (capteur B3 inclus), a été pratiquée afin de comparer l’efficacité des différents algorithmes. Les tests statistiques retenus sont le *Mean Square Error*, le coefficient de détermination du diagramme de parité et la fonction objectif. Les scores des différents algorithmes sont résumés ci-après :

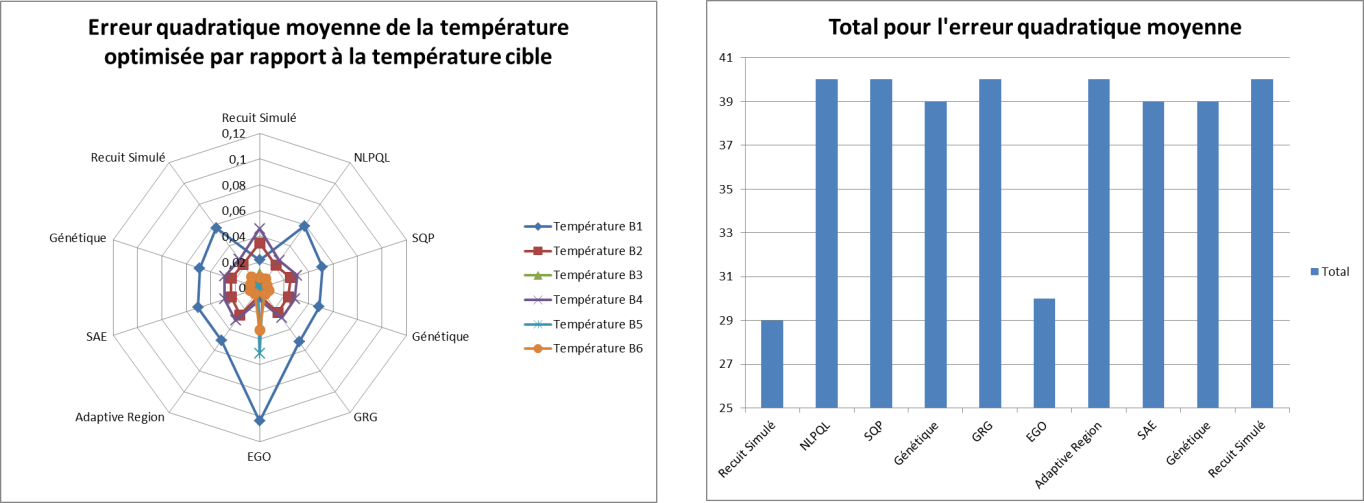


Figure : Scores des différents algorithmes pour l'Erreur Quadratique Moyenne

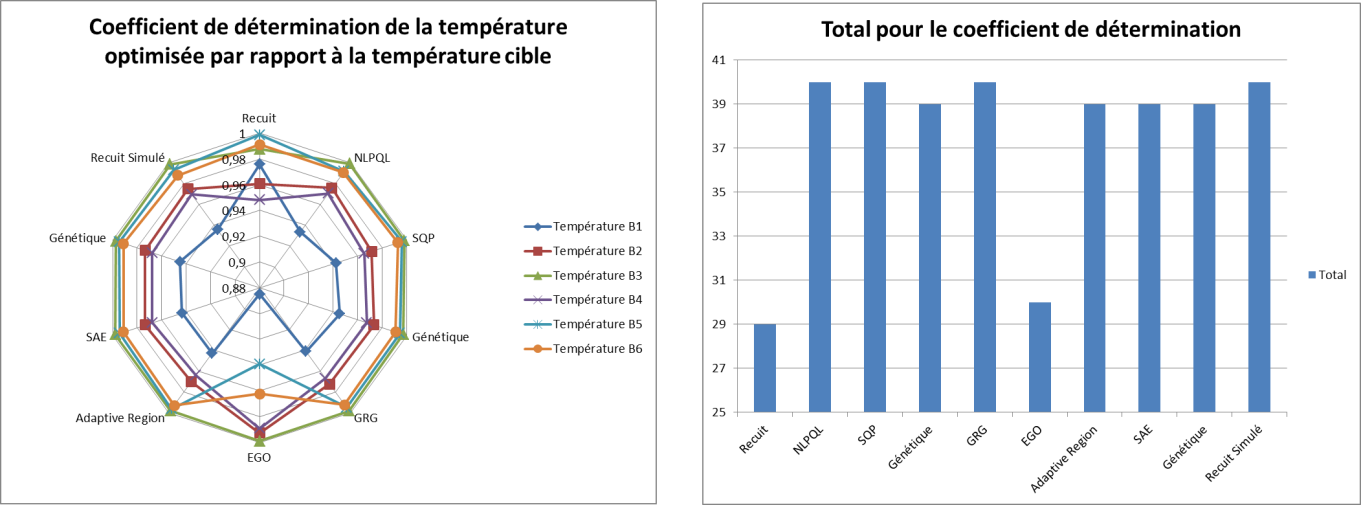


Figure : Scores des différents algorithmes pour le Coefficient de Détermination

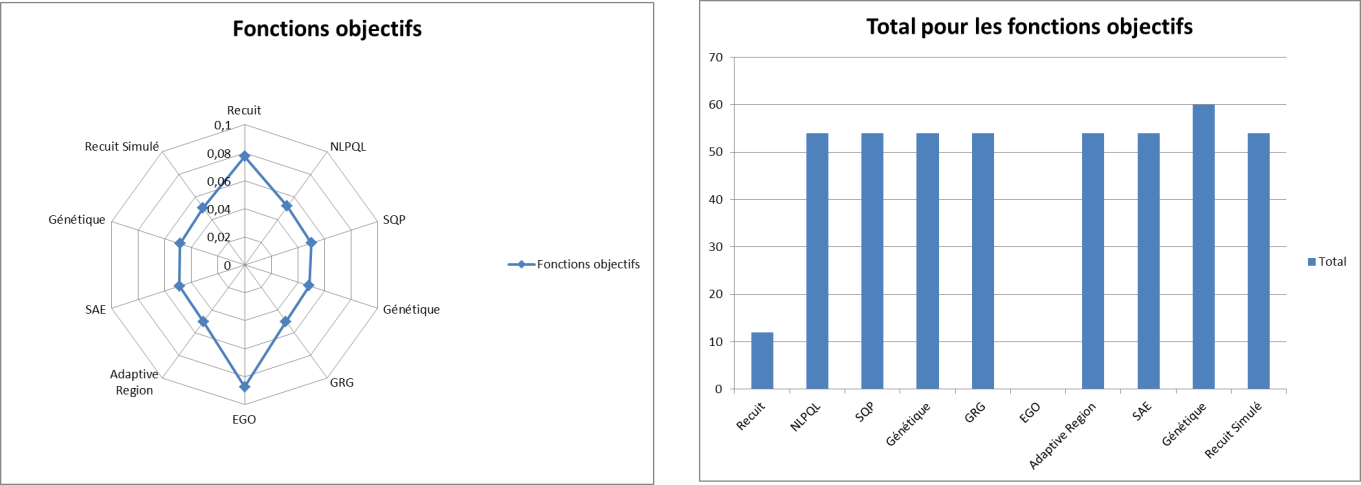


Figure : Scores des différents algorithmes pour la fonction objectif

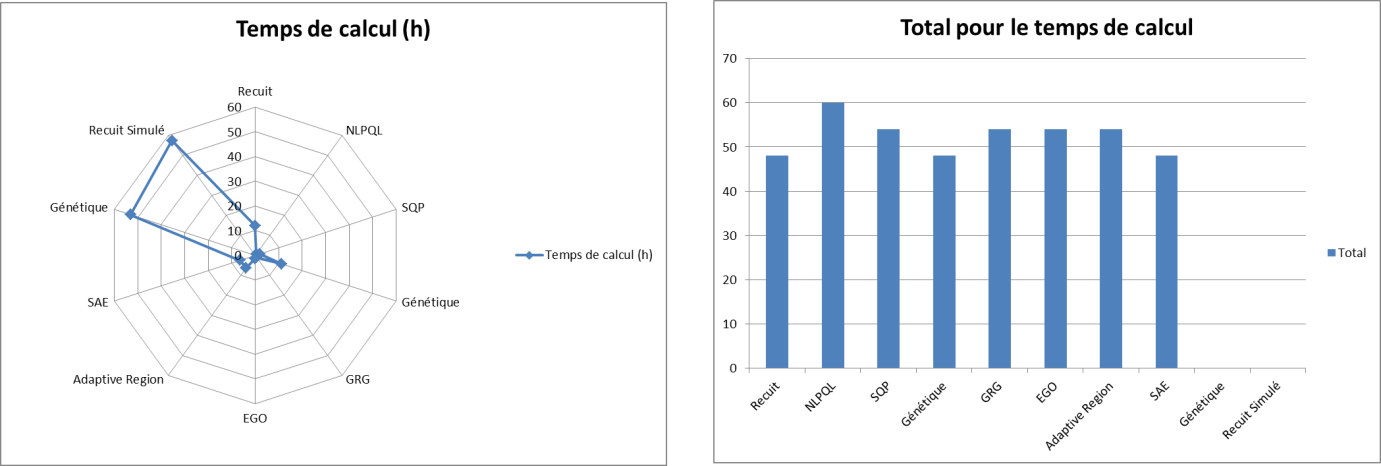


Figure : Scores des différents algorithmes pour le temps de calcul

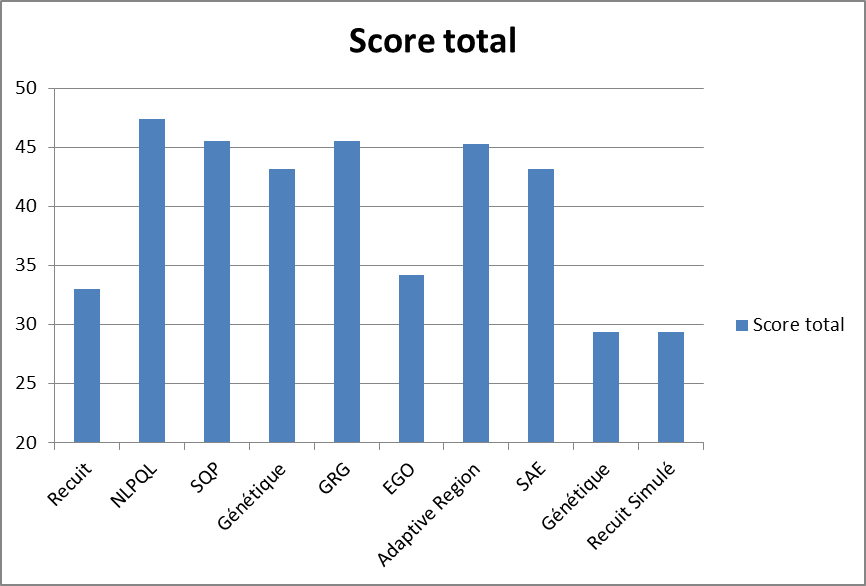


Figure : Scores totaux des différents algorithmes

Le score total a été calculé en pondérant les sous totaux selon la manière suivante :

* 0.3 affecté au score de l’erreur quadratique moyenne, du coefficient de détermination. En effet, ces tests sont des bons indicateurs de précision des différents modèles.
* 0.3 affecté au temps de calcul. Le temps de calcul doit être court pour pouvoir exploiter des résultats rapidement, sans avoir à attendre plusieurs jours.
* 0.1 affecté à la fonction objectif. Sa valeur doit être minimale.

Aux vues des premiers résultats, l’algorithme NLPQL est privilégié, suivi de près par le GRG. Le SQP a certes un bon score mais est sujet à des crashs (l’algorithme peut rentrer dans une boucle infini puis planter) : il n’est donc pas retenu. En revanche, si le temps de calcul n’était pas pris en compte, un algorithme génétique serait alors privilégié, le SAE étant un algorithme génétique modifié plus rapide ou une méthode hybride telle que l’algorithme Adaptive Region.

Cependant, en faisant le test statistique de Student sur l’ensemble des données optimisées par ces algorithmes, la plupart des modèles proposés pour B3 sont erronés. Comme il a été dit plus haut, les valeurs de B3 n’ont plus été prises en compte et la performance des algorithmes (tenant compte de B3 ou non) a été calculée puis comparée. Voici les résultats observés :

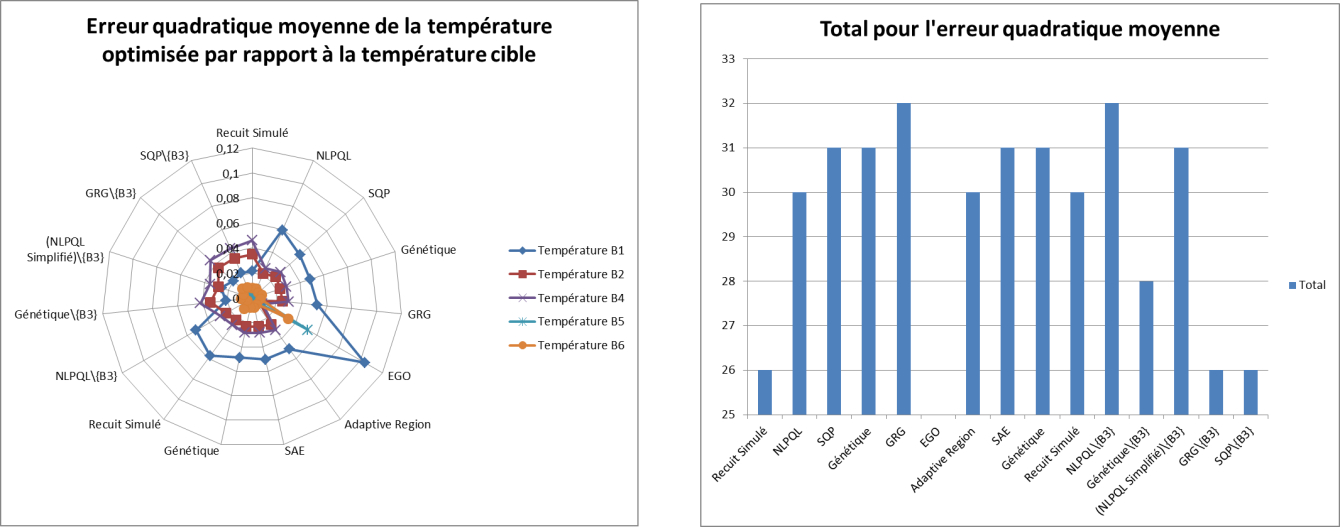


Figure : Nouveaux scores des différents algorithmes pour l'Erreur Quadratique Moyenne

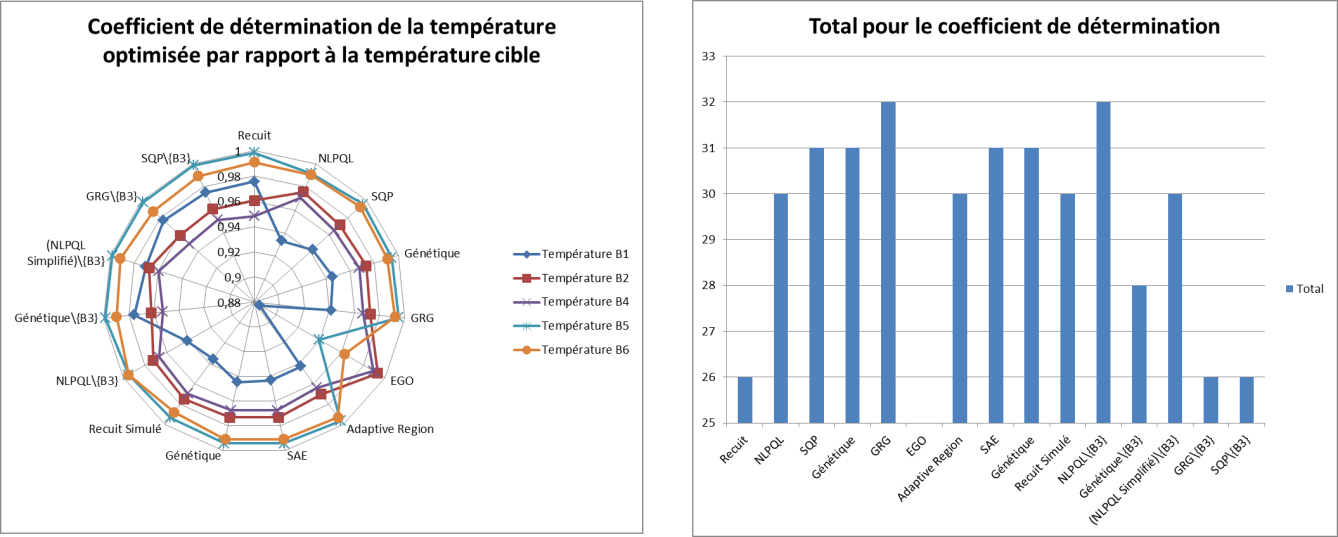


Figure : Nouveaux scores des différents algorithmes pour le Coefficient de Détermination

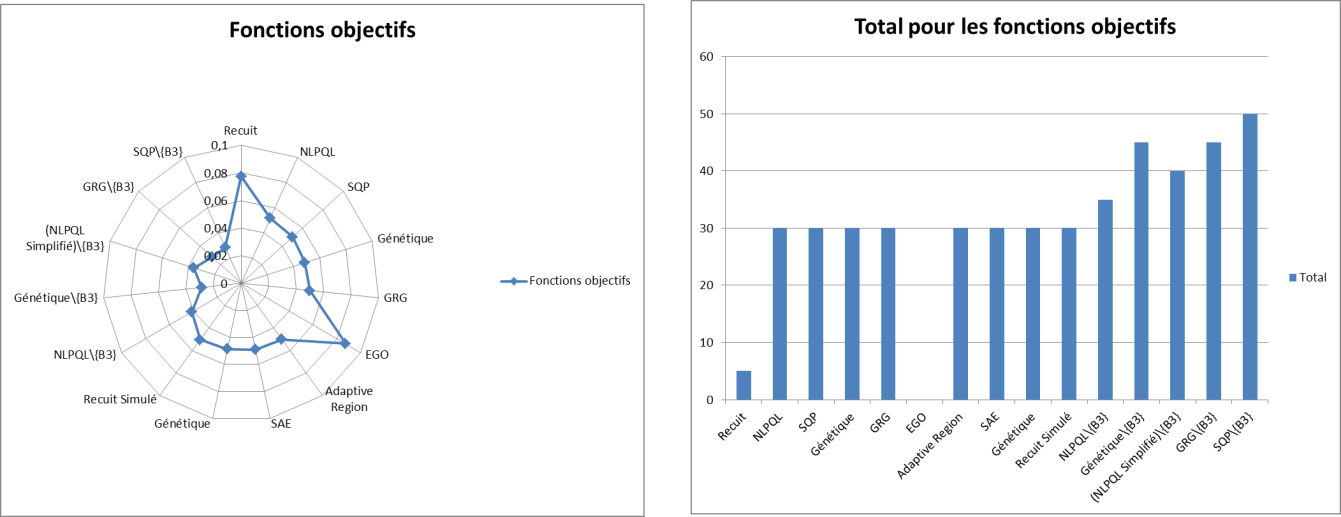


Figure : Nouveaux scores des différents algorithmes pour la fonction objectif

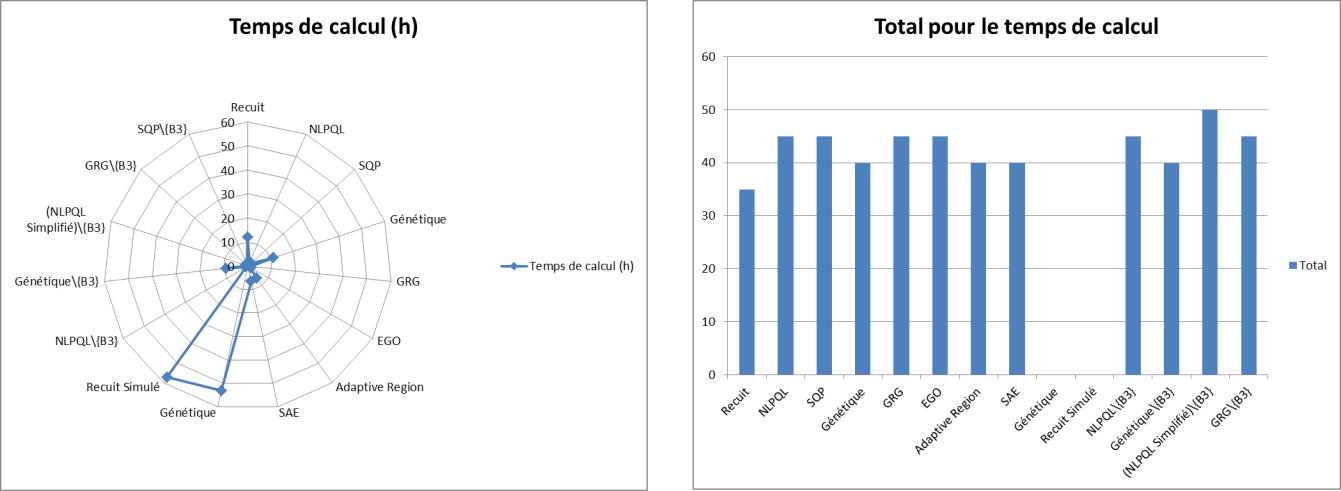


Figure : Nouveaux scores des différents algorithmes pour le temps de calcul

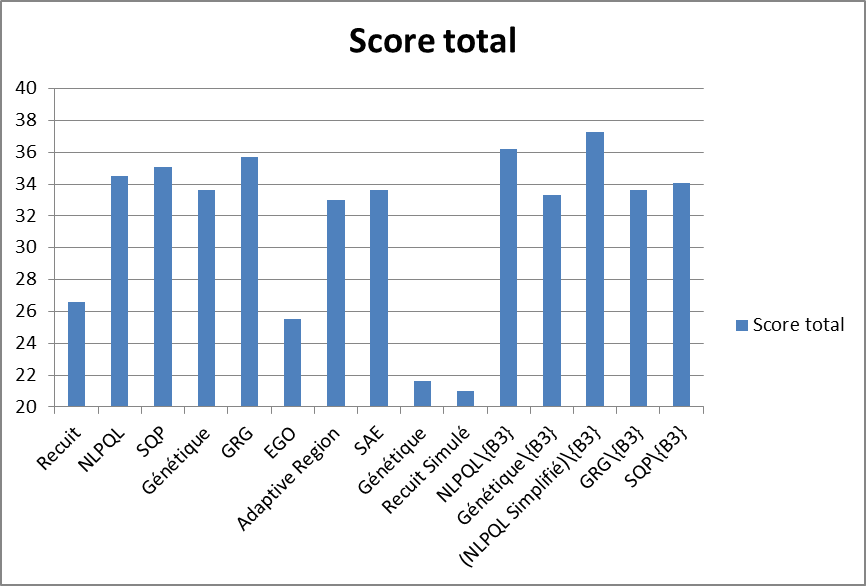


Figure : Nouveaux scores totaux pour les algorithmes utilisés

Cet histogramme de score montre que l’algorithme NLPQL est aussi précis que le GRG tout en ayant supprimé les valeurs aberrantes. Sa fonction objectif est plus faible et le calcule dure moins d’une heure. Globalement, il faut tout d’abord privilégier cet algorithme pour démarrer une optimisation. A l’aide de tests statistiques, le concepteur doit s’assurer de la validité des modèles. Si les tests ne sont pas bons, un algorithme stochastique peut être envisagé. En revanche, cette étude montre que les méthodes stochastiques ne sont pas les plus précises dans ce cas. Des tests complémentaires portés sur l’analyse des résidus peuvent également aider à départager deux méthodes en apparence similaires. De plus, il est intéressant de voir qu’un modèle simplifié (seuls les facteurs prépondérants, déterminés par un plan d’expérience, sont pris en compte) reste performant par rapport aux autres. De plus, le temps d’optimisation est raccourci ce qui en fait la méthode la plus performante de ce classement. Cependant, il faut prendre en compte le temps de calcul nécessaire pour le plan d’expérience ce qui n’a pas été le cas dans ce classement. A partir de ces résultats, il est possible d’établir une nouvelle méthode d’optimisation paramétrique présentée ci-après et améliorée par rapport à la méthode présentée précédemment :

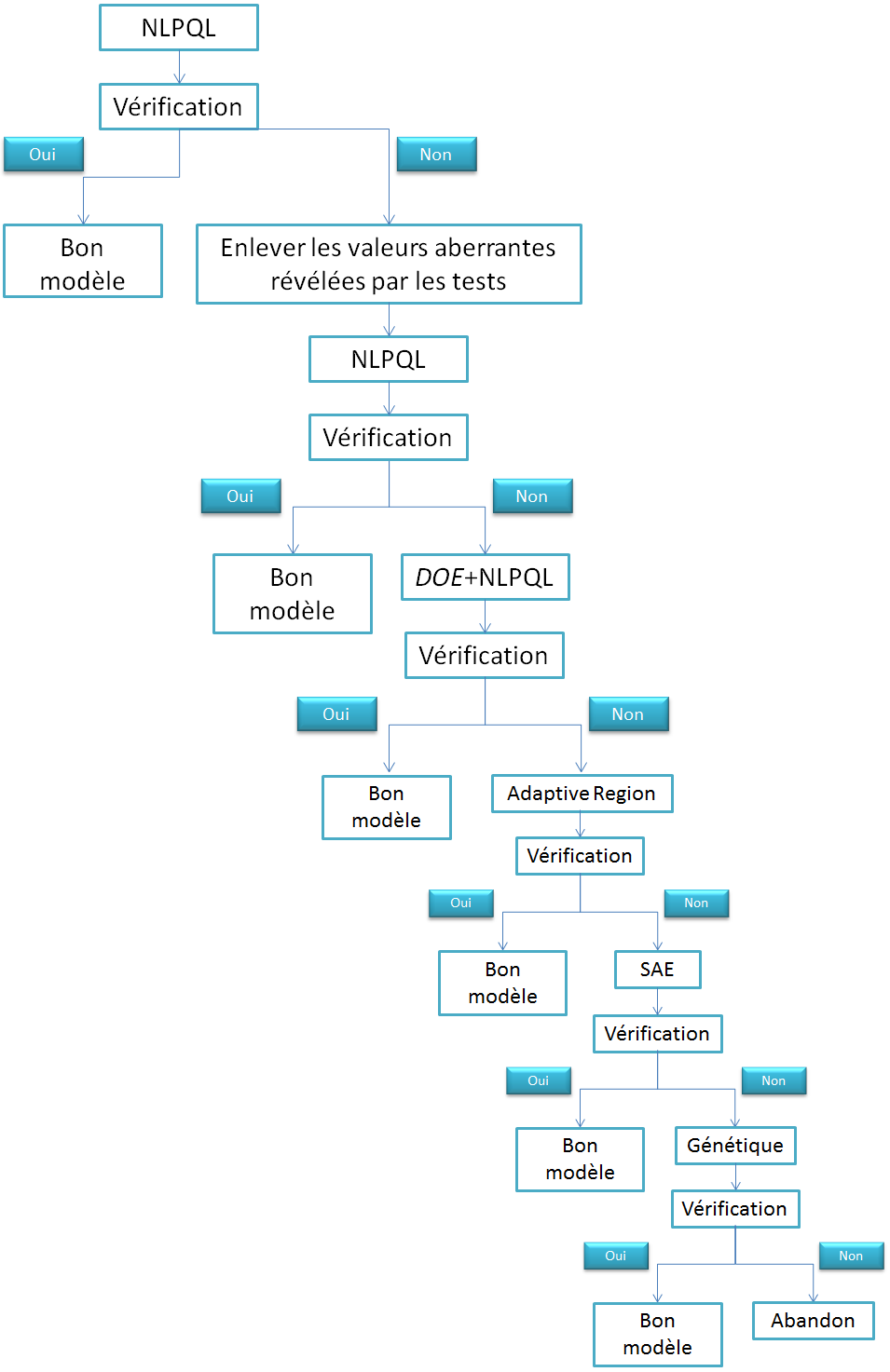


Figure : Méthode d'optimisation finale

# Optimisation paramétrique/topologique d’un bouchon de lessive

## Contexte

Dans le cadre de la réalisation d’un bouchon doseur fixé à une bouteille de lessive, le but est d’optimiser la géométrie afin de respecter les performances fixées par le cahier des charges. Les propriétés du fluide étant fixées, il faut ajuster des épaisseurs de zone de bouchon, des longueurs pour respecter les objectifs suivants :

* Un temps de dosage à minimiser (inférieur à 4s)
* Un volume de dose à respecter (supérieur à 25mL)

Le bouchon fonctionne à la manière d’un siphon. Trois trous sur la paroi permettent à la lessive de pénétrer dans le bouchon par différence de pression entre la bouteille et l’air extérieur. Le remplissage se termine lorsque la pression au bas de la bouteille est à la pression atmosphérique. Puis, la lessive coule en dehors du bouchon et ce dernier se vide par effet siphon. La phase de vidange s’arrête lorsque l’effet siphon est rompu.



Figure : Géométrie du bouchon de lessive

Des tests sont actuellement en cours pour valider la première géométrie proposée par Ingeliance. Cependant, aucune optimisation n’a été pratiquée sous ANSYS Workbench/Fluent. Le but est tout d’abord de voir s’il est possible de faire de l’optimisation avec ce logiciel et dans quelle mesure puis de l’appliquer à la géométrie du bouchon.

## Traitement d’un cas test : conduite coudée

Pour faire une optimisation test, un cas simple a été choisi. A l’entrée d’une conduite coudée, de l’eau s’écoule à la vitesse de 1.5m/s. La géométrie de la conduite va être optimisée afin de minimiser la pression d’entrée pour minimiser les pertes de charge. Les variables de conception sont le rayon de la conduite et le rayon de courbure du coude.

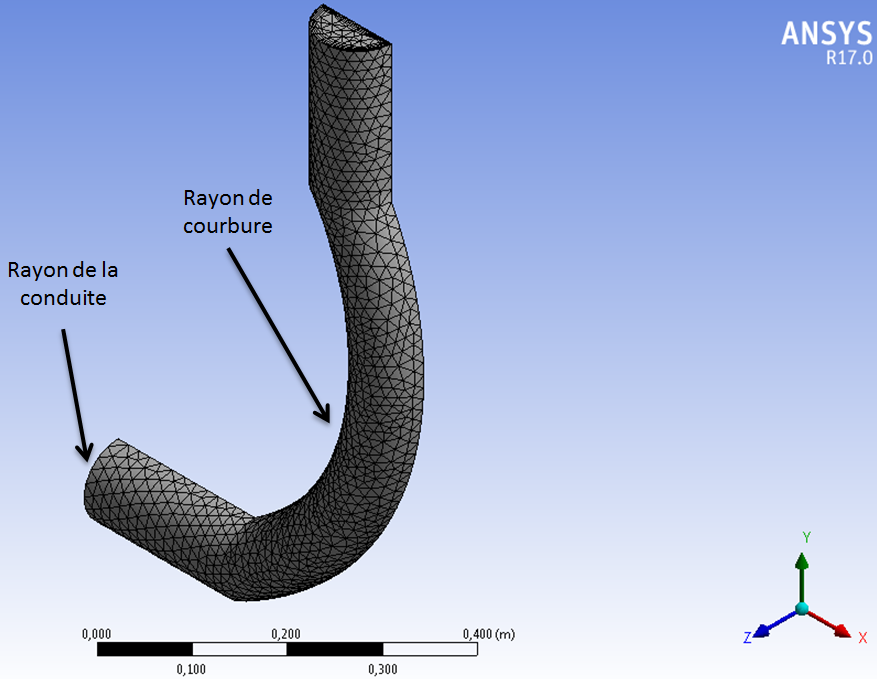


Figure : Géométrie de la conduite coudée réalisée sous ANSYS

Afin de limiter les temps de calcul et d’obtenir une géométrie satisfaisante, les variables de conception varieront entre 1 et 10 pouces. Après avoir vérifié que le calcul tourne bien sous Fluent, il est possible de mettre en place sous Workbench une routine d’optimisation décomposée en 3 étapes :

1. Génération d’un plan d’expérience qui balaye l’espace de solutions
2. Création d’une surface de réponse à partir des points calculés du plan d’expérience
3. Optimisation sur cette surface de réponse : recherche d’un min, d’un max, d’une cible, avec ou sans contrainte(s)

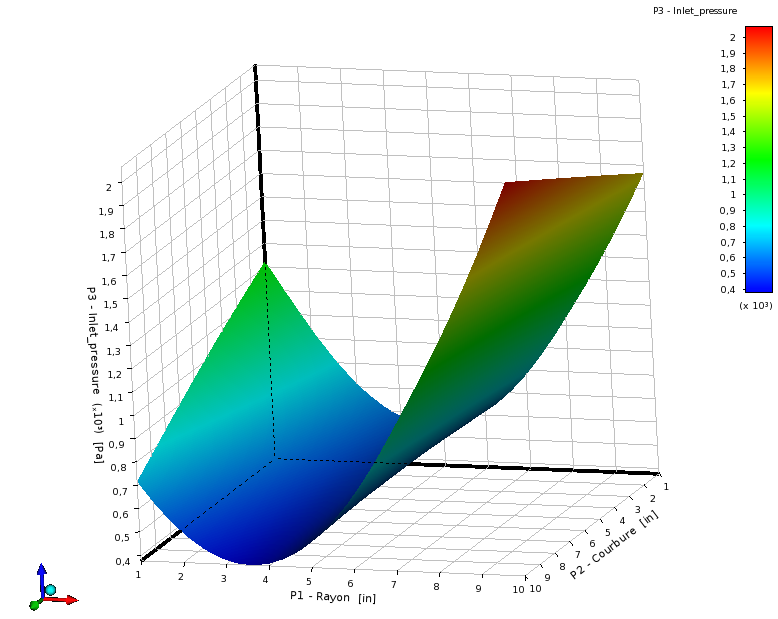


Figure : Surface de réponse générée lors de l'optimisation de la conduite coudée

Un algorithme de descente classique type NLPQL permet de déterminer le minimum de cette surface de réponse. Avec un rayon de conduite de 3.58in et une courbure de 10in, la pression d’entrée a diminué de 33.58%. Les résultats de pression et de vitesse sont comparés ci-dessous :

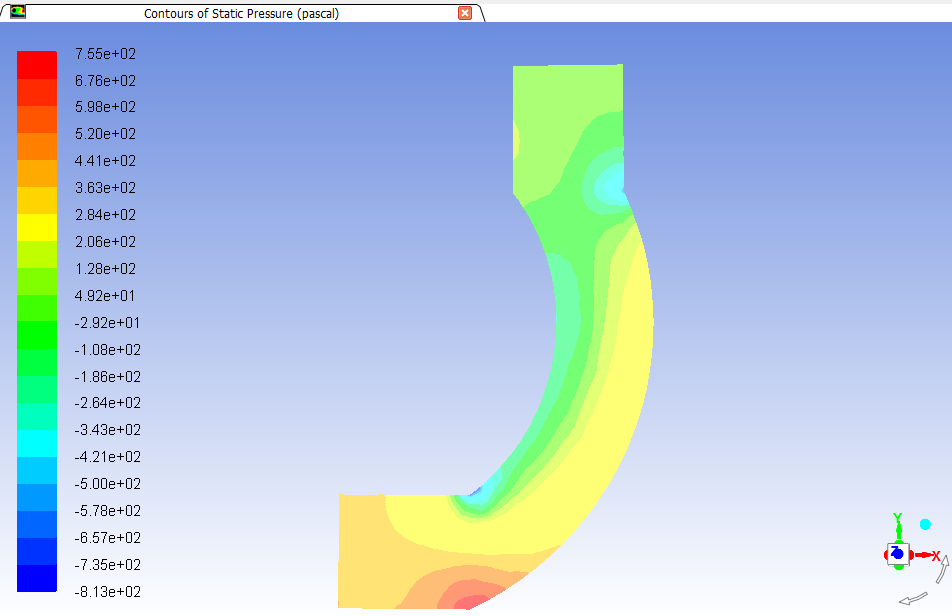
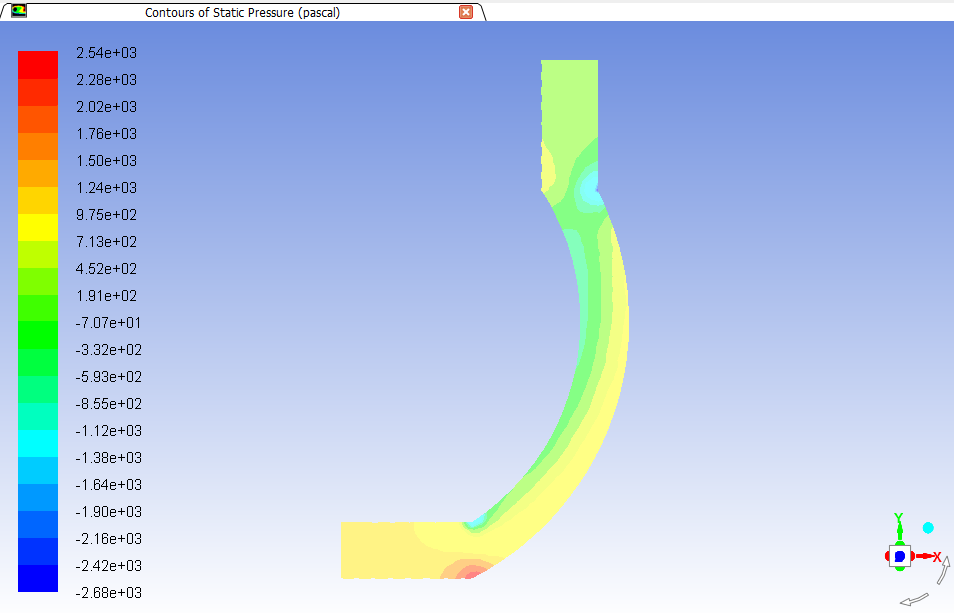
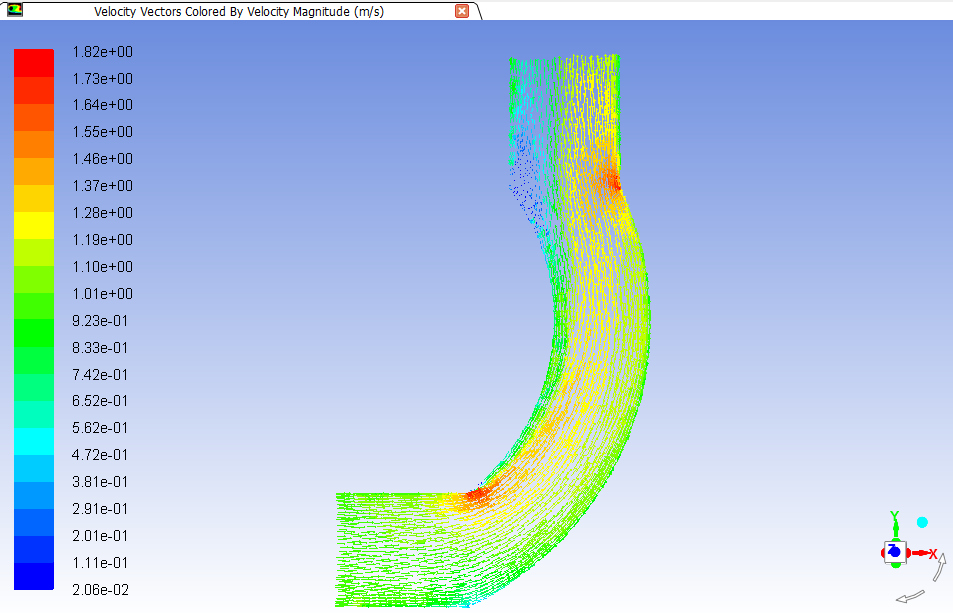
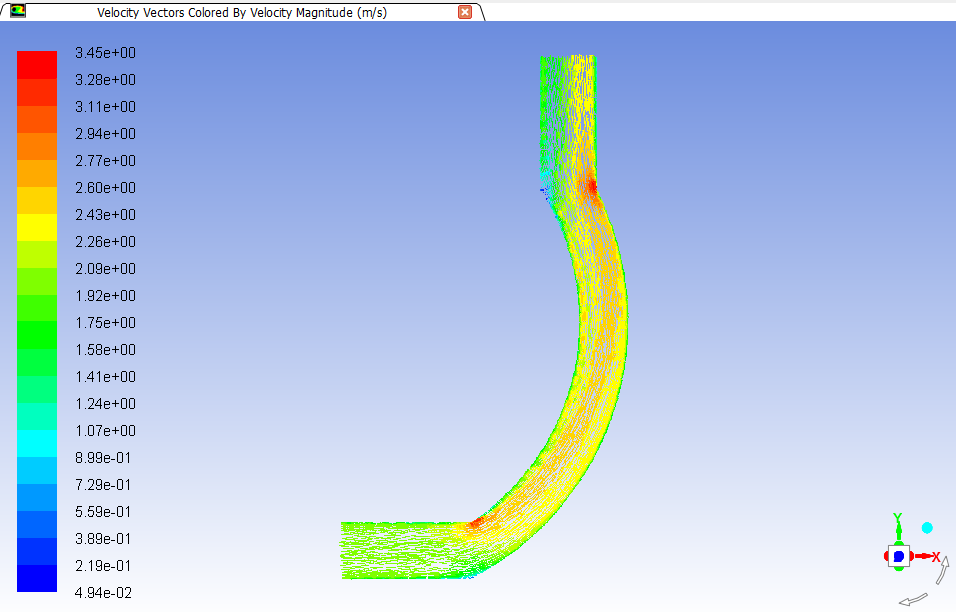


Figure : Evolution de la géométrie du coude avant et après optimisation

En haut : coloré selon la vitesse (m/s), max : 3.45 à g, 1.8 à d

En bas : coloré selon la pression (Pa), max : 2540 à g, 755 à d

La routine d’’optimisation fonctionne sous Workbench, il n’y a plus qu’à l’appliquer au bouchon de lessive. Cependant, l’optimisation ne fonctionne que lorsque l’écoulement est stationnaire. En effet, même lorsque le modèle est instationnaire, l’optimisation est incapable de définir pour quelle vitesse d’écoulement l’objectif est au mieux rempli. Par défaut, il utilise à chaque itération la dernière vitesse calculée. Par la suite, toutes les applications seront stationnaires.

## Application au bouchon de lessive

La géométrie du bouchon étant symétrique, il est possible de simplifier le problème selon son axe. Ainsi, la configuration suivante est générée sous Design Modeler :

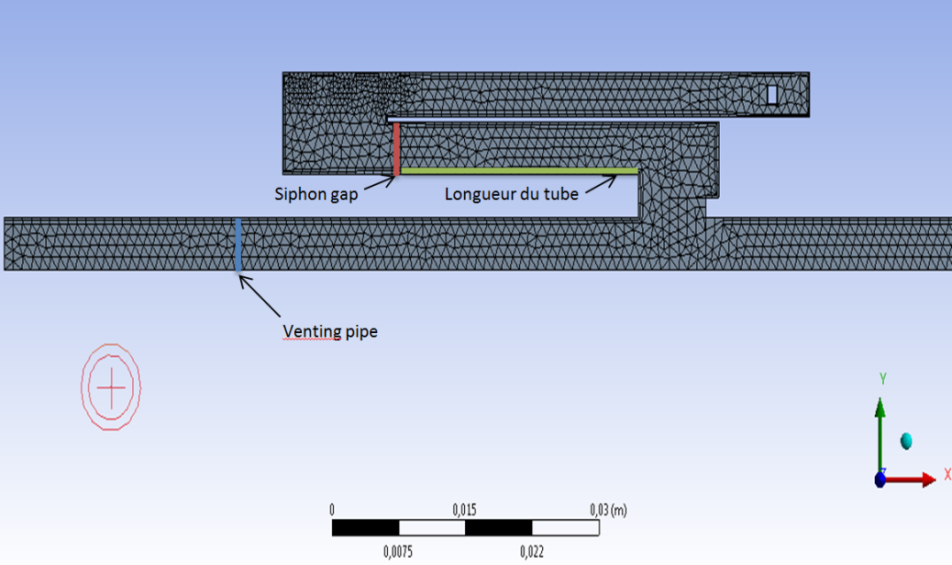


Figure : Visualisation sur la géométrie du bouchon des paramètres d'entrée de l'optimisation

En fonctionnement normal, il existe deux phases dans le bouchon : une phase liquide qui correspond à la lessive et une phase gazeuse qui correspond à l’air présent dans le bouchon pour créer l’effet siphon. Afin de simplifier le problème, un écoulement exclusivement composé de lessive sera considéré pour procéder à l’optimisation. L’optimisation consiste donc à améliorer les performances de l’écoulement de lessive provoqué par une différence de pression imposée entre l’entrée de la lessive et la sortie du bouchon. Les variables de conception sont le siphon gap, le venting pipe et la longueur du tube. Les deux premières variables sont liées : augmenter l’une va faire diminuer l’autre car le bouchon a une limite de largeur. En revanche, la longueur du tube influe sur le débit en sortie du bouchon et est indépendante des deux premières. Même si une optimisation avec comme paramètres d’entrée le siphon gap et le venting pipe est envisageable, il paraît plus judicieux de faire une optimisation avec des paramètres indépendants en entrée. La même routine d’optimisation décrite ci-dessus est utilisée pour le bouchon de lessive.

### Simplification de modèle

L’effet siphon est un phénomène assez complexe à modéliser. Le modèle initial, certes complet, présentait le désavantage de ne pas être adapté à l’optimisation du fait de son temps de convergence trop long. Il a donc fallu faire des choix afin de, dans un premier temps, réduire le temps de calcul pour une itération et d’adapter le calcul aux logiciels d’optimisation, à savoir Workbench et Python. Pour simplifier, seul le bouchon a été modélisé, sans la bouteille de lessive. Les différents modèles utilisés sont présentés ci-après, dans l’ordre chronologique :

* Modèle monophasique, stationnaire, paramètres Siphon Gap et Venting Pipe, prise d’air fermée
* Modèle monophasique, stationnaire, paramètres Siphon Gap et Longueur Tube, prise d’air fermée
* Modèle monophasique, stationnaire, paramètres Siphon Gap et Longueur Tube, prise d’air ouverte
* Modèle diphasique Volume of Fluid, stationnaire, paramètres Siphon Gap et Longueur Tube, prise d’air ouverte
* Modèle diphasique Volume of Fluid, stationnaire, paramètres Siphon Gap et Longueur Tube, prise d’air fermée
* Modèle diphasique Volume of Fluid, instationnaire, paramètres Siphon Gap et Longueur Tube, prise d’air ouverte

|  |  |  |  |
| --- | --- | --- | --- |
| **Monophasique** | **Stationnaire** | **Paramètres** | **Prise d'air** |
| Oui | Oui | SG+VP | Fermée |
| Oui | Oui | SG+Longueur Tube | Fermée |
| Oui | Oui | SG+Longueur Tube | Ouverte |
| Non | Oui | SG+Longueur Tube | Ouverte |
| Non | Oui | SG+Longueur Tube | Fermée |
| Non | Non | SG+Longueur Tube | Ouverte |

### Modèles monophasiques

Pour lancer une première boucle dont les paramètres d’entrée seront le siphon gap et le venting pipe, l’objectif sera de viser une valeur de débit volumique en sortie du bouchon : 25mL en 3s. Après un plan d’expérience, la surface de réponse est générée :

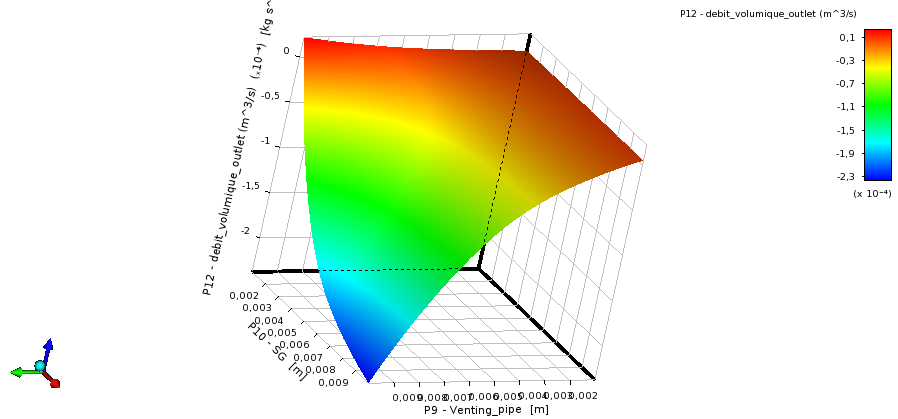
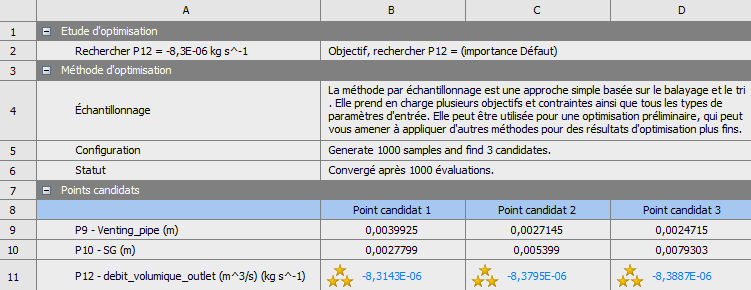


Figure : Surface de réponse générée après une première optimisation sur le bouchon de lessive

Un premier algorithme d’échantillonnage permet de balayer l’ensemble des solutions puis de raffiner les bornes des paramètres d’entrées pour obtenir des résultats plus précis lors d’une optimisation ultérieure. Les trois meilleurs résultats calculés par l’algorithme sont affichés.



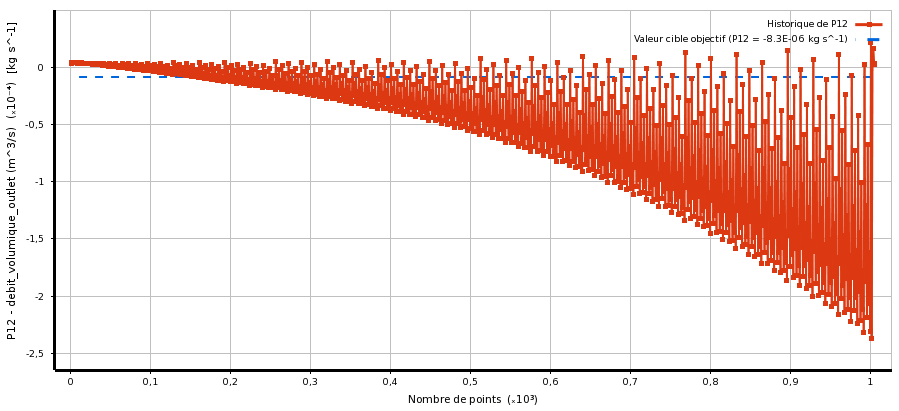
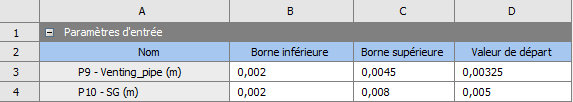


Figure : Méthode d'échantillonnage pour rechercher une valeur cible

Une fois les bornes raffinées, un algorithme de descente classique type NLPQL permet de rapidement converger vers une solution dont l’objectif est parfaitement rempli :



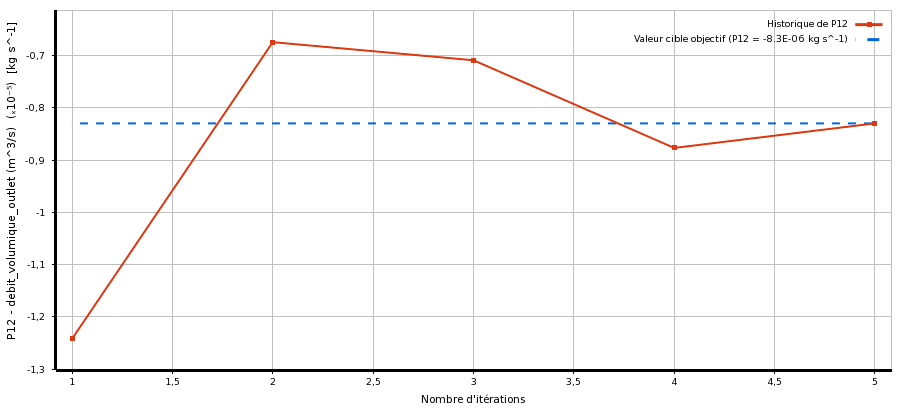


Figure : Convergence de la solution vers la valeur cible



En apparence, cette optimisation fonctionne. Cependant, il faudrait rajouter d’autres objectifs sur le volume de liquide présent dans le tube par exemple pour que les surfaces de réponses s’interceptent et diminuent l’espace de solution. De plus, il faudrait des paramètres d’entrée indépendants.

Les paramètres d’entrée sont le siphon gap et la longueur du tube, deux paramètres indépendants. Un plan d’expérience à 60 points est lancé (en faisant attention aux bornes afin que la géométrie soit toujours réalisable) puis la surface de réponse est générée.

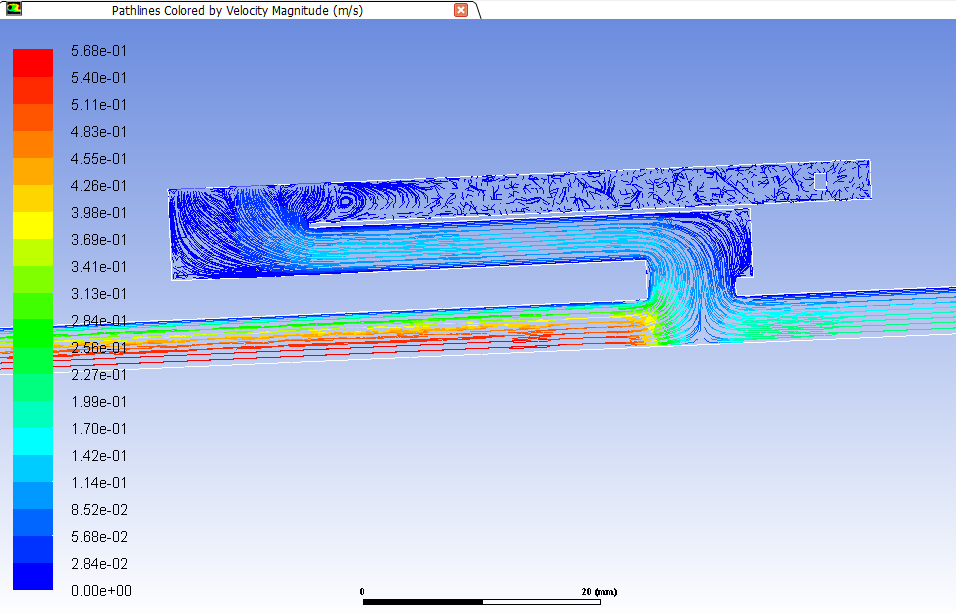


Figure : Lignes de courant dans le bouchon de lessive

La différence de pression entre les injecteurs de lessive et la sortie génère l’écoulement ci-dessus (figure 33). Ainsi, une première optimisation consistant à cibler le débit volumique en sortie ainsi que le volume de lessive présent dans le bouchon permet d’obtenir une première configuration : SG=0,0021m et L=0,032m. Bien que la longueur du tube change peu, le SG a diminué d’environ 50% par rapport à la configuration de départ. Cette différence s’explique par la différence de configuration des modèles : dans le modèle initial, il y avait une prise d’air au sommet du bouchon qui permettait d’augmenter l’effet siphon ; le siphon gap était plus large et l’objectif de débit volumique était respecté. Dans le modèle optimisé, la prise d’air a été remplacée par des parois. Afin de recaler les résultats par rapport au modèle initial, la prise d’air sera rouverte. De plus, l’ingénieur peut jouer sur les pressions d’entrée et de sortie afin de regarder l’évolution de la géométrie du bouchon.

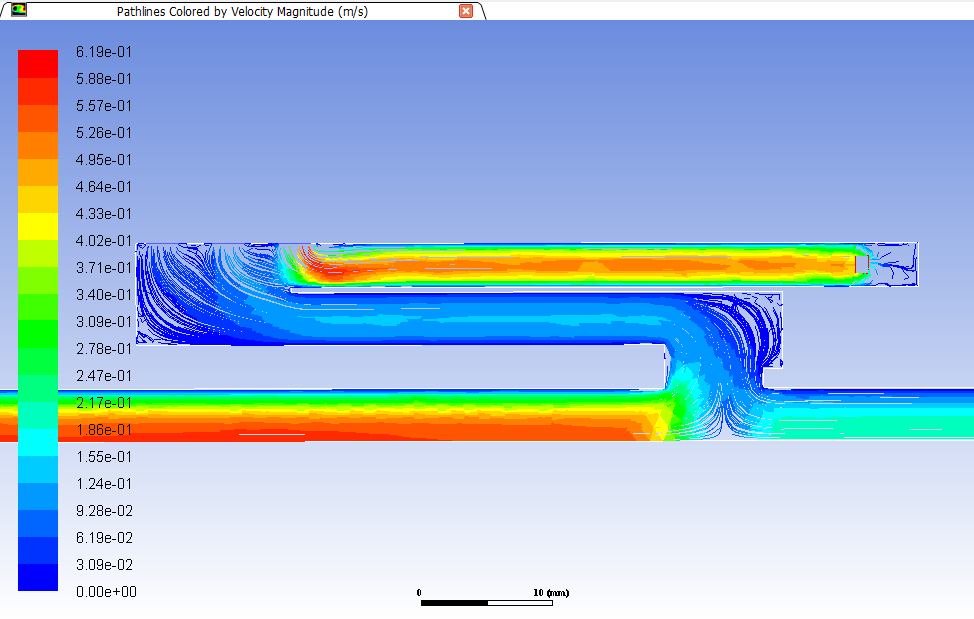


Figure : Lignes de courant dans le bouchon de lessive avec une condition sur la prise d'air

La différence de pression due à la prise d’air modifie radicalement l’écoulement : la quantité de lessive injectée par le troisième injecteur est quasiment totalement aspirée. Un objectif supplémentaire sera alors de minimiser le débit sortant par la prise d’air. Cependant, le débit de lessive sortant de la prise d’air est très important ce qui ne rend pas rentable la géométrie proposée. Il faut donc complètement changer le modèle pour mieux représenter l’effet siphon dans le bouchon.

### Modèles diphasiques

Une solution de modèle diphasique (lessive+air) est alors proposée. La différence de pression entre l’entrée et la sortie est toujours privilégiée. Pour éviter une recirculation au niveau de l’entrée de la lessive dans le bouchon causée par un gradient de pression hydrostatique, une UDF (user defined function) permet d’appliquer un profil de pression identique sur l’ensemble de l’inlet. Cependant, la prise d’air dans le bouchon a également dû être bouchée car une grande partie de la lessive dans le bouchon y était éjectée. Les résultats sont sensiblement les mêmes en optimisation qu’un modèle monophasique. La modélisation est toujours éloignée de la réalité.

La solution serait alors de rendre le modèle instationnaire pour appliquer une différence de pression entre l’entrée et la sortie qui varie temporellement. Il faut créer un script Python (disponible en annexe) qui puisse modifier les paramètres d’entrée d’un modèle instationnaire en pression, modifier la géométrie en conséquence, remailler, initialiser et lancer le calcul puis écrire les paramètres de sortie dans un fichier texte. Puis, avec la librairie d’optimisation disponible sur Python, il est possible d’en déduire la configuration optimale qui remplirait les objectifs.

Pour cela, le modèle a été mis à jour : il est désormais instationnaire et a été mis au point afin que le calcul ne prenne pas trop de temps pour chaque itération d’optimisation. Puis, à partir de Workbench, il est possible d’enregistrer les opérations exécutées sur la plateforme dans un journal. Les opérations effectuées sont les suivantes :

* Modifier les paramètres d’entrée du modèle : le siphon gap et la longueur du tube
* Mettre à jour la géométrie
* Mettre à jour le maillage
* Initialiser le calcul et répartir les phases (air/lessive) dans le bouchon
* Lancer le calcul
* Une fois le calcul terminé, sauver la configuration actuelle et fermer la fenêtre Workbench

Un tel journal peut être lancé par un code python qui fonctionne de la manière suivante :

* Une fonction de type write écrit un fichier d’entrée à partir des paramètres à optimiser. Le fichier d'entrée est écrit à partir d'un fichier initial "template" non modifié dont les valeurs des paramètres à optimiser sont des chaînes de caractères qu'on remplace par les valeurs numériques issues du calcul d'optimisation.
* Une fonction de type read lit les fichiers résultats générés par le calcul et extrait les valeurs souhaitées dans des dictionnaires
* Une fonction de type launch permet de lancer l’exécutable de Workbench en batch et de lire le journal décrit précédemment
* Une fonction de type update permet de mettre à jour les paramètres d’entrée du modèle en fonction des résultats de l’optimisation précédente
* Une fonction de type optimize met à jour les fichiers d’entrée, lit les résultats de la simulation et calcule l'écart quadratique avec les différents paramètres à optimiser. Elle crée et met à jour un fichier de suivi des différentes valeurs.
* Une commande qui spécifie l’algorithme d’optimisation utilisé

Une optimisation avec un algorithme de descente NLPQL a été lancée et, au bout de 40 heures et 87 itérations, les résultats suivants sont observés :

Un minimum local a effectivement été atteint au bout de la 47ème itération. Les objectifs ont été quasiment remplis puisque le bouchon délivre alors 24mL de lessive en 2.2s. Cependant, pour la configuration optimisée, le Siphon gap a augmenté de 50% par rapport à sa valeur nominale et la longueur du tube diminuée de 93%. Ces résultats sont donc à relativiser par rapport à la réalité puisque là aussi, des simplifications de modèle ont été opérées (modèle 2D axisymétrique, maillage plus grossier pour diminuer le temps de calcul, tension de surface pour limiter la diffusion, absence d’angles de contact entre la lessive et la paroi).

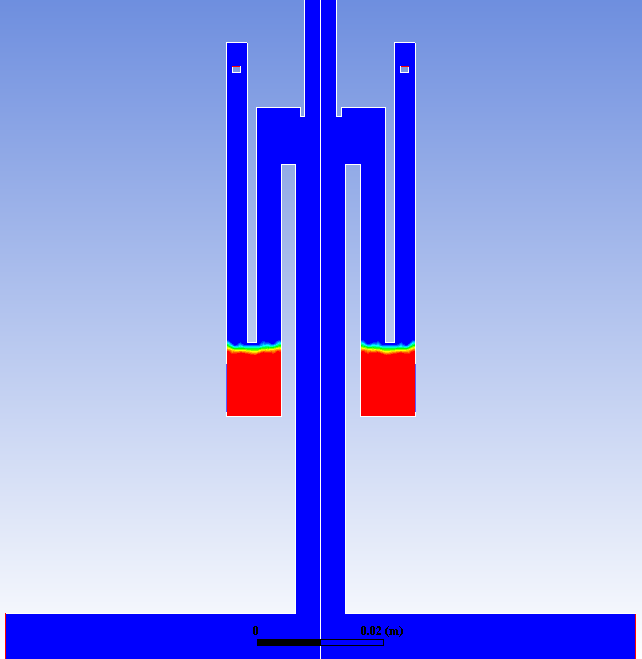
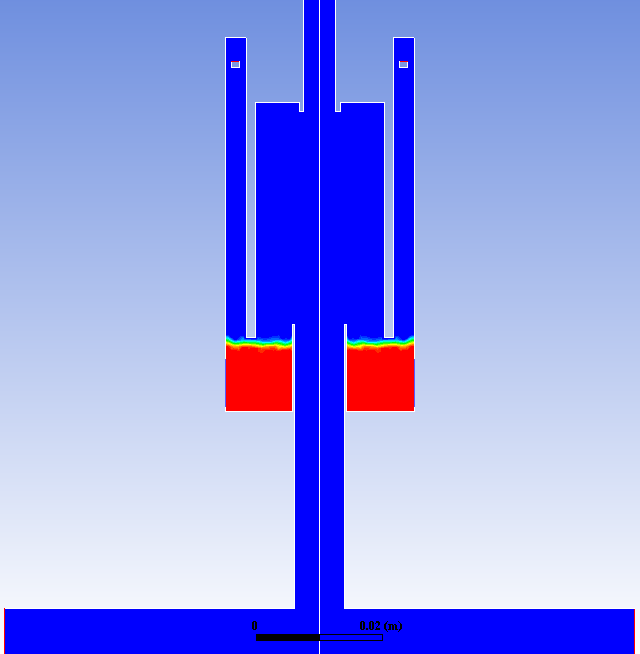


Figure : Configuration avant-après optimisation du bouchon

Cependant, aux vues des points calculés au cours de l’optimisation et de la valeur de la fonction objectif pour chacun d’eux, il est possible de dire qu’une tendance se dégage de cette optimisation :

* Augmentation du Siphon Gap
* Diminution de la longueur du tube

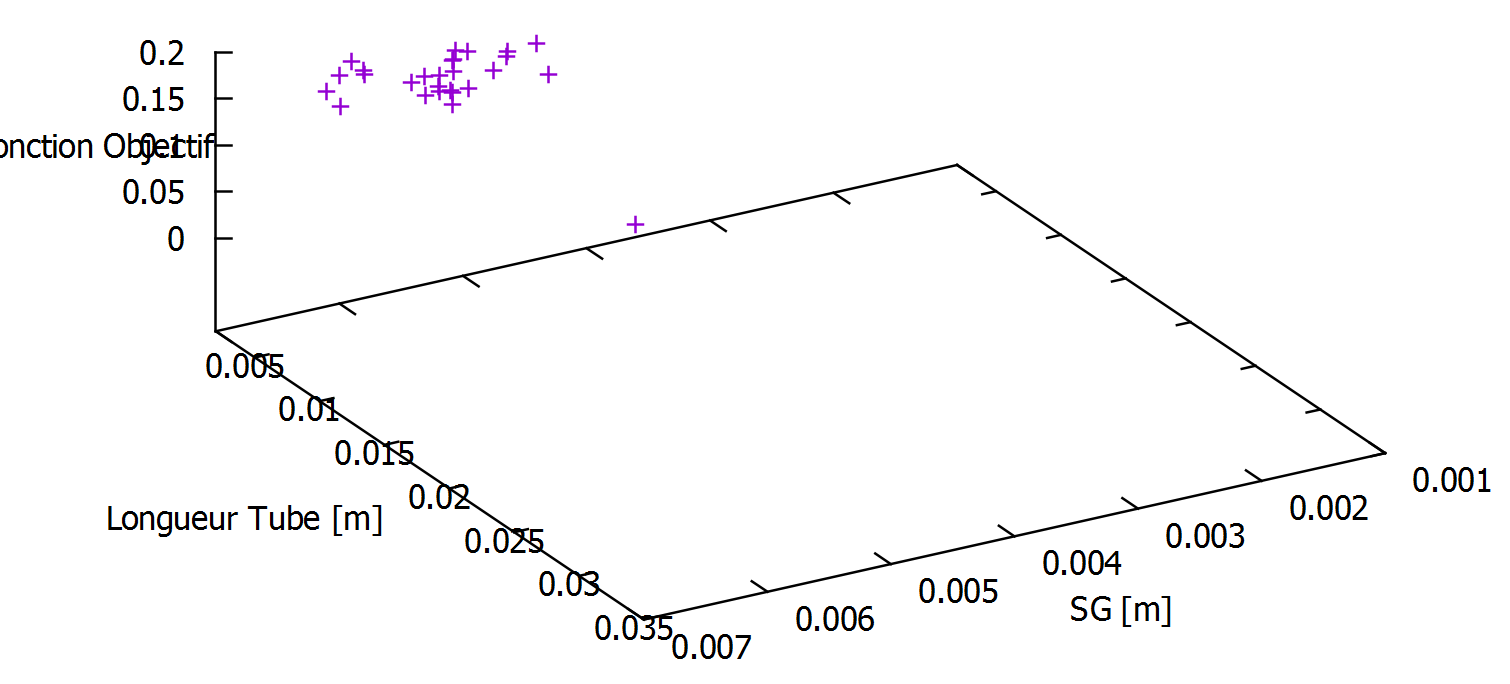


Figure : Valeurs les plus basses de la fonction objectif en fonction du SG et de la longueur du tube

La mise en place d’une boucle d’optimisation Python couplée avec un logiciel de calcul fournit des résultats assez satisfaisants. Cette boucle s’adapte à de nombreux logiciels (ANSYS, NX/Marc, Nastran…), fonctionne pour tout type de calcul et Python dispose d’une très grande bibliothèque d’algorithmes d’optimisation, dont ceux qui ont été identifiés comme les plus performants lors de la première partie. Cependant, le plus grand problème, inhérent à n’importe quelle boucle d’optimisation, reste le temps. Cette méthode demandant un certain nombre d’itérations, l’utilisateur doit parfois recourir à des simplifications de modèle inévitables pour réduire le temps de calcul.

# Conclusion

A travers ce projet de fin d’études réalisé chez Ingeliance Technologies, une partie des méthodes d’optimisation paramétrique en simulation numérique a pu être étudiée. Des recherches bibliographiques ont mis en évidence des algorithmes d’optimisation efficaces pour la résolution de problèmes variés. Après avoir démontré les avantages et les inconvénients de chaque algorithme, un premier cas industriel sur un calcul thermomécanique a permis d’établir une classification de ces algorithmes selon leurs performances. Ainsi, il a été mis en évidence que l’algorithme de descente NLPQL était non seulement un des plus utilisés mais aussi un des plus efficaces en raison de la prise en compte de la vitesse de descente de la fonction objectif pour adapter son pas. De plus, l’algorithme génétique a démontré ses capacités et pourra être utilisé si la configuration initiale est éloignée de la solution afin d’avoir plus de chances de trouver un minimum global. De plus, le post traitement des données par un rapide test statistique a prouvé son utilité afin de vérifier la validité de certaines valeurs de sortie. Pour terminer sur ce cas, une méthode générale d’optimisation a été proposée, consistant à lancer en premier lieu un algorithme de descente (rapide et précis si la solution est proche d’un minimum local) puis de post traiter les données par un test statistique puis d’éventuellement faire appel à un plan d’expérience pour visualiser la forme de la surface de réponse. En dernier recours, il est préconisé de faire appel à un algorithme génétique, certes lent, mais qui donne de meilleures chances de résultat.

Le second cas traitait de l’optimisation paramétrique des performances d’un bouchon de lessive fonctionnant par effet siphon. Une première tentative consista à simplifier le modèle initial afin de le rendre « optimisable » en raison de la lenteur de convergence. Un premier modèle stationnaire a pu être optimisé sous Workbench mais cet outil a rapidement montré ses limites (impossibilité de résoudre des cas instationnaires diphasiques). Pour résoudre de tels cas, il a fallu utiliser un code d’optimisation Python qui permettait de piloter Workbench qui lui-même contrôlait les modifications de géométrie, le maillage, l’initialisation des phases et le lancement du calcul dans DesignModeler, Meshing et Fluent. Après visualisation des résultats, ce code Python a pu être validé et a montré sa polyvalence puisqu’il peut s’adapter à de nombreux autres logiciels.

En conclusion, l’optimisation constitue un axe d’étude important pour Ingeliance dont il reste encore beaucoup à découvrir. Cependant, l’entreprise dispose de plusieurs outils performants s’appliquant à de nombreux problèmes. Ce stage aura été bénéfique dans tous ses aspects, que ce soit d’un point de vue de l’apprentissage technique ou de l’interaction humaine avec mon tuteur et les autres membres de l’équipe. J’ai le souhait de pouvoir retrouver cette atmosphère de travail et des projets orientés aéronautiques couplés à l’optimisation au cours de ma future carrière.

# Références

**A.A.Cournot** texte sur l'élasticité, proposé en ligne et commenté sur le site BibNum (1838)

**A.Collignan** Thèse de doctorat « Méthode d’optimisation et d’aide à la décision en conception mécanique » (2011)

**K.Saridakis** Article « Soft computing in engineering design – A review » (2008)

**R.Amen** Article “Case-based reasoning as a tool for materials selection” (2001)

**J.V.Neumann** Ouvrage “Theory of games and economic behavior” (1944)

**D.Jones** Article “Multi-objective meta-heuristics: an overview of the current state-of-the-art” (2002)

**A.Billionnet** Ouvrage “Optimisation discrète : De la modélisation à la résolution par des logiciels de programmation informatique” (2006)

**J.Nash** Article « Non-cooperative games » (1951)

**Y.Y.Haimes** Article « On a bicriterion formulation of the problems of integrated system identification and system optimization » (1971)

**L.LeBel** Présentation “Prise de decision Multi-critères, Université de Laval (2009)

**T.L.Saaty** Ouvrage « The analytic hierarchy process: planning, priority setting, resource allocation » (1980)

**L.A.Zadeh** Ouvrage “Fuzzy sets” (1964)

**E.C.Harrington** Article « The desirability function » (1965)

**R.Derringer** Article “Simultaneous optimization of several response variables” (1980)

**G.Cabodevilla** Article “Identification des systèmes”, Ecole Nationale Supérieure de Mécanique et des Microtechniques (????)

**M.Bergmann** Article « Algorithmes d’optimisation non linéaires sans contraintes », Université de Bordeaux (2012)

**X.Antoine, P.Dreyfuss, Y.Privat** Article « Introduction à l’optimisation : aspects théoriques, numériques et algorithmes » (2006-2007)

**R.Storn & K.Price** Article « Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces » (1997)

**J.Lepagnot** Thèse de doctorat “Conception de métaheuristiques pour l’optimisation dynamique. Application à l’analyse de séquences d’images IRM » (2011)

**Y.Vernat** Thèse de doctorat « Formalisation et qualification de modèles par contraintes en conception préliminaire » (2004)

**Y.Ledoux** Thèse de doctorat « Optimisation des procédés d’emboutissage par caractérisation géométrique et essais numériques » (2005)

**A.Moles** Ouvrage « Théorie des actes » (1993)

**R.G.D.Allen** Article « The concept of arc-elasticity of demand » (1934)

**A.Rondepierre, P.Weiss** Article “Méthodes standards en optimisation non linéaire déterministe” INSA Toulouse (2012-2013)

**Y.Colette, P.Siarry** Ouvrage « Optimisation multiobjectif » (2002)

**A.Tahan** Ouvrage « Analyse expérimentale pour ingénieurs » (2016)

**J.L. Dauvergne** Thèse de doctorat «Réduction et inversion de modèles de conduction thermique avec changement de phase » (2008)

**M.Bergman** Thèse de doctorat « Algorithmes d’optimisation non-linéaire sans contraintes » (2010)

# Annexes

## Formalisation des préférences sur les variables d’observation

Dans ce paragraphe, la formalisation des préférences du concepteur, déjà évoquées précédemment, sera étudiée.

### Le paradoxe de St Pétersbourg

Proposé par Nicolas Bernoulli au XVIIIème siècle, ce problème met en valeur le paradoxe du comportement humain face aux événements d’une variable aléatoire dont la probabilité est petite mais dont l’espérance est infinie. Par exemple, pour un joueur qui est face à la banque au casino par exemple, le risque auquel le joueur est confronté implique que son choix de jouer ou non n’est pas totalement rationnel.

Du paradoxe de Saint Pétersbourg découle l’idée de formaliser la valeur subjective que prend une variable d’observation pour le concepteur.

### Les fonctions de satisfaction

Les fonctions de satisfaction permettent de formaliser les préférences du concepteur sur chacune des solutions candidates en associant à chaque VObs d’une solution une variable d’interprétation Vint correspondant au niveau de satisfaction. Il y a donc autant de Vint que de VObs. Les fonctions de satisfaction présentent trois avantages :

* Adimensionner les VObs
* Traduire les préférences du concepteur
* Simplifier le problème de décision

Les fonctions de satisfaction permettent ainsi de convertir des valeurs naturelles (masse, volume, coût...) en valeurs adimensionnées entre 0 et 1. Cela permet de rendre comparable des VObs de natures différentes.

### Les fonctions d’appartenance

Les fonctions d’appartenance reposent sur la notion de logique floue proposée par Zadeh [Zadeh, 1964] qui stipule que les affirmations ne sont pas vraies ou fausses mais aussi partiellement vraies ou presque fausses. Cette zone floue est représentée par une fonction d’appartenance à valeurs entre 0 et 1 où 0 correspond au faux et 1 au vrai. Ainsi, au lieu de raisonner en vrai/faux, il est possible d’appliquer le principe des fonctions d’appartenance pour refléter un niveau de satisfaction. Cependant, les fonctions d’appartenance peuvent présenter des problèmes de discontinuité.

### Les fonctions d’utilités

Les fonctions d’utilité sont utilisées pour associer des probabilités à des variables d’observation où 0 correspond à la moins satisfaisante et 1 à la plus satisfaisante. Par la suite, la méthode de la loterie est utilisée pour paramétrer la fonction d’utilité. Toutefois, comme les fonctions d’appartenance, les fonctions d’utilité peuvent présenter des discontinuités.

### Les fonctions de désirabilité

Les fonctions de désirabilité sont des fonctions de satisfaction faciles à paramétrer et sont adaptées aux besoins des concepteurs. Deux fonctions seront présentées ci-après.

#### Fonctions de Harrington

Harrington a proposé ses fonctions de désirabilité dans le domaine de la qualité [Harrington, 1965].

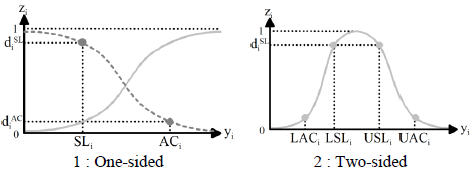


Figure : Fonctions de désirabilité de Harrington

Elles sont au nombre de trois. Les deux premières sont monotones et permettent de formaliser une préférence en terme d’augmentation ou de diminution : elles sont nommées One-sided. Elles sont chacune paramétrées par deux points dont les niveaux de satisfaction respectifs et  paramètrent la relaxation des préférences :

* Le Soft Limit (SL) qui indique la zone au-delà de laquelle les valeurs de la variable d’observation sont satisfaisantes
* Le Accurate Constraint (AC) qui indique la zone de non satisfaction

Les deux fonctions de désirabilité One-sided de Harrington sont paramétrées ainsi :

Avec, dans le cas d’une fonction One-sided décroissante :

Et dans le cas d’une fonction One-sided croissante :

La troisième fonction de Harrington, appelée Two-sided, permet le ciblage d’une valeur pour une variable d’observation, ou la spécification d’une zone de satisfaction pour plusieurs valeurs d’une VObs. Elle est paramétrée par 4 points :

* La Lower Soft Limit (LSL)
* La Upper Soft Limit (USL)
* La Lower Accurate Constraint (LAC)
* La Upper Accurate Constraint (UAC)

Qui obéissent aux contraintes suivantes :

La fonction Two-Sided de Harrington est calculée comme suit :

Ces fonctions de désirabilité présentent de nombreux avantages :

* Elles sont simples à comprendre
* Elles ne présentent pas de discontinuité
* Elles remplissent l’objectif de formalisation des préférences
* Elles sont adaptées aux buts recherchés en conception : augmentation, diminution ou ciblage

#### Fonctions de Derringer

En se basant sur les travaux de Harrington, Derringer [Derringer, 1980] propose ses fonctions de désirabilité appliquées à l’optimisation en conception. A la différence des fonctions de Harrington, les fonctions de désirabilité de Derringer présentent des discontinuités et sont définies par morceaux. En fonction du but recherché, Derringer propose deux types de fonction (tout comme Harrington) : le type One-sided en cas de maximisation/minimisation et le type Two-sided utilisé pour du ciblage.

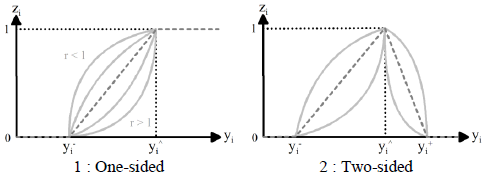


Figure : Fonctions de désirabilité de Derringer

La fonction One-sided dispose de trois paramètres :

* Le paramètre indique la limite au-dessous de laquelle les valeurs de la variable d’observation considérées sont inadmissibles pour le concepteur
* Le paramètre indique la limite au-dessus de laquelle les valeurs de la variable d’observation considérées sont satisfaisantes pour le concepteur
* Le paramètre r permet de jouer sur la courbure de la fonction

La fonction Two-sided est construite sur le même principe que la fonction One-sided, en la définissant par morceaux. Les paramètres de la fonction Two-sided sont les deux bornes et au-delà desquelles la désirabilité est nulle. La valeur cible est notée . Enfin, les paramètres r et t définissent les niveaux de courbures de la fonction, indépendants l’un de l’autre :

Les fonctions de Derringer sont faciles à comprendre et les paramètres de réglage des courbures permet de formaliser finement les préférences du concepteur. Cependant, ces fonctions présentent des discontinuités qui sont difficilement justifiables en conception de produit.

## Le système d’aide à la décision

Le système d’aide à la décision en conception est une méthode de modélisation des préférences du concepteur en vue de faciliter le mécanisme de décision au travers d’une fonction objectif. La variable de qualification VQa est la variable unique retournée par la fonction objectif en vue de qualifier un aspect du produit conçu. Cette variable est adimensionnée entre 0 et 1, où une valeur de 1 indique une solution candidate correspondant parfaitement aux préférences exigées par le concepteur. VQa doit être maximisée. La performance d’un produit est également exprimée sous la forme d’une variable, notée p et qualifie le comportement d’une solution en terme de satisfaction de ses objectifs de conception. Enfin, la confiance, notée c, qualifie la similarité entre une solution candidate X et une solution de référence notée X0.

### Identification des objectifs de conception

En conception, avant toute chose, il est nécessaire de procéder par une analyse fonctionnelle des besoins. Cette méthode permet d’identifier des actions visant à satisfaire les objectifs de conception.

### Etablissement des priorités

Une fois les actions déterminées, elles doivent être hiérarchisées et des priorités sont établies. Pour cela, le processus de hiérarchie analytique (AHP) décrit précédemment est utilisé.

### La notion d’arc-élasticité

En présence de deux VQa, le concepteur souhaite maximiser conjointement la confiance et la performance : une confiance élevée atteste d’une proximité avec une solution connue tandis qu’une performance élevée souligne la satisfaction des objectifs de conception. Ces variables n’étant pas comparables, il faut trouver une solution.

#### Compromis entre confiance et performance

Soit c0 et p0 les confiance et performance de la solution initiale X0 et c et p les confiance et performance de toute solution candidate X. Par définition, c0=1. Si une solution X présente une confiance maximale, il y a identité entre les solutions donc p=p0. En revanche, si une solution X est différente de X0, sa confiance est nécessairement inférieure à c0. Il apparaît évident que les concepteurs devront faire un compromis entre une amélioration de performance et une dégradation de confiance pour faire évoluer le produit vers une solution X. Ce compromis peut être rapproché du principe de «coût généralisé » de Moles [Moles, 1993] qui stipule qu’un passage à l’acte (de X0 vers X) n’est réalisé que si le bénéfice qui en est retiré est plus élevé que son coût. Dans ce cas-ci, le bénéfice est la variation de performance et le coût correspond à la diminution de confiance.

#### L’élasticité

Le concept d’élasticité fut introduit au XIXème siècle par Cournot [Cournot, 1838] pour modéliser mathématiquement l’offre et la demande. Soit Q la demande d’un produit (Quantity) et P son prix (Price). L’élasticité est définie comme le ratio de la variation ∂Q adimensionnée par la variation ∂P adimensionnée :

La valeur de varie et est sujette à interprétation :

* : une variation de P implique une variation opposée de Q. Une augmentation du prix d’un produit diminue ses ventes
* : Q reste fixe quand P varie. La consommation du produit est indépendante de son prix
* : une variation de P induit une variation de Q dans le même sens. Les ventes du produit augmentent quand son prix augmente (produits de luxe)
* : Q est dite inélastique. Une variation de P induit une faible variation de Q
* : Q est dite élastique. Une variation de P induit une grande variation de Q
* : Les variations adimensionnées de Q et P sont égales

La définition de l’élasticité suppose que la fonction reliant P et Q est connue et dérivable. Ce n’est pas toujours le cas. C’est la raison pour laquelle la notion d’arc-élasticité fut introduite.

#### L’arc-élasticité

Allen proposa la notion d’arc-élasticité au début du XXème siècle [Allen, 1934]. L’arc élasticité est calculée en deux points identifiés par leurs coordonnées respectives : (Q1; P1) et (Q2; P2). Elle est calculée comme suit :

Si ce principe est appliqué au problème rencontré :

* Les deux situations de Allen correspondent aux solutions X et X0
* La performance correspond à la demande Q
* La confiance s’apparente au prix P

Le but recherché étant une augmentation de la performance par rapport à X0 qui induisait une diminution de confiance, l’arc élasticité est négative. L’opposé de l’arc-élasticité, noté σ, comme indicateur à maximiser est préféré. Ainsi, σ s’exprime tel que :

L’arc élasticité de la fonction initiale X0 est fixée à 0 [Collignan, 2011]. Dès lors, deux cas d’interprétation se présentent :

* et  : La solution X propose une nette hausse de performance pour une variation de confiance faible. Les choix de conception initiaux doivent être révisés pour obtenir une solution bien plus performante.
* et  : La solution X propose une forte hausse de performance pour une grande perte de confiance. Dans ce cas, il s’agit d’un saut technologique : pour obtenir une solution vraiment meilleure, des risques doivent être pris pour proposer une solution innovante.

## Code Python pour l’optimisation

# coding: utf-8

# In[1]:

import subprocess

import sys

import os

from numpy import \*

from scipy import optimize

# # Définition des fonctions

# ## Ecriture des fichiers d'entrée

# `writeinput(fixdic\_,optidic\_,fichiertpl\_,fichierdat\_)`

#

# Cette fonction écrit le fichier d'entrée (qui fournit les paramètres à optimiser au programme tiers) `fichierdat\_` à partir de la liste de dictionnaires contenant les paramètres à optimiser `optidic\_` et les paramètres fixés `fixdic\_` (paramètres optimisables mais mis de côté pour le calcul. Le fichier d'entrée est écrit à partir d'un fichier initial "template" `fichiertpl\_` non modifié dont les valeurs des paramètres à optimiser sont des chaines de caractères qu'on remplace par les valeurs numériques issues du calcul d'optimisation.

# In[2]:

def writeinput(fixdic\_,optidic\_,fichiertpl\_,fichierdat\_):

with open(fichiertpl\_,'r') as fid:

lines=fid.readlines()

for i in range(len(lines)):

for dic in fixdic\_+optidic\_:

name=dic['name']

if name in lines[i]:

string=str(dic['value'])

lines[i]=lines[i].replace(name,string)

with open(fichierdat\_,'w') as fid:

for line in lines:

fid.write(line)

# ## Lecture des fichiers de résultat

# `readoutput(fichierpost\_)`

#

# Cette fonction lit les résultats du calcul du programme tiers écrites dans le fichier de résultats `fichierpost\_` et les range dans la liste de dictionnaires `resvals\_` retournée par la fonction. Le nom des entrées des dictionnaires de `resvals\_` doit correspondre aux noms des entrées des dictionnaires de `resdic`, qui contient les valeurs objectif.

#

# Cette fonction est à adapter selon les cas.

# In[3]:

def readoutput(fichierpost\_):

resvals\_={}

restemp = genfromtxt(fichierpost\_,skip\_header=2)

liste = list(restemp[:,1])

n\_min\_s = list(restemp[:,0]>=1.45).index(True)

t\_0\_s = restemp[0:n\_min\_s+1,0]

deb\_0\_s = restemp[0:n\_min\_s+1,1]

numsum\_deb\_0\_s = (sum(liste))

resvals\_['Volume']=(1/998.2)\*abs(trapz(deb\_0\_s,x=t\_0\_s))

resvals\_['Debit\_Volumique'] = numsum\_deb\_0\_s/(liste.\_\_len\_\_()\*998.2)

return resvals\_

# ## Lancement du programme tiers

# `launchGetdp(fichierbat\_)`

#

# Ici, on lance le logiciel getdp à partir d'un fichier batch `fichierbat\_` (le `fichierdat\_` mis à jour est inclus dans le script batch, dans d'autres cas, `fichierbat\_` et `fichierdat\_` peuvent être confondus). En l'occurrence, `fichierbat\_` est de la forme :

#

# `D:\GIV\Simu\gmsh\getdp-2.5.0-Windows\getdp.exe \*fichierdat\* -solve ... `

#

# `D:\GIV\Simu\gmsh\getdp-2.5.0-Windows\getdp.exe \*fichierdat\* -pos ...`

#

# `Pause`

# In[4]:

def launchModel():

l1 = subprocess.check\_output(r'"C:\Program Files\ANSYS Inc\v170\Framework\bin\Win64\RunWB2.exe" -X -R '+fichierdat,shell=True)

# ## Mise à jour du dictionnaire des paramètres optimisés

# `updateoptidic(optivals\_)`

#

# On met à jour la liste de dictionnaires `optidic` à partir des paramètres retournées par l'algorithme d'optimisation. `optivals\_` est directement obtenue à partir des valeurs de la solution de l'optimisation : `solution.x`.

# In[5]:

def updateoptidic(optivals\_):

assert len(optivals\_)==len(optidic)

for i in range(len(optivals\_)):

optidic[i]['value']=optivals\_[i]

# for dic,val in zip(optidic,optivals\_):

# ## Fonction à optimiser

# `optifunc(optivals\_)`

#

# Il s'agit de la fonction la plus importante : il s'agit de l'implémentation de la fonction à minimiser. Elle met à jour `optivals\_`, écrit les fichiers d'entrée du programme tiers, le lance et lit les résultats de la simulation à l'aide des fonctions précédentes. Ensuite, on calcule l'écart quadratique avec les différents paramètres à optimiser. Elle crée et met à jour un fichier de suivi `fichiertsv` des différentes valeurs.

# In[6]:

def optifunc(optivals\_):

updateoptidic(optivals\_)

writeinput(fixdic,optidic,fichiertpl,fichierdat)

subprocess.check\_output(r'DEL '+fichierpost,shell=True)

er = launchModel()

resvals=readoutput(fichierpost)

assert len(resvals)==len(resdic)

somme=0.0

for dic in resdic:

varname=dic['name']

somme=somme+(resvals[varname]-dic['target'])\*\*2/dic['target']\*\*2

with open(fichiertsv,'a') as ftsv:

for dic in fixdic+optidic:

ftsv.write(str(dic['value'])+'\t')

for dic in resdic:

varname=dic['name']

ftsv.write(str(resvals[varname])+'\t')

ftsv.write(str(somme)+'\n')

return somme

# ## Dictionnaires

# Les différentes valeurs de l'utilisateur ou issues des calculs sont stockées sous la forme de listes de dictionnaires.

#

# La liste de dictionnaires `optidic` contient les paramètres sur lesquels porte l'optimisation. La clé `name` correspond à la chaine de caractère qui doit être remplacée par sa valeur dans le fichier "template". La clé `value` correspond à la valeur initiale du paramètre, puis est remplacée par sa valeur courante pendant le calcul d'optimisation. La clé `bounds` renvoit à un tuple donnant l'intervale de variation autorisé du paramètre. La liste `optibounds`, nécessaire à l'algorithme d'optimisation, et qui contient des tuple correspondant aux intervales des différents paramètres est déduite d'`optidic`.

#

# La liste de dictionnaires `fixdic` contient les paramètres optimisables mais que l'on souhaite écarter du calcul d'optimisation sans avoir à modifier le fichier template.

#

# La liste de dictionnaires `resdic` contient les objectifs de l'optimisation. La clé `name` doit correspondre à la clé produite par la fonction `readoutput` de lecture des résultats de la simulation. Quant à `target`, il s'agit de la valeur objectif à atteindre.

# In[10]:

optidic=[

{'name':'SG' ,'value':0.00414746693961, 'bounds':(0.001,0.006)},

{'name':'Longueur\_tube' ,'value':0.0288477005874, 'bounds':(0.002,0.035)},

]

fixdic=[

]

resdic=[

{'name':'Volume', 'target': 25e-06},

{'name':'Debit\_Volumique', 'target': -1.7e-05},

]

optibounds=[dic['bounds'] for dic in optidic]

# ## Paramètres du calcul

# In[11]:

path\_folder = r'D:\QBOUNIOL\ANSYS\_Fluent\Bouchon\_vide\\'

os.chdir(path\_folder)

fichiertsv = path\_folder+r'suivi.txt' #Fichier de suivi

fichiertpl = path\_folder+r'Test\_journal\_tpl.wbjn' #Fichier template

fichierdat = path\_folder+r'Test\_journal.wbjn' #Fichier de paramètres simu

fichierpost= path\_folder+r'OptimisationGCS\_instationnaire\_bouchon\_vide\_test\_files\dp0\FLU\Fluent\mass\_flow\_rate\_outlet\_vf\_lessive.out' #Fichier contenant les résultats de la simu

with open(fichiertsv,'w') as ftsv: #Initiation du fichier de suivi

for dic in fixdic+optidic+resdic:

ftsv.write('%s\t' % (dic['name']))

ftsv.write('%s\n' % ('objective\_function'))

# # ## Algorithme differential evolution

# # À chaque itération, il y a un calcul pour chaque individu de la population.

# # In[ ]:

# solution=optimize.differential\_evolution(func=optifunc,bounds=optibounds,maxiter=8,

# popsize=10,tol=0.01,disp=True,polish=False)

# updateoptidic(solution.x)

# In[ ]:

x0=[dic['value'] for dic in optidic]

solution=optimize.minimize(optifunc, x0, bounds=optibounds,

options={'disp':True,'eps':0.0001,'gtol':1e-3})

updateoptidic(solution.x)

@echo off

:boucle

timeout /t 20

\\33fs003\TOOLS\_ET\_DOCS\TOOLS\SCIENTIFIC\CHECKLICENCES\Utils\lmutil.exe lmstat -c 1055@172.29.0.7;1055@172.29.0.8 -a > CheckLicenses.log

set a=0

findstr /C:\"acfd\_Fluent\_solver\" CheckLicenses.log

if not errorlevel 1 set /A a=a+1

findstr /C:\"acfd\_solver\" CheckLicenses.log

if not errorlevel 1 set /A a=a+1

echo %a% licenses prises

if %a%==2 goto:boucle

python Optimisation\_Workbench\_Fluent.py

pause