Approche multi-agents pour la conception optimale des systèmes mécatroniques

Amir Guizani

NNT : 2016SACLC017

THESE DE DOCTORAT
de
L'UNIVERSITE de SFAX
et de
L'UNIVERSITE PARIS-SACLAY
préparée à CentraleSupélec

ÉCOLE DOCTORALE
Sciences mécaniques et énergétiques, matériaux et géosciences - SMEMAG

Spécialité de doctorat : Sciences pour l'ingénieur

Par

M. Amir GUIZANI

Approche multi-agents pour la conception optimale des systèmes mécatroniques

Thèse présentée et soutenue à Paris, le 11 janvier 2016 :

Composition du Jury :
Mme. Christine PRELLE, Professeur, Université de Technologie de Compiègne, Présidente
Mme. Nadine PIAT, Professeur, FEMTO-ST, Besançon, Rapporteur
M. Anis CHELBI, Professeur, Ecole Nationale Supérieure d'Ingénieurs de Tunis, Rapporteur
M. Thierry SORIANO, Professeur, Université de Toulon, Directeur de thèse
M. Mohamed Slim ABBES, Professeur, Ecole Nationale d'Ingénieurs de Sfax, Directeur de thèse
M. Mohamed HADDAR, Professeur, Ecole Nationale d'Ingénieurs de Sfax, Membre
M. Jean-Yves CHOLEY, Maître de Conférences, SUPMECA, Membre
M. Moncef HAMMADI, Maître de Conférences, SUPMECA, Membre
M. Maher BARKALLAH, Maître Assistant, Ecole Nationale d'Ingénieurs de Sfax, Invité

ÉCOLE DOCTORALE
Sciences mécaniques et énergétiques, matériaux et géosciences (SMEMAG)

Titre : Approche multi-agents pour la conception optimale des systèmes mécatroniques

Mots clés : Approche multi-agents, système mécatronique, conception collaborative distribuée, véhicule électrique, modélisation, simulation, optimisation.

Résumé : La conception d'un système mécatronique est un problème d'optimisation multidisciplinaire et multi-objectif. Les approches d'optimisation actuellement utilisées, pour l'optimisation des systèmes multidisciplinaires, sont coûteuses en temps de calcul, difficiles à mettre en œuvre et non flexibles avec la phase de conception préliminaire, où les objectifs et les contraintes de conception changent fréquemment. D'où la nécessité de chercher une nouvelle technique plus simples à mettre en œuvre, moins coûteuse et qui permet d'adapter dynamiquement une solution suite à un changement des spécifications. C'est dans ce contexte que cette thèse se focalise sur le développement d'une approche multi-agents de conception qui, se basant sur les connaissances disciplinaires et par un comportement coopératif, permet de trouver collectivement une solution optimale qui satisfait les contraintes et les performances demandées.

L'approche proposée est basée sur un processus de conception pour faciliter la conception collaborative distribué des systèmes mécatroniques. Cette approche est appliquée à la conception préliminaire d'un véhicule électrique pour illustrer comment l'utilisation du paradigme multi-agent aide les concepteurs à prendre des décisions efficaces et de parvenir à une décision optimale de l'ensemble du problème. Une étude comparative avec les méthodes classiques d'optimisation est faite afin de démontrer la validité et l'efficacité de l'approche proposée.

Title : Muli-Agent approach for the Optimal Design of Mechatronic Systems

Keywords : Multi-agent approach, mechatronic systems, distributed collaborative design, electric vehicle, modeling, simulation, optimization.

Abstract : The design of a mechatronic system is a multidisciplinary and multi-objective optimization problem. Optimization approaches currently used for the optimization of multidisciplinary systems are expensive in computation time, difficult to implement, and inflexible with the preliminary design phase, in which the objectives and design constraints change frequently. It is therefore necessary to look for new techniques easier to implement and less expensive, that enable to adapt dynamically a solution due to a change in specifications. In this context that this thesis focuses on the development of a multi-agent design approach, based on disciplinary knowledge and cooperative behavior; make possible to collectively find an optimal solution that satisfies the required constraints and performance.

The proposed approach is based on a design process to facilitate collaborative distributed design of mechatronic systems. This approach is applied to the preliminary design of an electric vehicle to illustrate how the use of the multi-agent paradigm helps designers in making effective decisions and to achieve an optimal decision of the overall problem. A comparative study with traditional optimization methods is made to demonstrate the validity and effectiveness of the proposed approach.

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

Remerciements

Ce mémoire présente les travaux de recherche effectués au cours de ma thèse de doctorat en co-tutelle entre l'école nationale d'ingénieurs de Sfax (ENIS) au sein du laboratoire de recherche de mécanique, modélisation et productique (LA2MP) et l'institut supérieur de mécanique de Paris (Supmeca) au sein du Laboratoire Quartz EA7393. C'est avec un grand plaisir que je réserve ces quelques lignes en signe de gratitude et de reconnaissance à tous ceux qui ont contribué à l'élaboration de cette thèse de doctorat.

Je remercie dans un premier temps, mes directeurs de thèse M. Thierry SORIANO et M. Mohamed Slim ABBES, pour m'avoir soutenu et encouragé durant l'ensemble de mes travaux de recherche. Je tiens également à remercier M. Mohamed HADDAR ainsi que tous les membres du laboratoire de mécanique, modélisation et productique. Je lui exprime ma profonde gratitude pour sa haute compétence, il m'a initié à la recherche et m'a appris la rigueur scientifique.

Je remercie chaleureusement mes co-encadreurs de thèse, M. Moncef HAMMADI et M. Jean-Yves CHOLEY de m'avoir accueilli dans leur équipe de recherche IS2MP (Ingénierie des Systèmes Mécatroniques et Multi-physiques). Je leur témoigne toute ma reconnaissance pour les conseils qu'ils m'ont prodigués au cours de ces trois années, pour les nombreuses et fructueuses discussions que nous avons eues et pour m'avoir accordé leur confiance pour travailler à leurs côtés.

Je remercie également M. Alain RIVIERE de m'avoir accueilli dans le laboratoire Quartz. J'adresse également mes chaleureux remerciements à Mme Christel COMPAGNON et aux membres de QUARTZ, qui ont contribué chacun à sa façon, et toujours dans la bonne humeur au bon déroulement de cette thèse. Comme je tiens à remercier mes amis, parmi lesquels je remercierai plus particulièrement Hassen TRABELSI, Mahmoud MASMOUDI, Makrem MAGDOULI, Ammar BOUGUEDDIMA, Khaled DABBEBI, Fathi DJAMEL, Abdelghani LARBI et Antoine BRUNNER.

Je remercie Mme. Christine PRELLE, Professeur des Universités à Université de Technologie de Compiègne, qui m'a fait l'honneur de présider le jury.

J'adresse aussi mes remerciements à Mme. Nadine PIAT, Professeur des Universités, à l'Ecole Nationale Supérieure de Mécanique et des Microtechniques et M. Anis CHELBI, Professeur de l'enseignement supérieur à l'Ecole Nationale Supérieure des Ingénieurs de Tu-

nis d'avoir accepté la lourde tâche de rapporteur.

Je tiens à remercier et exprimer mon profond respect à M. Maher BARKALLAH qui a bien voulu accepter d'évaluer ce travail en tant que membre du Jury.

Enfin, je tiens à remercier profondément ma famille pour l'énorme amour qu'elle me porte. Mon cher père, était et restera pour toujours mon honorable exemple dans la vie. Ma mère, qui ne cesse de m'offrir tant d'affection et de dévouement, a su m'apporter tout le courage dont j'avais besoin. J'espère que ce travail sera pour elle la preuve de mon amour.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

1 Contexte de la thèse

Face à l'internationalisation des marchés et à la conjoncture économique actuelle, chaque industrie se doit d'être la plus compétitive possible en renouvelant sans cesse sa gamme de produits. Cependant, ce renouvellement n'a de sens que si le nouveau produit est proposé au bon moment aux clients et par conséquent s'il est fréquent. Pour assurer le suivi des fluctuations des marchés et se distinguer ainsi de ses concurrents, l'industrie en question se doit d'adopter de nouvelles méthodes de conception bien sûr moins coûteuses en termes financiers mais aussi moins coûteuses en termes de temps d'étude. Ces exigences du marché sont d'autant plus difficiles à satisfaire qu'à l'inverse, les produits à concevoir sont de plus en plus complexes et doivent être de plus en plus performants pour répondre, entre autres, aux nouvelles normes en termes de pollution et de sécurité ou encore à l'exigence accrue des consommateurs en termes de confort et d'assistance. Afin de concilier toutes ces contraintes de conception, de nouveaux systèmes dits systèmes mécatroniques sont apparus. Cependant, si leur pluridisciplinarité permet d'envisager des solutions originales qui n'avaient jusqu'alors pas été explorées ou autorisées par de nouvelles avancées technologiques, celle-ci demande également un changement de méthode dans le processus de conception. En effet, même si auparavant les systèmes complexes pouvaient utiliser différentes technologies issues de différents domaines de la physique, ils relevaient plus d'une juxtaposition de disciplines plutôt que d'une réelle synergie. L'approche classique de conception consistait alors à adopter une démarche d'ingénierie séquentielle où les savoir-faire des spécialistes étaient relativement cloisonnés. Puisque, par nature, les systèmes mécatroniques sont le lieu d'interactions entre différentes technologies, l'approche séquentielle de conception n'est plus appropriée pour assurer une synergie entre les disciplines de la mécatronique (mécanique, électronique et informatique temps réel). En effet, une collaboration est nécessaire entre les différents acteurs afin d'éviter tout problème d'interfaçage entre les différentes parties du système. D'autres concepts, tels que le cycle en V et le cycle en spirale, sont ainsi adoptés pour favoriser la conception collaborative. Mais, ces concepts ne fournissent aucune méthode qui permet de résoudre le problème de communication entre les outils de conception ou l'optimisation multi-disciplinaire du système.

Par ailleurs, dans le processus de conception mécatronique, la modélisation et la simulation numérique ont une place importante pour la vérification et la validation de la conception. Néanmoins, ces deux tâches sont d'une difficulté particulière si on traite la conception des systèmes mécatroniques. Tout d'abord, la mécatronique fait intervenir différents domaines

et par conséquent des méthodes de modélisation différentes. Le domaine de l'électronique inclut de l'analogique et du numérique comme il inclut les processus continus et événementiels. Le domaine de la mécanique est concerné par la modélisation géométrique 3D et le comportement cinématique et dynamique des mécanismes. Ensuite, l'intégration entre la mécanique et l'électronique a pour conséquence de faire interagir des phénomènes physiques différents et par conséquent des couplages entre : structure, fluide, électrique, thermique, électromagnétique, vibratoire, etc. Un effort supplémentaire est donc nécessaire pour identifier les grandeurs physiques communes aux phénomènes couplés et modéliser ces interactions pour pouvoir résoudre le problème et simuler le comportement du système.

L'optimisation aussi est une phase importante dans tout processus de conception et en particulier l'optimisation multi-disciplinaire en mécatronique. En effet, l'optimisation multi-disciplinaire permet aux concepteurs d'incorporer les effets de chacune des disciplines en même temps. L'optimum global ainsi trouvé doit être meilleur que les configurations trouvées en optimisant chaque discipline à part. C'est à ce niveau qu'apparaît l'importance de l'intégration synergique entre les disciplines de la mécatronique. Cependant, le problème d'optimisation multi-disciplinaire est plus compliqué et il entraîne souvent un surcoût au niveau du temps de calcul.

2 Problématique de la thèse

La conception d'un système mécatronique est un problème d'optimisation multi-disciplinaire et multi-objectif. Les approches d'optimisation actuellement utilisées, pour l'optimisation des systèmes multi-disciplinaires, sont coûteuses en temps de calcul, difficiles à mettre en œuvre et non-flexibles avec la phase de conception préliminaire, où les objectifs et les contraintes de conception changent fréquemment. D'où la nécessité de chercher une nouvelle technique plus simple à mettre en œuvre, moins coûteuse et qui permet d'adapter dynamiquement une solution suite à un changement des spécifications.

Notre proposition pour la conception optimale des systèmes mécatroniques est d'utiliser l'approche multi-agents. Cette approche est considérée comme un développement émergent d'une combinaison de tendances, y compris l'intelligence artificielle, la programmation orientée objet et des systèmes à base d'objets simultanés. Elle est déjà utilisée dans plusieurs applications industrielles pour traiter les problèmes d'analyse distribuée et la conception collaborative tels que : le développement de logiciels, la fabrication intelligente, et les systèmes de transport intelligents.

3 Objectifs de la thèse

Cette thèse se focalise sur le développement d'une approche multi-agents de conception qui, se basant sur les connaissances disciplinaires et par un comportement coopératif, permet de trouver collectivement une solution optimale qui satisfait les contraintes et les performances demandées. Cette approche est basée sur un processus de conception pour faciliter la conception collaborative distribuée des systèmes mécatroniques. L'approche développée est appliquée à la conception préliminaire d'un véhicule électrique pour illustrer comment l'uti-

lisation du paradigme multi-agent aide les concepteurs à prendre des décisions efficaces et de parvenir à une décision optimale de l'ensemble du problème. Une étude comparative avec les méthodes classiques d'optimisation est faite afin de démontrer la validité et l'efficacité de l'approche proposée.

4 Organisation du mémoire

Pour répondre à ces objectifs, le manuscrit de cette thèse s'organise de la manière suivante :

Le premier chapitre, est consacré à l'établissement du cadre général de ce travail de thèse. Après avoir décrit l'historique et présenté les différentes définitions des systèmes mécatroniques, la conception de ces systèmes est abordée suivant les deux approches ascendantes et descendantes. Ensuite, quelques outils et langages de modélisation et de simulation sont présentés. Enfin les enjeux liés à la conception des systèmes mécatroniques sont discutés.

Le deuxième chapitre présente une analyse synthétique des méthodes existantes pour supporter les enjeux liés à la conception des systèmes mécatroniques, plus particulièrement la problématique d'optimisation. Nous avons montré qu'il existe deux approches classiques pour traiter ce type de problème. La première approche l'aborde d'une manière globale, en englobant l'ensemble des données et des modèles dans une unique fonction. Des techniques d'optimisation multi-objectif servent ensuite à résoudre les conflits. À l'inverse, la seconde approche cherche à utiliser la structure du problème pour le décomposer en plusieurs sous-problèmes. Dans cette seconde approche, on cherche des solutions aux conflits en utilisant les relations entre les modèles.
Nous avons montré que l'utilisation de l'une de ces deux approches pour la résolution d'un problème d'optimisation multi-disciplinaire nécessite un nombre important d'évaluations des fonctions objectifs ce qui rend ces approches coûteuses en terme de coût de calcul et parfois empêche la convergence des algorithmes utilisés à l'intérieur de ces approches. Afin de surmonter ces problèmes, nous avons discuté dans la dernière partie de ce chapitre une technique de modélisation de substitution qui consiste à remplacer une grande partie de ces évaluations par des approximations construites à partir des modèles simples appelés méta-modèles.

Dans le troisième chapitre, nous avons commencé par une présentation des systèmes multi-agents ainsi que les différents problèmes liés à leurs structures organisationnelles. Ensuite, nous avons présenté notre approche basée sur les paradigmes multi-agents pour la conception optimale des systèmes mécatroniques. Enfin, nous avons détaillé le processus de coordination utilisé pour faciliter la conception collaborative distribuée ainsi que la mise en œuvre de l'approche multi-agents avec la plateforme de développement JADE.

L'approche adoptée est validée, dans le quatrième chapitre, par une application à un cas de conception préliminaire d'un véhicule électrique. Ainsi, une modélisation et des simulations de niveau système dans l'environnement Modelica sont présentées.

Ce mémoire se termine avec des conclusions qui rappellent les objectifs de la thèse et

les résultats proposés ainsi que des perspectives qui montrent les limites de l'approche et les améliorations qui peuvent être apportées.

Chapitre 1 : Introduction sur la conception de systèmes mécatroniques

Chapitre 1

Introduction sur la conception de systèmes mécatroniques

1.1 Introduction

L'apparition des systèmes mécatroniques depuis une vingtaine d'années peut être considérée comme une révolution pour le monde industriel. En effet, l'utilisation de ces systèmes s'est rapidement généralisée et influence actuellement la quasi-totalité des secteurs de l'industrie. En parallèle, la conception de ces systèmes est devenue de plus en plus complexe du fait de leur caractère pluridisciplinaire [14, 15]. Dans ce contexte, ce chapitre a pour objectif d'introduire la discipline d'ingénierie mécatronique.

L'organisation de ce chapitre est la suivante : après avoir décrit l'historique et présenté les différentes définitions et applications des systèmes mécatroniques, la conception de ces systèmes est abordée d'une manière très générale. Ensuite, les techniques et les outils de modélisation et de simulation qui peuvent être employés pour la conception mécatronique sont présentés. Enfin, les enjeux d'interopérabilité et d'optimisation multi-disciplinaire des systèmes mécatroniques sont discutés.

1.2 Historique

Avant les années cinquante, les systèmes complexes sont essentiellement considérés comme des ensembles électromécaniques [1]. En 1950, les semi-conducteurs sont apparus avec le développement de l'électronique de commande et de puissance. Pendant les années 60-70, les calculateurs fiables sont apparus permettant la conception de systèmes de contrôle-commande. Ces systèmes sont plus performants, plus flexibles et plus fiables puisqu'ils sont programmables. Les problèmes liés à l'intégration des systèmes contrôle-commande avec les ensembles électromécaniques a donné naissance à la discipline de la mécatronique.

Le terme mécatronique (mechatronics en anglais) a été inventé en 1969 par l'ingénieur Ja-

ponais TETSURO MORI [16] de la compagnie japonaise YASKAWA Electric Corporation [1], fabricant de composants et systèmes d'automatismes, pour désigner le contrôle des moteurs électriques par ordinateur. Il a été déposé par cette entreprise comme marque internationale en 1969, marque enregistrée en 1971. Ce mot ayant une portée générale et étant utilisé de plus en plus largement dans le jargon technique, Yaskawa abandonna ses droits en 1982 [17].

Au début des années 80, la mécatronique est définie comme étant une discipline combinant l'électronique et la mécanique (la mécanique dans son sens large qui contient la thermique, l'hydraulique, etc.). Les années 1990 ont vu un développement important de l'utilisation du terme avec de très nombreuses propositions de définitions et une évolution sensible de son périmètre et de son sens.

On pourra se référer utilement au guide VDI 2206 [18] pour plus de détails sur l'historique du terme et son évolution d'une description technique (incorporation de composants électroniques dans les mécanismes) vers une description en terme d'association de disciplines (ingénierie mécanique et ingénierie électronique) et de processus (conception et fabrication) pour aboutir au sens actuel associant une méthodologie/démarche d'ingénierie et une description technique/fonctionnelle d'un produit.

1.3 Définitions liées à la mécatronique

La définition qui a été donnée par la compagnie japonaise YASKAWA Electric Corporation est [19] : « Le mot, mécatronique, est la combinaison de 'méca' de mécanisme (mechanism en anglais) et 'tronique ' de l'électronique (electronic en anglais). En d'autres termes, les technologies et les produits développés incorporeront de plus en plus l'électronique aux mécanismes, étroitement et organiquement, de manière à rendre impossible de dire où commence l'une et où se termine l'autre ».

Par ailleurs, plusieurs définitions sont apparues, nous citons, de manière non-exhaustive, certaines d'entre-elles :

« L'intégration synergique de l'ingénierie mécanique, avec l'électronique et le contrôle informatique intelligent pour la conception et la fabrication des produits et des processus industriels» [20].

« L'intégration synergique des systèmes physiques, des actionneurs, des capteurs, de contrôle, d'électronique, et des ordinateurs à travers un processus de conception permettant une prise de décision complexe » [21].

En France, le terme mécatronique est apparu pour la première fois dans le dictionnaire Petit Larousse dans son édition 2005 et une définition normalisée a été publiée en novembre 2008.

1. "Yaskawa Electric Corporation" est une entreprise japonaise crée en 1915 et dont le siège se trouve dans la ville de Kitakyushu, Prefecture de Fukuoka (http ://www.yaskawa.co.jp/en).

Le petit Larousse définit la mécatronique comme étant une : « Technique industrielle consistant à utiliser simultanément et en symbiose, la mécanique, l'électronique, l'automatisme et l'informatique pour la conception et la fabrication de nouveaux produits ».
En novembre 2008, la norme française NF E01-010 [2] a donné sa première définition pour le terme mécatronique comme étant : « Une démarche visant l'intégration en synergie de la mécanique, l'électronique, l'automatique et l'informatique dans la conception et la fabrication d'un produit en vue d'augmenter et/ou d'optimiser sa fonctionnalité » en précisant que l'objectif de la mécatronique est d'aboutir à une valeur ajoutée supérieure à la simple somme des valeurs ajoutées des fonctions prises séparément, c'est-à-dire qu'il ne s'agit pas seulement d'assembler les composants de différents domaines technologiques, mais il est nécessaire de considérer le système mécatronique dans sa fonctionnalité globale pendant tout le cycle de conception [22].

D'une façon générale, les définitions liées au terme mécatronique mettent en évidence les aspects suivants :

- La multi-disciplinarité : la mécatronique est la "science" qui englobe la mécanique, l'électronique, l'informatique, l'hydraulique, etc. (*Figure 1.1*) ;
- La compacité : la mécatronique combine un ensemble des disciplines nécessaires à la conception et la fabrication de produits industriels plus compacts ;
- La simultanéité : la mécatronique caractérise l'utilisation simultanée et en étroite symbiose des techniques du génie mécanique, de l'automatique, de l'électronique et de l'informatique pour concevoir et fabriquer de nouveaux produits plus flexibles ;
- La fiabilité et la performance : la combinaison des aspects mécaniques et des aspects électronique garantit l'évolution vers des produits plus fiables et plus performants.

Selon la norme NF E01-010 l'intégration mécatronique se fait suivant deux niveaux physiques et fonctionnels :

- Intégration physique : interpénétration des supports électroniques et mécaniques ;
- Intégration fonctionnelle : apport de fonctions de communication, de détection, de traitement de l'information et de rétroaction aux fonctions de base mécaniques.

Suivant les niveaux d'intégrations [1], on peut aussi distinguer entre les composants mécatroniques qui ont une intégration faible ou moyenne et les produits mécatroniques qui sont caractérisés par une intégration physique et fonctionnelle élevée :

- Composant mécatronique : un composant mécatronique est un produit mécatronique qui présente un niveau partiel d'intégration mécatronique, du point de vue fonctionnel

2. NF : Mécatronique - Vocabulaire (Norme Nr. NF E01-010) : AFNOR, (2008)

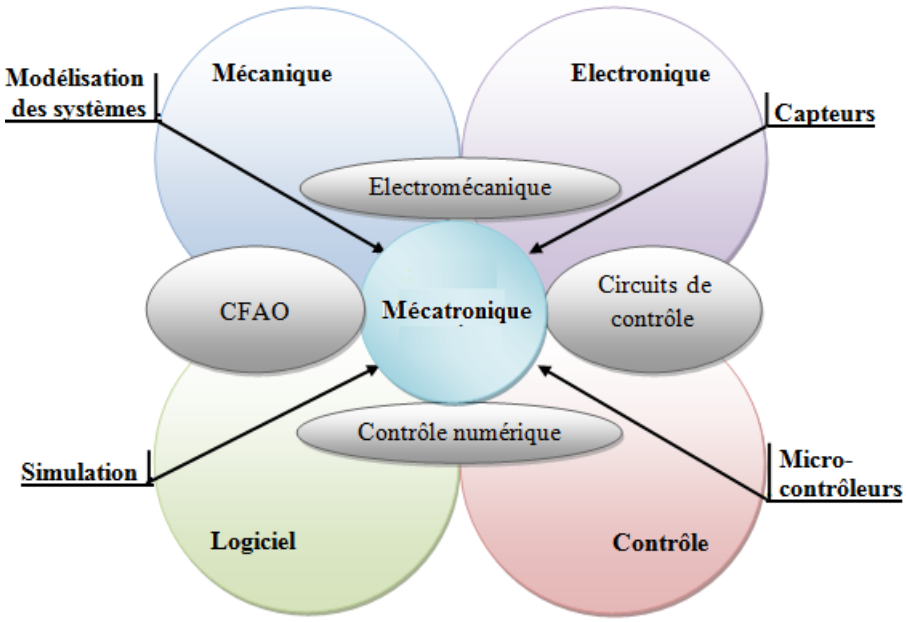


FIGURE 1.1 – Pluridisciplinarité de la mécatronique [1].

et physique. Il combine la mécanique avec l'électronique et permet le traitement de l'information.

- Produit mécatronique : un produit mécatronique est un produit ayant la capacité de traiter l'information, de percevoir, de communiquer et d'agir dans son milieu environnant. Il est caractérisé par un niveau complet d'intégration mécatronique, du point de vue fonctionnel et physique. En fonction des domaines d'activité, le mot « produit mécatronique» recouvre les notions de système, équipement de production, sous-ensemble autonome, etc.
  Dans la suite, nous utilisons le mot « système mécatronique » pour désigner à la fois les composants mécatroniques et les produits mécatroniques.

Généralement, les systèmes mécatroniques sont classifiés selon leurs caractéristiques comportementales en trois classes [23] :

- Système automatisé ;
- Système mécatronique 'intelligent' ;
- Système mécatronique en réseau.

Un système automatisé est caractérisé par un système de régulation qui lui permet de s'adapter aux changements prévisibles de son environnement de façon préprogrammée. Une machine d'usinage à commande numérique classique est un exemple de système automatisé.

Un système mécatronique intelligent se diffère d'un système automatisé par sa capacité

d'atteindre des objectifs donnés dans des conditions d'incertitude de manière imprévisible. Il est doué d'une flexibilité de régulation qui le rend capable de répondre à des fréquents changements dans son environnement sans être reprogrammé. Le système de transport personnel Segway (ou Gyropode) est un exemple de système mécatronique intelligent, il est équipé d'un système de stabilisation gyroscopique qui permet de réguler le système malgré les changements imprévisibles dans son environnement.

Un système mécatronique en réseau est un système formé des composants intelligents concurrents en vue de la réalisation des objectifs du système [7]. Une voiture moderne est un système mécatronique en réseau. En effet, les différentes unités mécatroniques constituant la voiture communiquent entre elles à travers un bus pour coordonner leurs tâches.

1.4 Différentes applications de la mécatronique

De nos jours, de nombreuses applications des systèmes mécatroniques sont développées aussi bien dans notre vie quotidienne (aspirateur, machine à café, machine à laver, etc) que dans le secteur industriel (machine-outil, chaîne d'assemblage, etc). Nous présentons dans la *Figure 1.2* quelques applications des systèmes mécatroniques dans différents domaines tels que la fabrication, le transport, la recherche spatiale, etc.

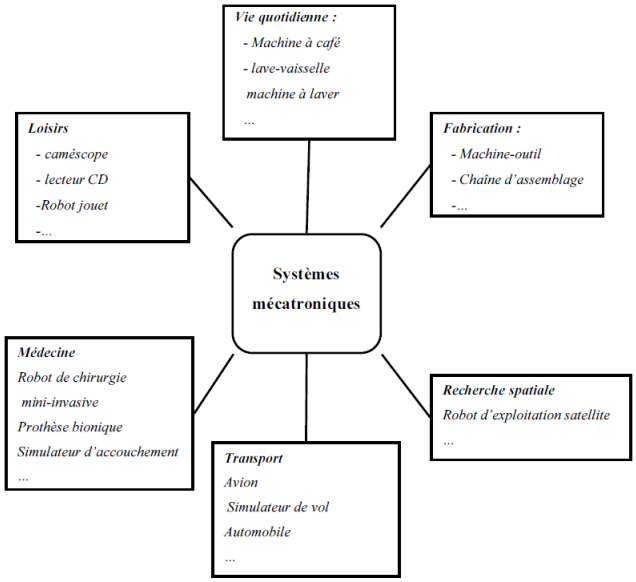


FIGURE 1.2 – Les différents domaines d'application des systèmes mécatroniques.

1.5 La conception des systèmes mécatroniques

La conception est un processus de développement [24, 25] selon lequel un ensemble de spécifications fonctionnelles est transformé à travers une série d'activités et de décisions de

conception en une description complète d'un système (produit) physique satisfaisant les exigences du marché. D'après Pahl et al. [26], un processus de conception est composé par les phases suivantes : identification des besoins du marché, analyse du problème, formulation de l'énoncé de projet, définition des spécifications du produit à concevoir, développement conceptuel, conception de forme (embodiment design), conception préliminaire, conception détaillée, conception de l'assemblage, et l'analyse du cycle de vie et l'évaluation.

Espanet [27] définit la conception de la manière suivante : « la conception correspond à la définition d'un objet ou d'un système (ensemble d'objets) répondant à un besoin défini dans un cahier des charges ». Étant entendu que le cahier des charges doit contenir la ou les fonctions à réaliser et les contraintes qui reposent sur ces fonctions. Pour cela, la conception doit prendre en compte toutes les contraintes émergeant du cycle de vie du système, en adéquation avec la démarche d'ingénierie système. Il s'appuie sur une "organisation phrasée des activités qui jalonnent la vie du système depuis l'émergence de son besoin jusqu'à son retrait de service ". En effet, le cycle de vie correspond à une vision séquentielle des phases de la vie du système. Une étude comparative des cycles de conception des systèmes suivant les différentes normes a été réalisée par l'INCOSE (International Council on Systems Engineering) [2]. Dans la partie suivante, nous nous intéressons à ces différents cycles de conception.

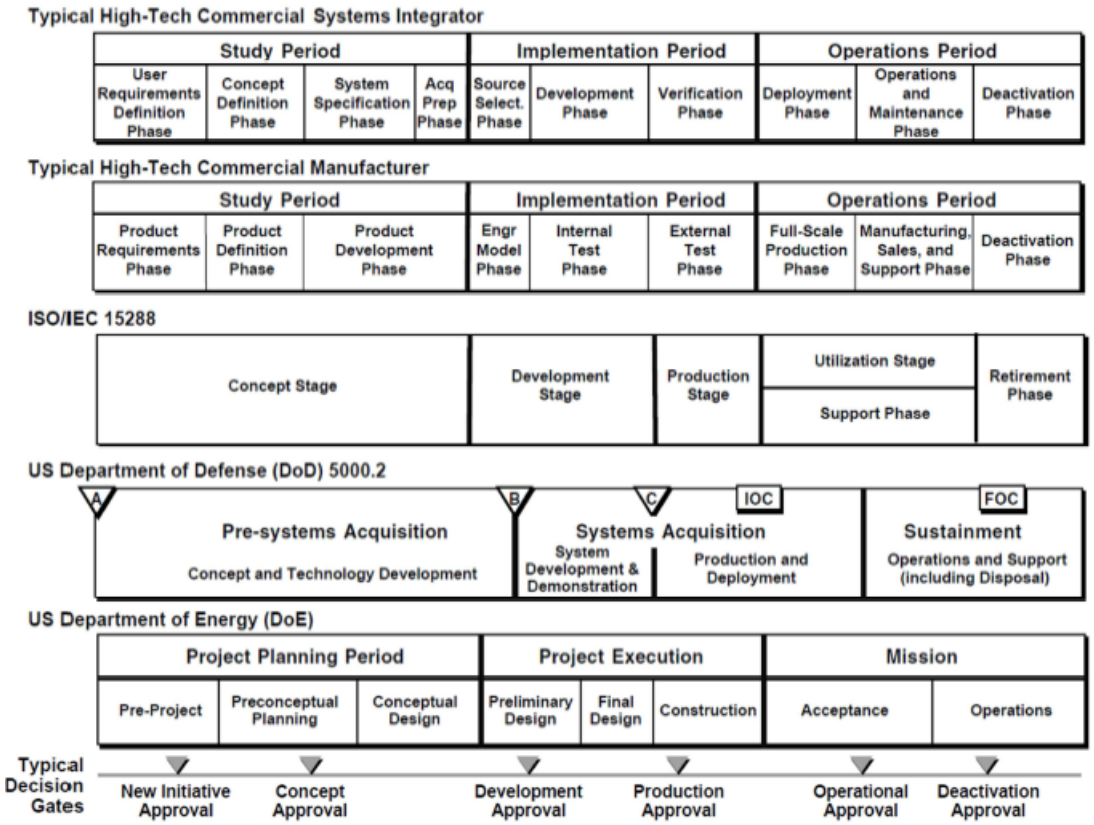


FIGURE 1.3 – Comparaison des cycles de vie selon INCOSE [2].

Globalement, les phases principales identifiées sont la conceptualisation, le développement, la production/réalisation, l'utilisation et le retrait de service [28].
Selon la norme ISO/IEC-15288 (*Figure 1.3*), le processus de conception des produits est constitué de deux étapes : l'étape des concepts (Concept stage) et l'étape de développement. L'étape des concepts, selon le US Department of Energy (DoE), est constituée des phases : avant-projet, planification pré-conceptuelle et le 'conceptual design'. L'étape de développement est constituée d'une phase de conception préliminaire et d'une phase de conception détaillée. Au final, le processus de conception dans ces phases est basé sur l'élaboration des modèles, leur vérification et leur validation [7].

1.5.1 Les principaux cycles de conception

La vie d'un produit peut être décrite de l'étape de spécifications jusqu'à l'étape de destruction suivant un cycle [29]. Les démarches de conception s'appuient généralement sur cette notion. En s'appuyant sur la synthèse développée par Tahan et al. [6], voici une présentation des principaux cycles de conception, qui se termine par le cycle en X (le cycle qu'ils ont développé).

1.5.1.1 Cycle en cascade

Le modèle de cycle de vie en cascade a été mis au point dès 1966, puis formalisé aux alentours de 1970 par Winston W. Royce [30]. Il est couramment utilisé dans l'industrie du Bâtiment et Travaux Publics (BTP). Les phases de conception ou de développement sont effectuées les unes après les autres, avec un retour sur les précédentes, pour en vérifier la conformité avant de passer à la suivante (symbolisées dans la *Figure 1.4* par des flèches vers le haut).

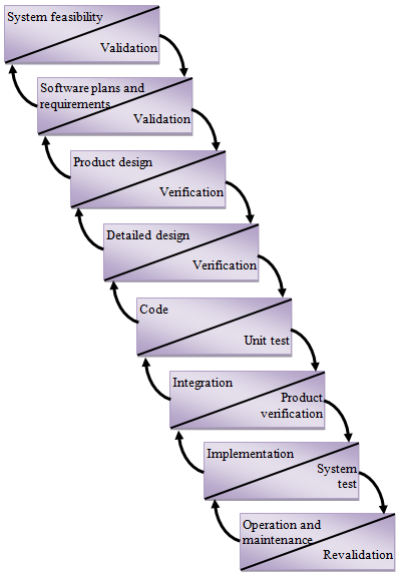


FIGURE 1.4 – Le cycle de conception en cascade [3].

Les avantages de ce modèle sont de deux natures. D'un point de vue technique, le processus est clair et systématique. Les informations sont très bien organisées, transmissibles et réutilisables. D'un point de vue gestion de projet, il fournit un cadre de référence pour la planification et le contrôle ainsi qu'une bonne visibilité des progrès et des résultats. Malgré ces avantages, ce modèle de cycle ne convient pas à la conception des systèmes complexes, car la conception de ces systèmes n'est pas séquentielle. Il est difficile de définir tous les besoins dès le début du projet. De plus, la validation est tardive.

1.5.1.2 Cycle en V

Le cycle en V [31] a été développé en 1986 par Goldberg suite aux problèmes de réactivité du modèle en cascade. Aujourd'hui, c'est le cycle le plus connu et le plus utilisé dans la conception de systèmes. Il permet de détecter très tôt d'éventuelles anomalies et donc de limiter le retour aux phases précédentes.

Comme il est indiqué dans la *Figure 1.5*, le cycle en V se compose de deux branches : une branche descendante analogue au modèle en cascade et une branche montante permet une analyse et une vérification avant l'intégration des composants : elle renvoie les spécifications de la partie descendante sur les phases en vis-à-vis de la partie montante. De la même manière, les phases de la branche montante signalent les défauts détectés/écarts aux spécifications pour chaque phase de la branche descendante.

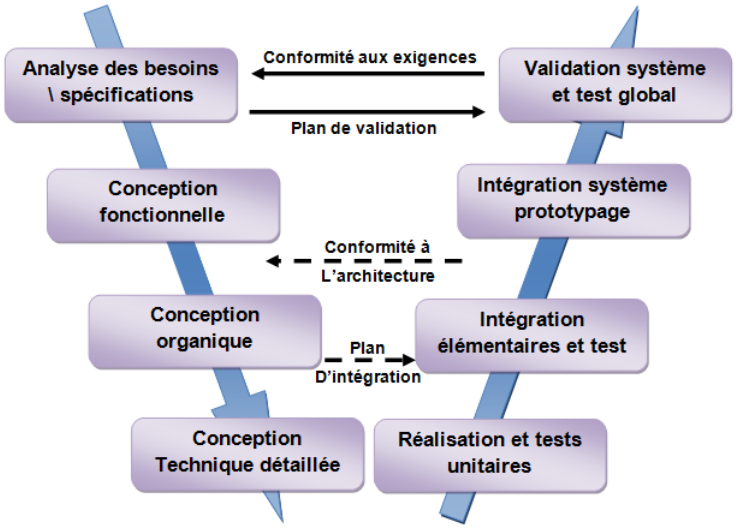


FIGURE 1.5 – Cycle de conception en V [3].

1.5.1.3 Le modèle en b

Le cycle en b [32] est une reformulation du cycle en cascade. Ce modèle de cycle a été élaboré en partant du constat que les systèmes complexes nécessitaient beaucoup de maintenance. Cette dernière couvre environ les deux-tiers de modèle de cycle de vie. Le principe de

base est la réutilisation de scénarios de tests établis pendant la phase de développement (procédures de mise en exploitation, tests de non-régression). C'est ici que réside d'ailleurs son

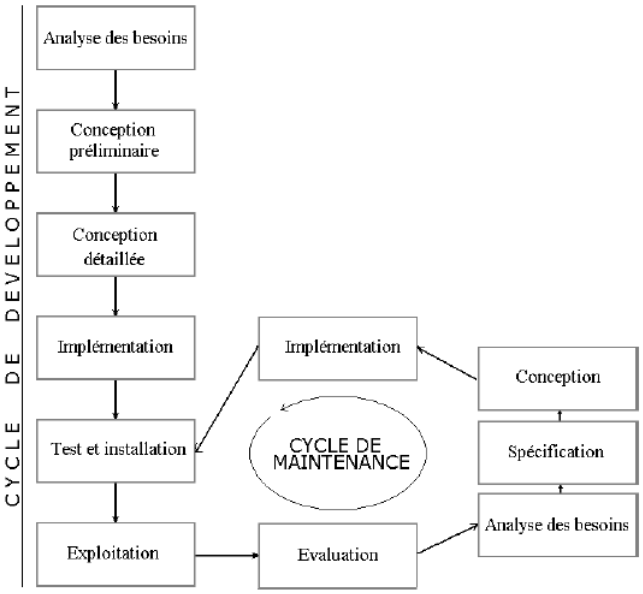


FIGURE 1.6 – Modèle de conception en b [4].

principal intérêt : on prévoit, lors de la phase projet, la réutilisation des tests et des procédures de mise en exploitation.

1.5.1.4 Cycle en spirale

Le cycle en spirale est un modèle générique de cycle de vie évolutif qui a été proposé en 1988 par Barry W. Boehm [5]. Comme il est présenté dans la *Figure 1.7*, ce cycle est divisé en quatre quadrants [33] :

- Détermination des objectifs, des alternatives et des contraintes ;
- Analyse des risques, évaluation des alternatives ;
- Développement et vérification de la solution retenue. Il est à noter qu'un modèle "classique" (cascade ou en V) peut être utilisé ici ;
- Revue des résultats et vérification du cycle suivant.

Ce cycle reprend les différentes phases du cycle en V, mais en implémentant des versions successives. L'avantage de ce cycle est d'avoir progressivement un système (produit) de plus en plus complet, au fur et à mesure que les informations sont disponibles. Sa mise en œuvre demande de grandes compétences et devrait être limitée aux grands projets innovants à cause de l'importance qu'il accorde à l'analyse des risques. L'un des inconvénients majeurs de ce type de modèle est qu'il nécessite que l'utilisateur soit régulièrement présent.

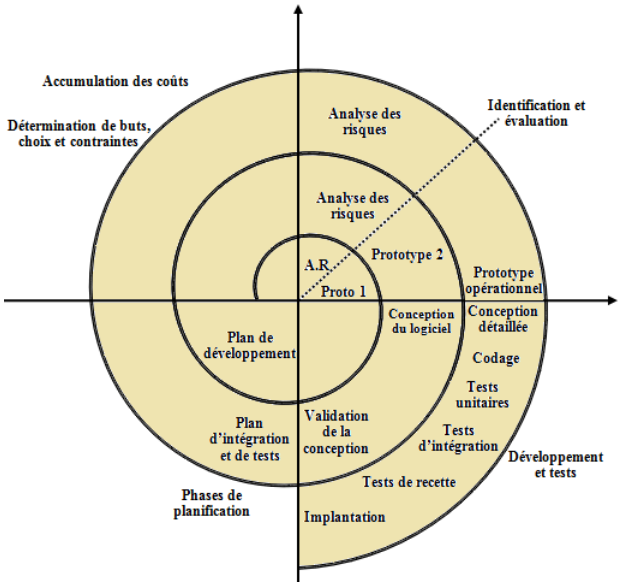


FIGURE 1.7 – Cycle de conception en spirale [5].

1.5.1.5 Cycle de conception en Y

Le cycle en Y, appelé aussi 2TUP (2 Tracks Unified Process), a été proposé au début des années 2000 par la société VALTECH [3]. Il prend en considération la dissociation entre les deux aspects fonctionnels et techniques. Le cycle est composé par trois branches (*Figure 1.8*) :

- Une branche fonctionnelle ;
- Une branche technique ;
- Une branche de réalisation.

Les deux premières branches sont abordées simultanément durant la phase de capture des besoins et la phase d'analyse. La dernière branche consiste à réunir les deux branches précédentes, permettant de mener la réalisation et la livraison du produit (ou système). L'inconvénient majeur de ce type de modèle réside dans la difficulté de mise en œuvre dans le développement de systèmes complexes et dans le cadre de grands projets. La validation nécessite de faire valider, par le client, le livrable étape par étape.

1.5.1.6 Cycle de conception en X

Le cycle de développement en X a été élaboré en 2010 par le laboratoire Lab-STICC [4]. Il est considéré comme une synthèse des cycles de conception décrits précédemment à la différence qu'il prend en considération la présence de la contrainte de conception liée à l'environnement avant, pendant et après la conception du système. Le cycle est composé par deux

3. www.valtech.fr
4. http ://www.labsticc.fr/le-pole-cid/

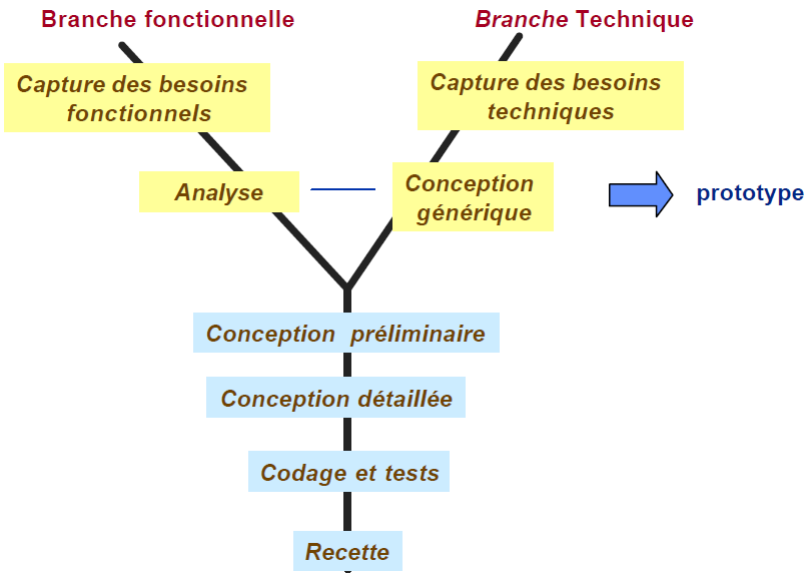


FIGURE 1.8 – La méthode 2TUP [6].

parties séparées par un axe horizontal représentant l'axe du temps (*Figure 1.9*) :

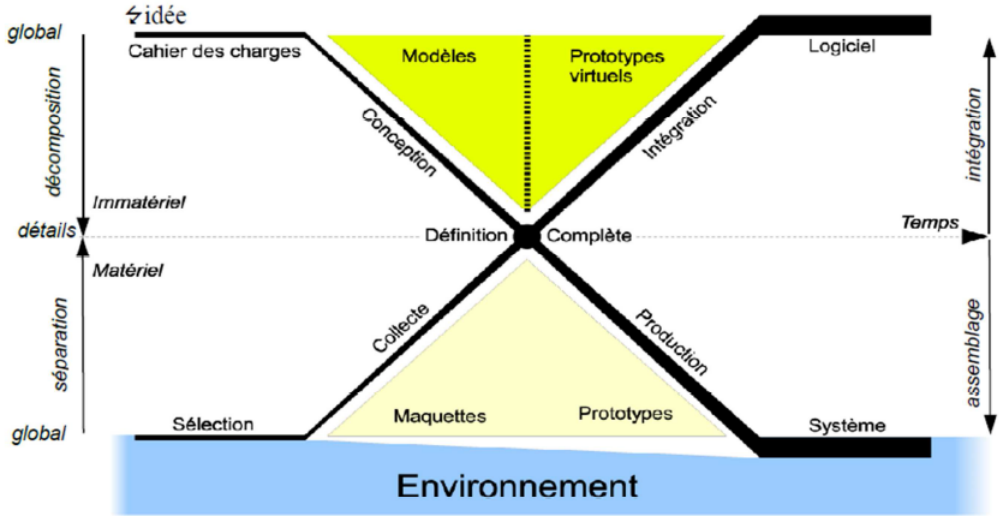


FIGURE 1.9 – Méthode de développement en X [6].

- Partie basse : correspond à tous les éléments matériels qui sont au contact avec l'environnement ;
- Partie haute : correspond à tout ce qui est immatériel et qui n'a pas d'effet direct avec l'environnement (les idées, les données, les modèles, les savoirs, etc).

Maintenant, que nous avons vu les principaux cycles de conception, nous nous intéressons aux méthodes sur lesquelles peuvent s'appuyer ces différents cycles.

1.5.2 Les méthodes de conception

Il existe deux principales méthodes de conception. Il s'agit de la conception ascendante dite « Bottom-up » et de la conception descendante dite « Top-down » [7, 28].

1.5.2.1 Approche Bottom-up

Elle dispose comme point de départ les spécifications fonctionnelles du système à concevoir, puis les composants de base (résistances, transistors, masse, ressort, etc.) sont ajoutés et liés successivement pour constituer des sous-systèmes et des ensembles plus compliqués.

L'avantage de cette méthode est sa simplicité pour construire des structures complexes. Cependant, le système global ne pourra être validé qu'à la fin de la modélisation quand toutes ses parties seront reliées. Ceci a pour conséquence de ne détecter les faiblesses du système que trop tard, après avoir investi un temps considérable en modélisation. La conception classique en mécanique ou en électronique l'utilise couramment.

1.5.2.2 Approche Top-down

Cette approche se base sur une démarche de conception allant du plus abstrait jusqu'au plus détaillé. Le point de départ est un modèle descriptif des attentes de l'utilisateur, puis le modèle est enrichi d'un ensemble de fonctions qui couvre les spécifications du système. Le modèle est ainsi successivement partitionné et raffiné jusqu'à aboutir à sa définition détaillée.

En comparaison avec la méthode « Bottom-up», les erreurs et les faiblesses du système sont découvertes au plus tôt, car la validation se fait à chaque niveau d'abstraction avant de passer au niveau suivant plus détaillé. Ce qui permet de gagner un temps important de conception. Cependant, la structure physique de certains blocs ne peut être détaillée que dans des phases très en retard, ce qui engendre des pertes de temps et nécessite l'emploi d'outils informatiques spécifiques de modélisation des systèmes. Les démarches d'ingénierie système pour le développement des produits exploitent couramment cette approche. L'inconvénient de cette approche séquentielle est que chaque séquence verrouille certains aspects de la conception devenant des contraintes supplémentaires à la séquence suivante. À cela s'ajoute-le fait que le système final n'est non seulement pas optimisé, mais en plus, la durée du processus de conception s'accroît étant donné que la tâche suivante ne peut commencer que lorsque la précédente a terminé.

Dans la section suivante, nous présentons les outils les plus fréquemment utilisés en conception des systèmes mécatroniques.

1.6 Les outils de modélisation et de simulation

La conception d'un système mécatronique est un processus délicat et long, il existe plusieurs langages et outils de modélisation et qui permettent la conception et la validation de ce processus. Parmi les langages et les outils les plus employés en conception mécatronique, on trouve :

1.6.1 SysML

SysML (Systems Modeling Language) [7, 34] est un langage standard de modélisation utilisé dans le domaine de l'ingénierie système. Il est développé par un groupe piloté par International Council on Systems Engineering (INCOSE). SysML est basé sur le profil UML (Unified Modeling Language), un langage déjà très connu pour le développement des applications informatiques. Ce langage permet de couvrir les différentes phases de conception des systèmes complexes. SysML est basé sur une modélisation graphique utilisant trois types de diagrammes :

- diagrammes pour la modélisation structurelle ou architecturale ;
- diagrammes pour la modélisation du comportement ;
- diagrammes pour la spécification des exigences.

La *Figure 1.10* montre la composition de ces différents types des diagrammes ainsi une comparaison entre les diagrammes SysML et les digrammes UML. SysML permet de gérer une grande partie des activités de conception. À titre d'exemple, pour le développement des produits, Christophe et al. [35] ont montré la capacité de ce langage à intégrer au niveau conceptuel des modèles de connaissance. En termes de spécification d'exigences, SysML permet, à partir du cahier des charges, de transcrire les exigences sous forme de diagrammes et de blocs ce qui aide à définir facilement les concepts du système à concevoir et à tracer rapidement les modèles déployés avec les exigences correspondantes. En terme de modélisation logique, SysML permet de décrire les architectures logiques du système et de définir des diagrammes de spécification des liens (association, composition, agrégation, etc.) entre les concepts. En terme de modélisation physique, SysML permet d'une part de définir l'architecture physique du système et d'autre part de décrire les lois physiques à associer aux composants et sous-systèmes en définissant les flux échangés (signaux, énergies, etc.) et les attributs physiques (surface, masse, etc.).

Par contre, SysML possède deux inconvénients majeurs, il ne permet pas de :

- Raffiner les architectures physiques 0D avec une modélisation géométrique 2D ou 3D ;
- Vérifier les architectures physiques par simulation pour pouvoir les valider.

Malgré les limitations de ce langage pour analyser les systèmes physiques, il peut être couplé en aval avec des outils de modélisation et simulation [36] tels que les langages (Bond-Graph, Modelica, Simulink, etc.). Par exemple [37] ont proposé de développer le profile ModelicaML et de l'intégrer avec SysML pour pouvoir simuler les modèles développés dans SysML avec le langage de modélisation et de simulation Modelica.

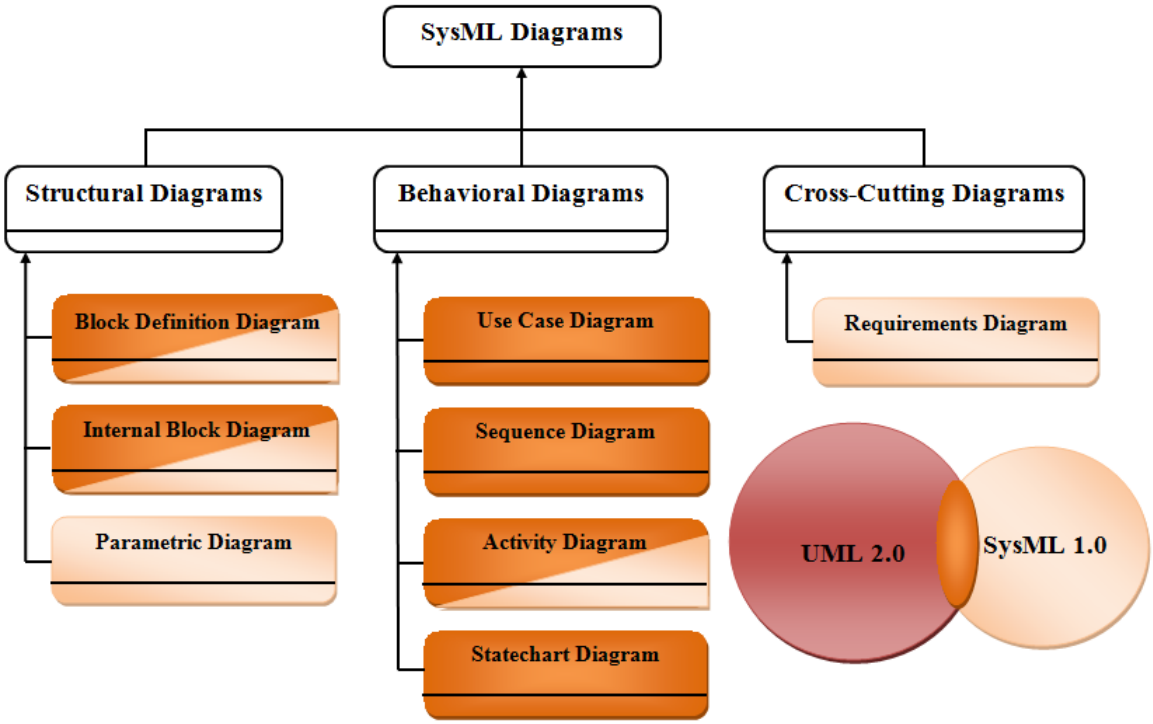


FIGURE 1.10 – Les diagrammes SysML [7].

1.6.2 Bond-Graph

L'outil Bond-Graph, appelé aussi graphe de liaisons ou graphe à liens, est une représentation graphique d'un système dynamique physique (mécanique, électrique, pneumatique, hydraulique, etc) qui a été introduit par Paynter [38]. Il est constitué par des liens qui représentent les transferts des flux d'énergie ou d'information dans le système. Le graphe de liaison peut être considéré comme un outil mathématique utilisé en ingénierie des systèmes. Il permet de modéliser un système piloté afin d'optimiser son dimensionnement et la conception de sa partie de commande. Cet outil possède plusieurs avantages :

- La distinction entre les flux d'information et les flux d'énergie ;
- Impossible d'insérer de l'énergie inexistante dans le système, puisqu'il est basé sur le principe de la conservation de l'énergie ;
- La mise en évidence de la causalité entre les flux (vitesse, courant, débit) et les efforts (force, tension, pression). Ceci qui permet de détecter des phénomènes modélisés qui ne sont pas physiques (imposé un courant dans une bobine, la vitesse d'un volant d'inertie).

Par contre, l'inconvénient majeur du Bond-Graph, est que cette méthode est un peu délicate au niveau représentation et nécessite un effort supplémentaire par le concepteur pour effectuer la modélisation correcte. La *Figure 1.11* montre d'un système masse-ressort avec son équivalent Bond-Graph.

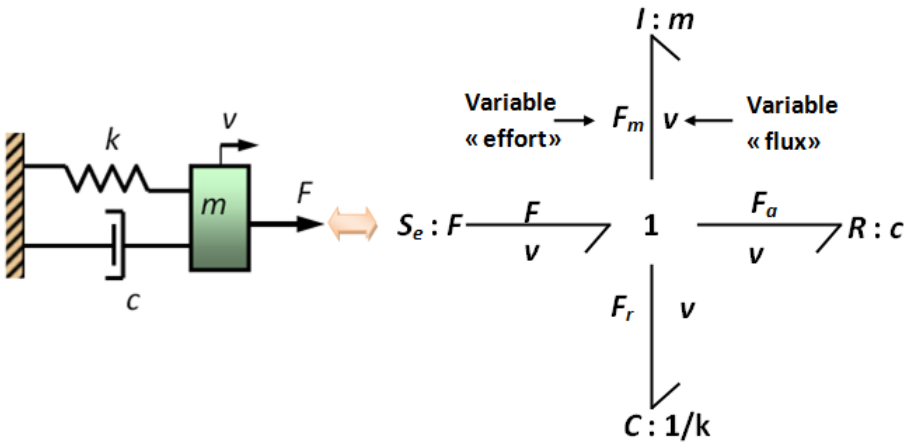


FIGURE 1.11 – Modélisation d'un système masse-ressort par Bond-graph [3].

1.6.3 Modelica

Modelica [8, 39] est un langage orienté objet de modélisation et de simulation des systèmes complexes. Il est basé sur une description mathématique des modèles. Il propose également une modélisation graphique des modèles à l'aide des diagrammes blocs disponibles dans sa bibliothèque (*Figure 1.12*). La technique de modélisation est multi-ports et peut être acausale (les ports d'un modèle n'ont pas besoin d'être déclarés comme entrée ou sortie). Les modèles créés sont réutilisables, ce qui aide à créer des bibliothèques de composants.

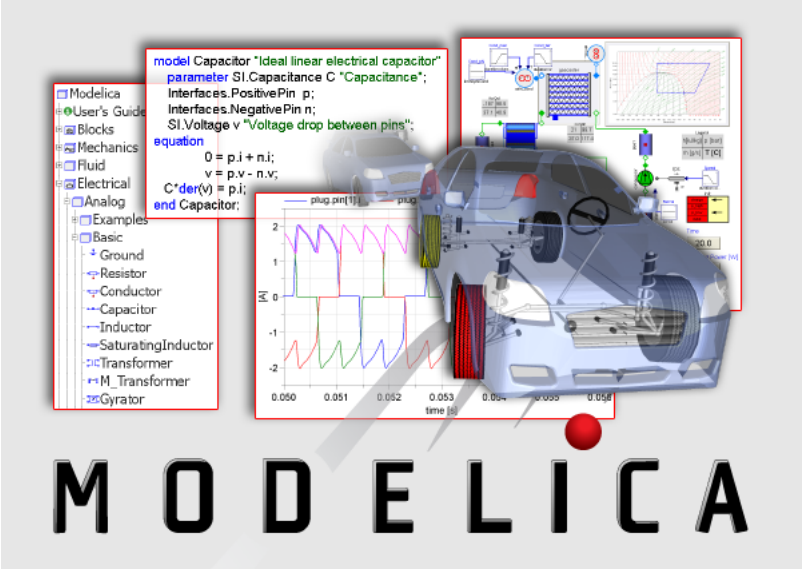


FIGURE 1.12 – Interface graphique de Modelica [8].

Modelica permet la modélisation multi-physique (mécanique, électrique, thermique, etc.)

mais il ne supporte pas la modélisation fonctionnelle et la modélisation graphique 3D. Mais, suite à l'intégration de ce langage dans l'environnement de la Conception Assistée par Ordinateur CAO (par exemple CATIA V6), en plus de l'intégration de la modélisation fonctionnelle, la société Dassault Systèmes présente déjà une plate-forme intégrée pour la conception multi-niveaux de systèmes complexes. Pourtant, il reste à faire des développements pour assurer la continuité de la simulation multi-physique entre le niveau système et le niveau détaillé des Sous-systèmes et des composants.

1.6.4 Simulink

Simulink [7, 40] est un logiciel de simulation multi-domaine et de modélisation de systèmes multi-physiques. Il fournit une plateforme graphique et un ensemble de bibliothèques contenant des blocs (*Figure 1.13*) qui permettent la modélisation, la simulation, l'implémentation et le contrôle de systèmes de communication et de traitement du signal. Simulink est intégré à MATLAB [41], ce qui permet l'accès aux nombreux outils de développement algorithmique, de visualisation et d'analyse de données de MATLAB.
Simulink permet de simuler des problèmes physiques de type : équations différentielles ordi-

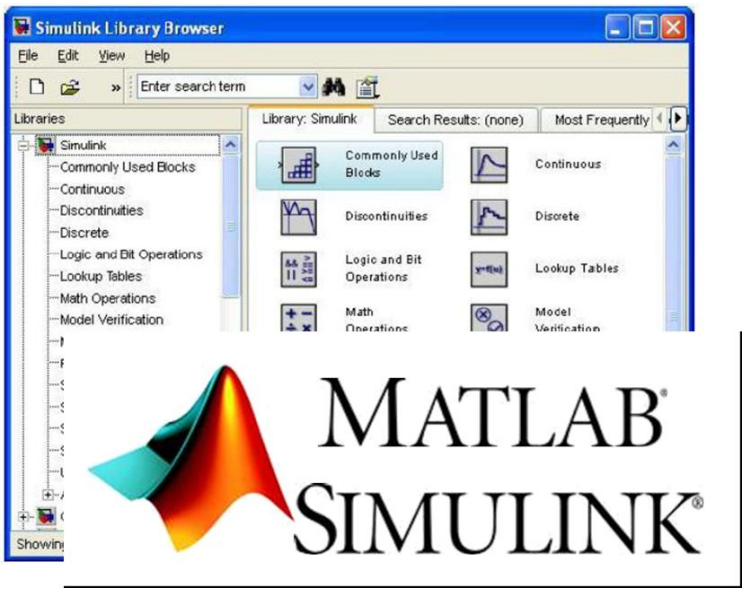


FIGURE 1.13 – Interface graphique de Simulink [3].

naires (ODE), équations algébriques (AE), et équations algèbro-différentielles (DAE). Mais, il ne supporte pas les analyses multi-physiques 3D et il est limité devant des problèmes définis par des équations aux dérivées partielles (PDE). Dans ce cas, il existe d'autres outils tels que Comsol ou Ansys qui sont mieux adaptés.

1.6.5 Comsol

Comsol [7] est un logiciel de simulation des problèmes définis par des équations aux dérivées partielles (PDEs) basé sur la méthode des éléments finis. Ce logiciel a été développé

au début comme une boîte à outils 'Toolbox' de Matlab, mais aujourd'hui il dispose de son propre environnement de modélisation. Il permet de coupler différentes PDE, de manière à définir différents phénomènes multi-physiques. Quand le modèle implémenté est de grande taille, Comsol est contraint par le coût de calcul, comme tout logiciel de calcul par éléments finis.

1.6.6 Étude comparative entre les différents outils de modélisation et de simulation

Dans le cadre du projet Outils de Modélisation Mécatronique O2M [5](Pré-dimensionnement mécatronique), une étude comparative a été effectuée entre plusieurs logiciels de simulation [9] comme il est indiqué dans la *Figure 1.14*.

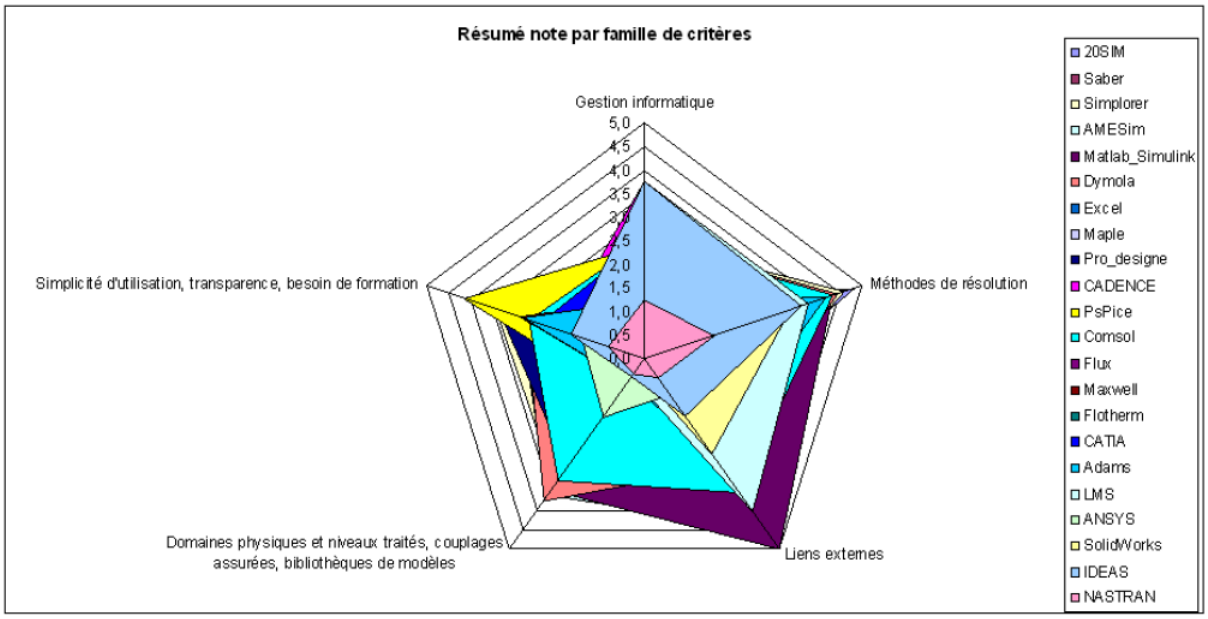


FIGURE 1.14 – Étude comparative d'outils de simulation couramment utilisés [9].

Ces outils ont été évalués suivants certains critères tels que :

- La Simplicité d'utilisation, transparence et besoin de formation ;
- Les domaines physiques traités par chaque outil ;
- Les types de couplages multi-physiques possibles ;
- La co-simulation avec d'autres outils ;
- Génération automatique des résultats d'évaluation, etc.

Dans cette étude, les langages de modélisation Modelica, VHDL-AMS et Bond-Graph ont été évalués à travers les logiciels Dymola, Simplorer et 20Sim respectivement.

5. http ://www.pole-moveo.org/pdf-projets-das/O2M-A.pdf

Cette étude comparative montre qu'aucun logiciel ne satisfait tous les critères d'évaluation à 100 %. Il existe des critères d'évaluation tels que les liens avec d'autres logiciels externes sont bien satisfaits par certains outils, mais ces outils sont limités de point de vue couplages multi-physiques assurés et domaines physiques traités.

En conclusion des parties précédentes, une bonne méthode de conception mécatronique est celle qui offre une meilleure collaboration entre les concepteurs multi-disciplinaires et qui favorise la communication entre eux. Elle doit permettre également de détecter les erreurs de conception au plus tôt. Cette méthode doit de même tenir compte des contraintes de continuité de la conception entre les différents niveaux d'abstraction de modélisation. D'où une nécessité de la prise en compte de la problématique d'interopérabilité entre les outils de modélisation par la méthode de conception.

1.7 La problématique d'interopérabilité

Généralement, la conception d'un système mécatronique est effectuée dans un entourage collaboratif constitué de différents concepteurs multi-disciplinaires. Le test et la validation du système conçu sont assurés par des logiciels de simulation. En effet, plusieurs outils de modélisation et de simulation sont utilisés durant le processus de conception afin d'obtenir la conception qui vérifie les exigences imposées dans le cahier des charges. La multi-disciplinarité des logiciels utilisés engendre des problèmes d'échanges de données aussi bien pour la modélisation que pour la simulation.

Pour résoudre le problème d'interopérabilité en conception collaborative, les concepteurs ont eu recours à l'utilisation de formats d'échanges standards de données. Comme format d'échanges standards de données, nous citons "STandard for the Exchange of Product model data (STEP)" qui correspond au standard de l'International Organization for Standardization (ISO)-10303. À partir de ce standard la représentation et l'échange de données de produits est effectuée dans le but d'intégrer son cycle de vie (conception, développement, fabrication et maintenance). Ainsi, la notion de protocole d'application (Application Protocol (AP)) a été définie pour couvrir les différents domaines du cycle de vie du produit et pour permettre de spécifier le contexte, le domaine, les besoins en information et les méthodes utilisées. En conséquence, nous trouvons les standards suivants :

- l'AP203 est un standard d'échange de données entre les logiciels de CAO ;
- L'AP214 est un format d'échange de données adapté pour le processus de conception des automobiles ;
- L'AP210 est le format d'échange des assemblages électroniques et de leurs éléments de connexion ;
- L'AP233 est destiné à l'échange de données en Ingénierie Système (IS).

Le langage d'échange de données "eXtensible Markup Language (XML)" est un format d'échange générique adapté pour l'échange de données entre les systèmes d'information et pour certains logiciels de simulation multi-domaine.

La majorité des formats d'échange sont définis en respectant la syntaxe XML. Parmi les formats d'échanges qui existent, nous citons :

- STEP-XML qui est un format d'échange STEP respectant le langage XML.
- "XML Metadata Interchange (XMI)" est un standard d'échange de données (ISO/CEI 19503) basé aussi sur le langage XML, il est largement utilisé en IS.

Ainsi, la simulation collaborative multi-domaine ou multi-physique peut être assurée de deux façons : suivant une approche modulaire ou une approche monolithique [42] :

– L'approche modulaire est une technique qui consiste à diviser le système complexe en des sous-systèmes (ou modules). Le but de cette division est d'avoir des sous-systèmes simples que nous pouvons les simulés avec des outils de simulation spécifiques disponibles. L'intégration des modules et leur simulation peuvent être assurées par la co-simulation.

– Dans l'approche monolithique, un seul environnement de développement basé sur un seul outil ou un seul langage est utilisé. Cela assure une cohérence pour l'intégration des modèles à simuler. Néanmoins, cette approche ne peut être utilisée que pour un champ limité de problèmes de simulation. Comme logiciels multi-domaines appartiennent à cette catégorie, nous trouvons (Simulink, Dymola, etc).

La co-simulation est une technique de simulation permettant d'assembler deux ou plusieurs simulateurs pour la simulation multi-physique, multi-disciplinaire et multi-échelle. Comme première solution pour appliquer la co-simulation est de développer une interface spécifique entre deux outils de simulation qui permet à l'un de deux logiciels d'accéder au processus de simulation de l'autre (par exemple l'interface de co-simulation entre Dymola et Abaqus). La deuxième solution est d'utiliser un environnement de co-simulation comme milieu d'échange commun entre plusieurs simulateurs.

Bien que la co-simulation résolve en grande partie le problème d'interopérabilité entre les simulateurs, la difficulté réside essentiellement dans sa mise en œuvre qui est en lien avec les problèmes de communication entre les simulateurs, leur synchronisation, le temps de calcul, etc. De plus, le processus de co-simulation ne permet pas d'avoir une optimisation multi-disciplinaire vue que la continuité de communication entre les simulateurs ne peut pas être assurée durant tout le processus d'optimisation.

En résumant, il n'existe pas un format d'échange standard unique pour tout le cycle de conception. En effet, au niveau détaillé de conception le format d'échange STEP est bien employé, par contre au niveau conceptuel, des formats basés sur le langage XML sont mieux adaptés. Pour la simulation multi-physique ou multi-disciplinaire, la technique de co-simulation résout un grand problème lié à l'interopérabilité entre les simulateurs, mais, elle n'offre pas la flexibilité nécessaire au concepteur qui le permet d'intervenir pendant la simulation pour modifier ou analyser les paramètres de simulation. De plus, la co-simulation n'est

pas bien adaptée pour coupler des simulateurs de différents niveaux de modélisation tels que le cas de simulateurs multi-domaines et les simulateurs des éléments finis. Enfin, pour des processus d'optimisation multi-disciplinaire, l'interopérabilité par co-simulation n'est pas une solution pratique.

1.8 Optimisation des systèmes mécatroniques

Dans un processus de conception d'un système mécatronique, l'optimisation intervient dans les différents niveaux de modélisation. En effet, au plus haut niveau de conception, l'optimisation permet d'orienter le choix parmi des architectures logiques et physiques, celles qui satisfont mieux les objectifs définis par les exigences du système et de l'entreprise. Ainsi, pour modéliser les métriques d'évaluation (telles que la performance, les coûts, la fiabilité, la sureté de fonctionnement, la disponibilité, etc.), des modèles paramétriques valides doivent être élaborés. Dans les niveaux de modélisation intermédiaires, le comportement dynamique d'un système mécatronique est optimisé à travers le bon choix entre les paramètres multi-physiques liés au modèle dynamique et les paramètres du système de contrôle.

Au niveau de la conception détaillée, le choix de la forme, des dimensions et de la topologie convenable des composants de la solution retenue est effectué par une optimisation géométrique. Cette dernière, est nécessaire pour déterminer les configurations géométriques optimales en tenant compte des contraintes multi-physiques. Ces configurations géométriques peuvent être en étroite liaison avec les paramètres physiques des composants.

Dans les différents niveaux de conception et par rapport aux différentes disciplines, l'optimisation d'un système mécatronique est une tâche un peu complexe vu qu'elle est multi-physique et que les solutions obtenues ne convergent pas toujours vers l'optimum global (problème d'optimisation locale). Ainsi, il est nécessaire d'avoir un couplage entre les différents niveaux de conception et entre les différentes disciplines pour converger vers l'optimum global.

De plus, plus que la complexité du système mécatronique augmente, plus la taille des domaines de recherches de l'optimum augmente et la tâche d'optimisation devient plus complexe. En conséquence, le coût et le temps de calcul augmentent vu qu'ils se basent sur des boucles itératives. La majorité des algorithmes d'optimisation convergent vers des résultats fiables si le nombre des variables de conception est petit, mais l'efficacité de ces algorithmes diminue en fonction du nombre des variables du système à concevoir. Une étude détaillée des différents algorithmes d'optimisation est présentée dans la suite de ce travail [43].

En conclusion, dans un processus de conception d'un système mécatronique, l'enjeu de l'optimisation multi-disciplinaire est lié à la méthode de conception, les outils de modélisation et l'interopérabilité.

1.9 Conclusion

Dans ce chapitre, nous avons commencé par une généralité sur les systèmes mécatroniques (historique, définitions et différentes applications). Nous avons vu qu'il existe de nombreuses définitions liées au terme mécatronique qui diffèrent dans leurs notions, mais qui se rencontrent sur un point commun : celui de l'intégration synergique entre les différentes disciplines (mécanique, électrique, automatique, informatique, etc.) pour concevoir et fabriquer des systèmes fiables et performants. Par la suite, nous avons parcouru les différentes méthodes de conception et les outils de modélisation les fréquemment utilisés pour développer ces systèmes.

Enfin, les enjeux liés à la conception d'un système mécatronique ont été présentés. Une analyse de ces enjeux montre que la complexité de la conception mécatronique provient d'une complexité organisationnelle liée aux méthodes de modélisation et d'une complexité technique liée aux outils de modélisation. La difficulté de l'optimisation d'un système mécatronique est liée à l'interaction de ces deux complexités.

Une synthèse des méthodes existantes pour l'optimisation de la conception des systèmes mécatroniques est présentée dans le chapitre suivant.

Chapitre 2 : Analyse synthétique des méthodes existantes pour l'optimisation des systèmes mécatroniques

Chapitre 2

Analyse synthétique des méthodes existantes pour l'optimisation des systèmes mécatroniques

2.1 Introduction

L'optimisation d'un système mécatronique est un problème difficile à réaliser. Actuellement, deux approches existent afin de résoudre ce genre de problème. La première le traite d'une façon globale, en regroupant toutes les données et modèles dans une seule fonction d'évaluation. Contrairement à la première approche, la seconde démarche consiste à travailler avec les constituants du problème afin de le décomposer en sous-problèmes. On cherche ici à trouver des solutions au problème global en considérant les relations entre les modèles. Dans ce contexte, ce chapitre a pour objectif de donner un aperçu des techniques d'optimisation qui répondent à chacune de ces deux approches.

Ce chapitre est organisé comme suit, tout d'abord nous présentons les techniques d'optimisation multi-objectif MOO (Multi-Objective Optimisation), qui vont de pair avec la première approche. Par la suite nous présentons une synthèse sur quelques techniques d'optimisation distribuées dans laquelle nous nous intéressons particulièrement aux approches d'optimisation multi-disciplinaire MDO (Multi-Disciplinary Optimisation) ainsi que l'optimisation combinatoire à base de problèmes de satisfaction de contraintes CSP (Constraint Satisfaction Problem) qui sont adaptées avec la seconde démarche.

L'utilisation de l'une de ces deux approches pour l'optimisation d'un problème multi-disciplinaire nécessite un grand nombre d'évaluations des fonctions objectifs ce qui rend ces approches coûteuses en terme de coût de calcul. L'objectif de la dernière partie de ce chapitre est de remplacer une grande partie de ce nombre d'évaluations par des approximations construites à partir des modèles appelés modèles de substitution ou méta-modèles afin de réduire le temps de calcul.

2.2 L'optimisation multi-objectif (MOO)

Le traitement du problème de conception d'un système mécatronique d'une façon globale permet de définir un ensemble d'entrées/sorties ayant des objectifs pour lesquels on se fixe une seule fonction d'évaluation. Par conséquent, l'utilisation de l'optimisation multi-objectif est adéquate pour résoudre ce type de problème.

2.2.1 La formulation du problème

Dans le but de résoudre un problème multi-objectif, il faut déterminer l'état qui obéit à un ensemble de contraintes et améliore un ensemble de fonctions objectifs. Les contraintes s'appuient sur des paramètres pour lesquels on a des exigences à satisfaire. Les objectifs sont des fonctions de coût à optimiser qui doivent tenir compte des préférences que l'on peut donner sur un certain nombre des paramètres.

Les fonctions objectifs sont généralement interdépendantes de sorte que le problème d'optimisation n'a pas une solution unique, mais plutôt un ensemble de solutions. Ceci s'explique du moment où si on minimise une fonction, on augmente la plupart du temps une autre. Donc l'application du principe d'optimum ne peut pas être clairement établie.

Dans ce cas, un problème multi-objectif est défini de la manière suivante [44] :

– Un vecteur de décisions est formé des variables du problème :

$$x = (x_1,\ \ x_2, ...,\ \ x_n) \tag{2.1}$$

avec n le nombre de variables

– Un ensemble de contraintes, désigné par :

$$g_i(x)\ \ avec\ \ i = 1, ..., m \tag{2.2}$$

avec m le nombre de contraintes

– un vecteur de fonction objectif f définit par :

$$f(x) = (f_1(x),\ \ f_2(x), ...,\ \ f_k(x)) \tag{2.3}$$

avec k est le nombre d'objectifs et f_i sont les objectifs à optimiser

Généralement, les fonctions objectifs f_i sont définies de façon à ce qu'elles soient minimisées.

Un problème d'optimisation détermine donc un vecteur x, qui obéit à l'ensemble des contraintes et optimise les fonctions qui portent sur les objectifs. Ainsi, le domaine de définition des variables de décision et les contraintes forment un espace d'actions réalisables et qui se projette dans l'espace des objectifs souhaités, comme le montre la *Figure 2.1*.

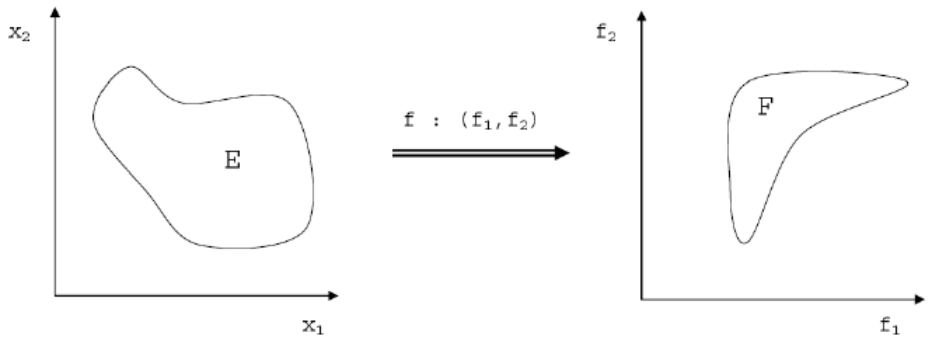


FIGURE 2.1 – Projection de l'espace des paramètres vers l'espace des objectifs [10]

Parmi les difficultés des problèmes multi-objectifs, on peut noter la non-existence d'une solution optimale. D'une manière générale, il n'existe pas une solution meilleure que toutes les autres, mais le décideur peut juger uniquement si une solution est préférable à l'autre selon certains critères [11].

2.2.2 La détermination et la comparaison des solutions

Afin de résoudre ce genre de problème, la communauté de recherche a proposé deux approches. La première cherche à rendre le problème comme un problème d'optimisation monocritère, tandis que la deuxième consiste à prendre en considération l'ensemble des critères simultanément [45].

2.2.2.1 L'agrégation des objectifs

Dans le but de rendre un problème multi-objectif à un problème mono objectif, il faut utiliser des méthodes d'agrégation des critères. Plusieurs méthodes ont été utilisées, nous citons par exemple : la méthode goal programming, la méthode du moyenne pondérée et la méthode min-max [10].

Cependant, le fait de ramener le problème à un problème monocritère prouve l'existence des limites évidentes [11]. Comme évoqué précédemment, un problème d'optimisation multi-objectif présente des interdépendances qui font qu'une solution optimale selon un critère donné ne satisfait pas les autres critères pris indépendamment.

Dans la plupart des cas, il n'y a pas de point dans l'espace de recherche où toutes les fonctions objectifs soient optimales en même temps. Dans ce cas, les décideurs vont chercher des solutions alternatives, ou des indications sur les propriétés d'un résultat.
En partant de ce principe, ramener le problème en un problème mono-objectif n'est pas évident car [11] :

– Si l'optimisation peut garantir l'existence d'une solution optimale, alors elle est unique (non adaptée à la recherche des solutions alternatives). Ainsi pour toutes les situations, le problème nécessite d'être résolu plusieurs fois.

– La mise en place d'une définition des objectifs de l'utilisateur est toujours compliquée. D'une façon générale, il est difficile de définir dès le début les critères de recherche. De ce fait, la monocriticité est très sensible à l'ensemble des paramétrages (définition des contraintes, définition des objectifs, passage à une fonction de coût unique, etc.).

Finalement, l'objectif d'un problème d'optimisation multi-objectif est de trouver un bon compromis au lieu d'une seule solution. Du moment où il y a plusieurs objectifs, la définition d'optimum change, il est donc conseillé d'employer d'autres termes notamment l'optimum de Pareto. Ce dernier est détaillé dans le paragraphe suivant.

2.2.2.2 L'approche Pareto

Les travaux en économie de Edgeworth et Pareto [46] ont abouti à l'approche Pareto. Elle est fondée sur la dominance dans la sélection des solutions trouvées. Contrairement à l'approche expliquée précédemment qui est basée sur une fonction d'utilité globale, l'approche Pareto propose un ensemble de solutions qui répondent à un certain problème sous la forme d'un front appelé Front de Pareto.

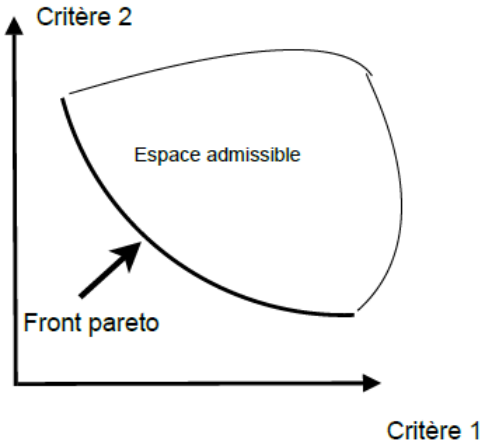


FIGURE 2.2 – Exemple de front de Pareto pour deux fonctions objectifs [10].

Pareto optimale

Une solution est dite Pareto optimale sur un ensemble d'objectifs si aucune amélioration n'est possible sur l'un de ses objectifs sans la dégradation d'au moins d'un autre [46].
Soit $f(x) = (f_1(x),\ f_2(x), \ldots,\ f_n(x))$ un ensemble d'objectifs à minimiser, on dit que $x^* \in E$ (un vecteur de variables dans un espace admissible) est une solution dominante par rapport à un autre vecteur x, si $\forall j,\ f_j(x^*) \leq f_j(x)$ et $\exists\ i$ tel que :

$$f_i(x^*) < f_i(x)$$

Les fronts de Pareto

En appliquant la définition précédente à un problème d'optimisation multi-objectif, on peut déterminer un ensemble des points (solution) dominants. La représentation graphique de cet ensemble permet de mesurer le niveau d'interdépendance entre les différentes solutions. La *Figure 2.2* représente un front de Pareto pour une fonction à deux critères. L'illustration graphique du front se fait en associant chaque critère à un axe. L'amélioration de l'un des critères provoque le détriment de l'autre. L'ensemble P des solutions Pareto optimales forme les meilleurs compromis possibles au problème.

L'utilisation de la notion de rang pour construire le front

Pour construire un front de Pareto, il suffit d'utiliser la définition précédente de la notion de dominance. On considère à nouveau un problème d'optimisation à deux critères représenté par la *Figure 2.3*. Un certain nombre de comparaisons est nécessaire afin d'établir les relations de dominance entre les différentes solutions du problème considéré. Chaque solution examinée doit être comparée à toutes les autres et si une solution B n'est pas dominée par une autre solution A alors ça ne signifie pas que B domine A.

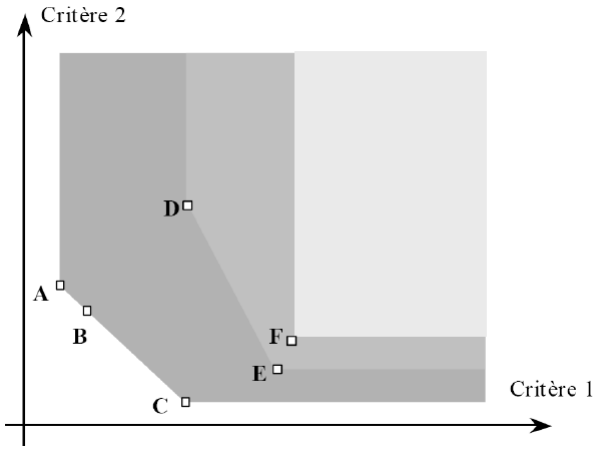


FIGURE 2.3 – Exemple de la notion de dominance [11].

À partir de la définition de dominance, on peut déduire que le front de Pareto de la *Figure 2.3* est caractérisé par :

- Trois solutions non dominées A, B, et C ;
- Les solutions A et B domine D ;
- C domine à la fois D, E et F ;
- E domine F ;
- D et F ne dominent aucun point.

La comparaison des différentes solutions le long du front de Pareto permet de définir une nouvelle notion appelée la notion de rang [46]. En effet :

- une solution qui n'est pas dominée est dite de rang 0 ;

- une solution qui est dominée par une solution de rang i est dite de rang i + 1.

Dans l'exemple de la *Figure 2.3*, les solutions *A*, *B* et *C* sont de rang 0 ; les solutions *D* et *E* sont de rang 1 et la solution *F* est de rang 2.

La méthode de comparaison des solutions à travers les fronts de Pareto est souvent plus efficace que la technique d'agrégation de paramètres. Néanmoins, elle présente certaines limites surtout lorsque la taille du problème d'optimisation est grande alors dans ce cas le parcours des fronts ainsi que la représentation graphique deviennent difficiles.

2.2.3 Les techniques de résolution

Il existe plusieurs approches pour l'optimisation d'un problème multi-objectif. Ces approches sont appliquées selon le choix du mode de représentation (Pareto ou non Pareto) [10] :

- Dans le cas de l'agrégation des objectifs (pour rendre le problème mono-objectif), le concepteur/décideur intervient en amont du processus d'optimisation. Le choix de la solution dépend des préférences fixées par le concepteur lors du choix de la méthode d'agrégation. Ces méthodes sont dites a priori.

- Contrairement au premier mode de résolution, dans la deuxième approche dite a posteriori, le concepteur intervient en aval du processus d'optimisation pour choisir une solution finale à partir d'un ensemble de solutions non dominées.
  Les *Figures 2.4* et *2.5* représentent l'intervention du concepteur avec ces deux modes de résolution.
- Le troisième mode de résolution est appelé approche interactive. Elle permet au concepteur de définir des préférences au cours du processus d'optimisation du problème de telle sorte que la formulation du problème ainsi que les solutions dominantes évoluent au cours de la résolution.

Ces trois approches d'optimisation génèrent à la fois des besoins en termes de résolution et de représentation des résultats. Et par conséquent, elles sont basées sur des algorithmes de résolutions assez différents.

2.2.3.1 La résolution par décision a priori

L'approche de résolution a priori est la méthode la plus utilisée et adaptée en milieu industriel. Son principal avantage est qu'elle se base sur des algorithmes très connus et éprouvés. Mais cette approche est dite naïve [47], puisqu'elle ne prend pas en considération la nature du problème.

Tel que décrit précédemment dans le paragraphe 2.2.2, après la pondération et l'agrégation des objectifs, on se prive d'une partie des solutions ainsi que d'une amélioration de la compréhension du problème à résoudre. Mais une fois que le problème d'optimisation multi-objectif est formulé sous forme d'un problème mono-objectif, tous les algorithmes d'optimi-

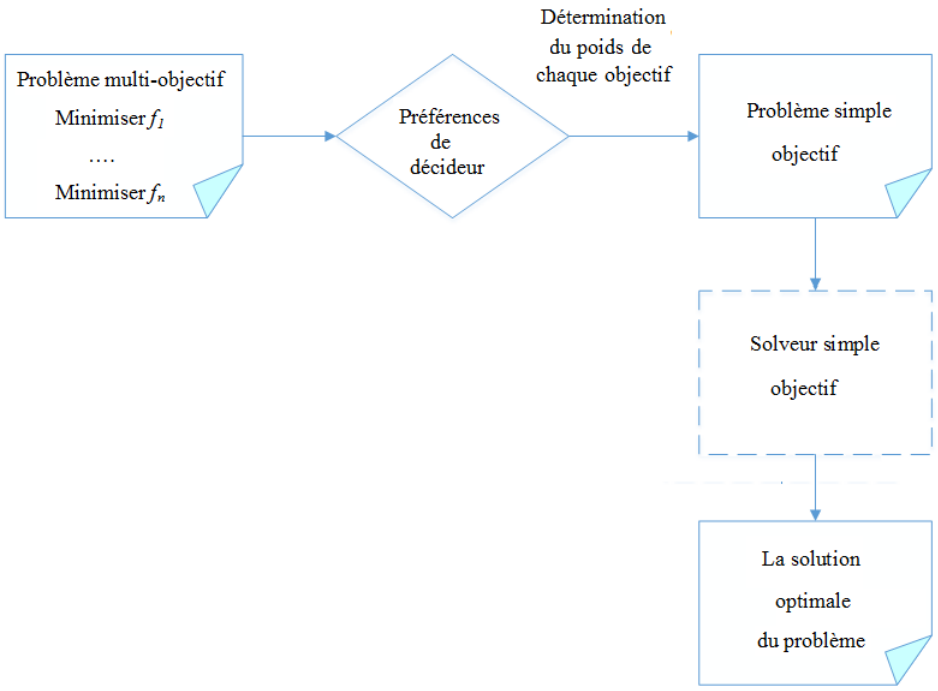


FIGURE 2.4 – Méthode de résolution a priori.

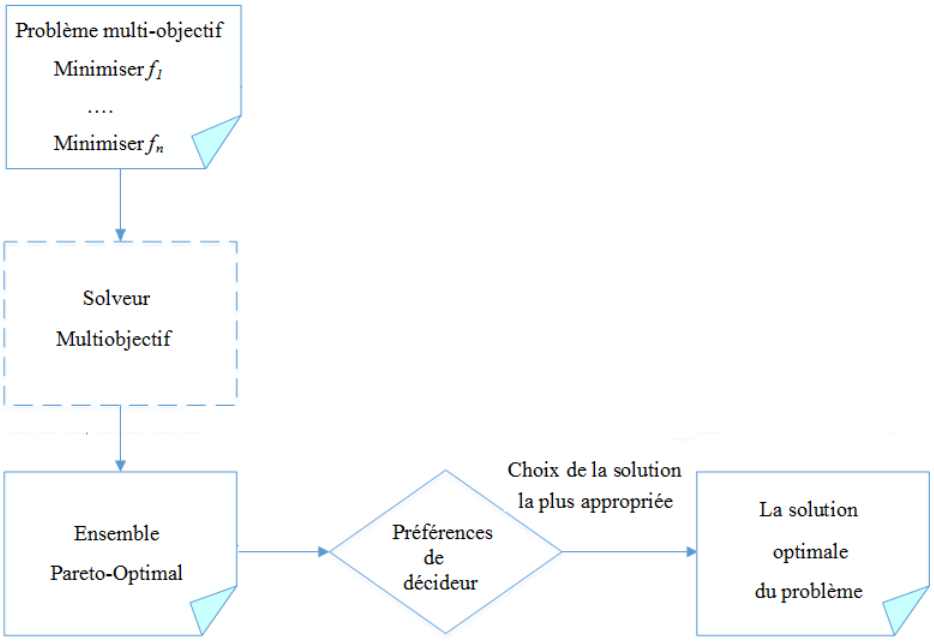


FIGURE 2.5 – Méthode de résolution a posteriori.

sation traditionnels (la recherche tabou, le gradient, le recuit simulé, etc) peuvent être utilisés pour le résoudre.

D'une façon générale, dans le cas des problèmes d'optimisation avec plusieurs interdépendances, il est préférable d'utiliser les méta-heuristiques basées sur les approches stochastiques, telles que : Monté Carlo et le recuit simulé. Ces algorithmes sont plus robustes aux minimas locaux et aux discontinuités de l'espace de recherche. Pour une description détaillée de ces algorithmes, le lecteur peut consulter [48].

2.2.3.2 La résolution par décision a posteriori

Le mode de résolution a posteriori nécessite la recherche d'un ensemble de solutions non dominées à partir duquel le concepteur/décideur pourra exercer son choix final. Les méthodes évolutionnaires ou stochastiques (telles que les méthodes basées sur les algorithmes de fourmi [49], les algorithmes génétiques, le recuit simulé et les algorithmes d'essaims particulaires [50], la recherche tabou [47]) sont les plus utilisées pour la résolution d'un problème d'optimisation multi-objectif.

Les méthodes qui traitent ce genre de problème sont classées principalement en deux catégories [10, 11] :

Les méthodes non-Pareto :

Elles sont basées sur des opérateurs nécessaires pour la recherche des solutions dominantes des différents objectifs pris séparément. À titre d'exemple, les algorithmes génétiques à sélection parallèle favorisent l'évolution simultanée de nombreuses populations, où chaque population est caractérisée par des modes de sélection qui permettent l'optimisation d'un objectif différent. En favorisant la recherche d'une solution qui optimise d'une manière indépendante chaque objectif, et en s'appuyant sur les échanges entre les différentes populations, ces méthodes peuvent aboutir à des bonnes solutions au sens de Pareto.

Les méthodes Pareto :

À l'inverse, la recherche d'une solution dans les méthodes Pareto ne favorise pas la recherche d'un seul critère, mais celle d'un ensemble de solutions non dominées, en utilisant une mémoire d'individus dominants. Dans ce cas, les solutions sont mémorisées et réutilisées pour attirer l'ensemble de la population vers les fronts. Néanmoins dans ce mode de résolution, il est nécessaire d'ajuster les paramètres à l'intérieur de l'algorithme génétique afin d'assurer l'équilibre entre l'exploitation des solutions identifiées comme dominantes et l'exploration de nouvelles solutions.

2.2.3.3 La résolution par décision interactive

Ce mode de résolution est le plus varié puisqu'il a été utilisé pour aborder deux types de problèmes, où :

- on se ramène à un problème monocritère ;
- on cherche un ensemble de solutions multi-objectif, que l'on adapte.

Afin de traiter le premier cas, on peut utiliser les méthodes d'optimisation traditionnelles pour déterminer un point de départ que l'on adapte par la suite selon les interactions avec le décideur/concepteur. Dans la plupart des cas, ce point de départ est une solution non dominée au sens de Pareto que l'on peut trouver à partir d'une première agrégation de paramètres. Ensuite, la logique de ce mode de résolution nécessite l'utilisation des analyses de sensibilité autour de ce point afin d'aider le décideur à modifier ses contraintes et/ou les facteurs de pondération (les poids) des paramètres de la fonction d'agrégation et par conséquent à affiner sa recherche en déplaçant de point à un autre pour trouver une solution que le satisfait. C'est le cas par exemple de la méthode NIMBUS [51] et iMOODS [52].

Dans le deuxième cas, les méthodes d'optimisation multi-objectifs, inspirées de celles utilisées dans la prise de décision a posteriori sont les plus adoptées pour résoudre ce type de problème. Néanmoins, une dimension supplémentaire a été ajoutée à ces approches qui permettent de garantir plus d'adaptation et d'exploration.
D'une façon générale, ceci se traduit par une diversification des individus, capables d'apporter de nouvelles solutions en cas de changements environnementaux. L'objectif est de préserver un vivier d'individus capable d'occuper une grande partie de l'espace de recherche, et d'utiliser cette caractéristique comme un facteur d'adaptation lorsque les objectifs changent. C'est le cas de la plupart des algorithmes basés sur une population d'individus des approches proposées par [10] .

2.2.4 Synthèse et limites des approches MOO

Les méthodes MOO sont classées suivant le mode de résolution en trois approches : a priori, a posteriori et interactive. La *Figure 2.6* permet de synthétiser les différents algorithmes utilisés dans chaque approche.

Comme nous avons expliqué précédemment les algorithmes multi-objectifs offrent un ensemble important de solutions qui répond à des choix conceptuels et des besoins bien précis. Le choix d'une démarche de résolution d'un problème d'optimisation multi-objectif doit se faire en essayant de répondre à un certain nombre de questions tel que [11] :

– Le problème à traiter nécessite-t-il et/ou permet-il la recherche d'un ensemble de solutions ? Ce critère permet de limiter le choix entre les approches d'agrégation des objectifs qui offrent une seule solution et celles à solutions multiples.

– Le problème est-il composé de minima locaux et/ou fortement couplé ? Ce critère est dé-

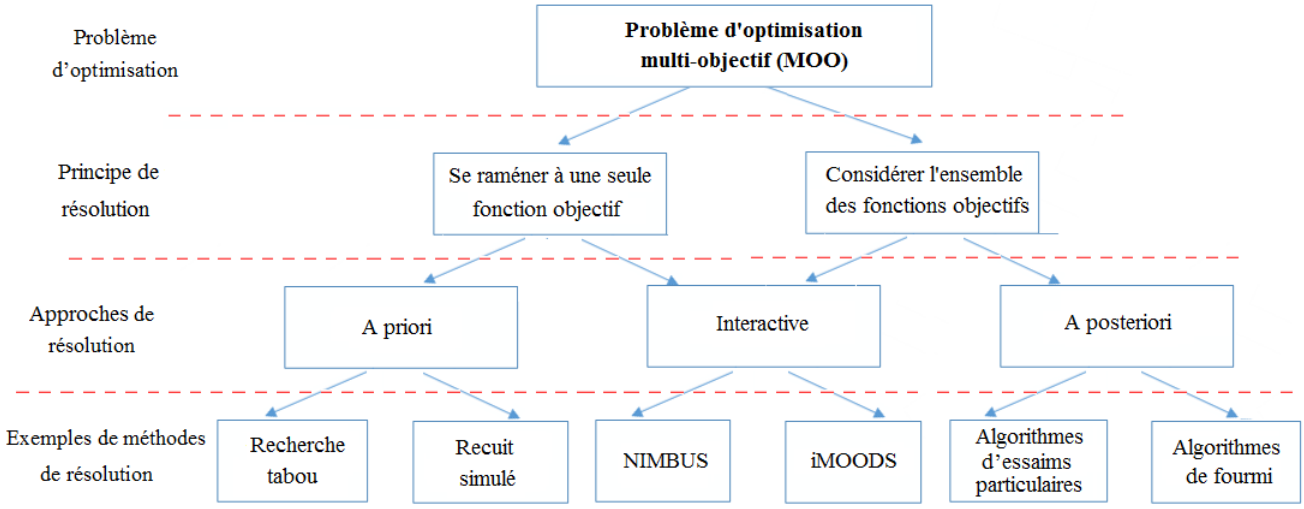


FIGURE 2.6 – Les approches et les méthodes de résolution d'un problème MOO.

licat, puisqu'on n'a pas une grande idée sur la structure exacte du problème, mais il conduit souvent au choix d'une approche d'optimisation par recherche globale (méthodes stochastiques : algorithmes à particules, évolutionnaires, de fourmis, etc.) ou locale (recuit simulé, recherche tabou, gradient, etc.). Cependant dans le cas où on cherche à trouver un ensemble de solutions, il est préférable d'utiliser les approches stochastiques.

-La dimension du problème permet-elle de déterminer l'ensemble de solutions (c.à.d. le front de Pareto) ? La réponse à cette question permet de spécifier le type de la stratégie de recherche des solutions à utiliser pour résoudre le problème d'optimisation (stratégies Pareto ou non-Pareto)

– Le concepteur a-t-il une bonne compréhension du problème à traiter ? La réponse à cette question permet définir l'emplacement du concepteur : en amont, durant ou en aval du processus de résolution.

Toutes ces questions sont loin d'être indépendantes les unes des autres, et le choix de la démarche la plus appropriée à un problème MOO donné est souvent difficile, et plusieurs améliorations peuvent être apportées. Le passage à l'échelle pose par exemple les deux problèmes suivants :

Problème lié à la représentation des Front de Pareto : dans le cas des approches a posteriori (exploratoires), la représentation des solutions sous forme d'un front de Pareto est difficile à réaliser dès que le nombre d'objectifs devient supérieur à trois. Et par conséquent, leur compréhension et leur analyse sont aussi difficiles. Or la qualité finale des solutions obtenues dépend de l'analyse et de la compréhension du problème ainsi que des choix réalisés par le concepteur, il s'agit donc, d'un inconvénient majeur (limite) de ces approches. De même, les approches a priori (d'agrégation de paramètres) sont aussi difficilement envisageables

lorsque le concepteur n'a pas une idée précise sur la nature des solutions.

L'augmentation du nombre de degrés de liberté : elle pose des problèmes aux approches a posteriori basées sur des méthodes stochastiques, car elle conduit souvent à une explosion combinatoire ou à une mauvaise exploration de l'espace.

L'approche MOO traite le problème de la conception comme une boite noire ce qui empêche l'analyse des couplages entre les différentes disciplines qui composent le système global. Pour cela, il est nécessaire de tendre vers des approches interactives qui aident les concepteurs à voir les problèmes différemment en considérant davantage les aspects disciplinaires, et en ne se concentrant pas uniquement sur des fonctions objectifs, pré-établies. Ainsi, la décomposition du problème en sous-problèmes a été envisagée par plusieurs approches, et principalement par la MDO (Multi-Disciplinary Optimisation). Pour ces approches, il s'agit, lorsqu'un problème multi-objectif est composé de plusieurs sous-fonctions (disciplines), d'essayer d'utiliser la structure du problème pour le résoudre.

2.3 L'optimisation multi-disciplinaire (MDO)

2.3.1 Définitions et terminologie

Dans la suite de cette partie, nous utilisons les notions suivantes [45, 53] :

- *Variable locale ou privée* : c'est une variable interne qui est utilisée par une seule discipline.
- *Variable partagée* : appelée aussi variable publique puisqu'elle est considérée par plus d'une discipline. Cette variable est analysée en fonction des exigences disciplinaires spécifiques.
- *Variable de conception* : elle peut être locale ou partagée.
- *Variable de couplage* : elle représente le résultat (sortie) d'une discipline et l'entrée à d'autre discipline.
- *Contrainte* : elle représente une limite au choix du décideur/concepteur. Une contrainte porte généralement sur des règles métiers, des limites imposées par les normes ou des lois physiques.
- *Contrainte interdisciplinaire* : appelée aussi contrainte de cohérence, c'est une relation mathématique (linéaire, non-linéaire, quadratique, etc.) entre les différentes disciplines.
- *Analyse* : analyser un problème, c'est résoudre un ensemble d'équations pour trouver les valeurs des variables de conception (solutions optimales) qui satisfont les performances demandées (les objectifs et les contraintes du problème).
- *Analyse interdisciplinaire* : elle permet de résoudre les équations de couplage entre les différentes disciplines.
- *Evaluation* : évaluer une solution, c.à.d. calculer le résultat de systèmes d'équations (et non pas résoudre).

2.3.2 Problématique

Une conception multi-disciplinaire englobe une multitude de composantes de différentes disciplines qui interagissent entre elles. L'optimisation globale du système à concevoir ne se limite pas à l'optimisation de chaque composante toute seule, mais elle doit tenir compte de l'effet du couplage entre les disciplines. La conception multi-disciplinaire d'un problème d'optimisation multi-objectif est identifiée comme un problème difficile à résoudre.

Dans le cas où l'analyse d'une discipline nécessite des résultats d'une autre, les deux disciplines sont dites couplées et le couplage peut être faible ou fort. Il est faible lorsque l'analyse est résolue séquentiellement. Il devient fort au moment où les résultats d'analyse d'une discipline est nécessaire pour traiter les résultats des autres, ce qui rend impossibles les calculs séquentiels. La *Figure 2.7* représente un système composé de deux disciplines avec les types de couplages.

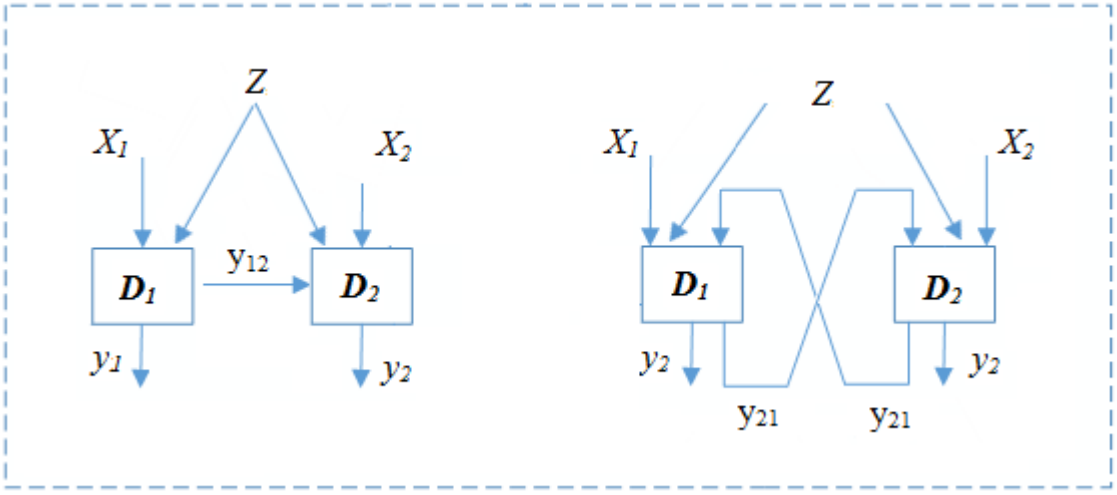


FIGURE 2.7 – Exemple de couplage entre deux composantes disciplinaires D_1 et D_2 : couplage faible à gauche et fort à droite.

Plusieurs approches ont été développées dans la littérature pour résoudre le problème de couplage. Dans la partie suivante, nous commencerons par énoncer la formulation générale d'un problème MDO. Ensuite, nous détaillerons quelques exemples de formulations des approches utilisées pour traiter ce genre du problème.

2.3.3 Formulation d'un problème MDO

La représentation simplifiée d'un problème d'optimisation multi-disciplinaire est décrite comme suit :

Afin de montrer l'interaction entre les différentes disciplines, nous détaillons dans la *Figure 2.9* le bloc analyse de la *Figure 2.8*. En effet, dans l'exemple qu'on va utiliser par la suite pour illustrer la formulation de chaque approche, le système global est composé de trois

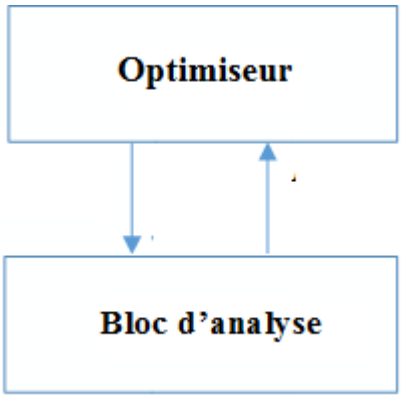


FIGURE 2.8 – Formulation d'un problème MDO.

composantes analysables en fonction des connaissances dans différents domaines disciplinaires notés D_1, D_2 et D_3, avec les deux cas deux couplages forts et faibles.
Les notations utilisées dans les *Figures 2.9* et *2.10* sont les suivantes :

- n : nombre de disciplines (dans l'exemple de la *Figure 2.9*, n=3),
- x_i : l'ensemble des variables disciplinaires de la discipline i,
- y_i : l'ensemble des résultats (sorties) de la discipline i. Les valeurs de y_i entrent dans le calcul des contraintes et des fonctions objectifs du système,
- y_{ij} : l'ensemble des variables de couplage, sont les résultats d'une discipline i et les entrées à une autre discipline j,
- X : l'ensemble des variables disciplinaires,
- Y : l'ensemble des sorties de l'analyse du système,
- Z : l'ensemble des variables partagées,
- V : l'ensemble des variables disciplinaires et partagées, $V = X \cup Z$,
- M : l'ensemble des variables de couplage,
- F : l'ensemble des fonctions objectifs,
- C : l'ensemble des contraintes.

La formulation générale d'un problème d'optimisation multi-disciplinaire est donnée par :

$$\begin{cases} Minimiser \;\; F(V) \\ Sous \;\; les \;\; contraintes \\ C(V) \leq 0 \end{cases}$$

Comme il est indiqué dans la *Figure 2.9*, le couplage entre les deux disciplines D_1 et D_2 est faible, c'est-à-dire qu'au cours de l'analyse d'une solution donnée par l'optimiseur, l'ensemble des valeurs des variables de couplage entre D_1 et D_2 peut être déterminé par des interactions séquentielles. À l'inverse, les disciplines D_2 et D_3 sont fortement couplées puisque les entrées de chaque discipline dépendent des sorties de l'autre. Dans ce cas, les variables de couplage deviennent plus difficiles à estimer et théoriquement incalculables sé-

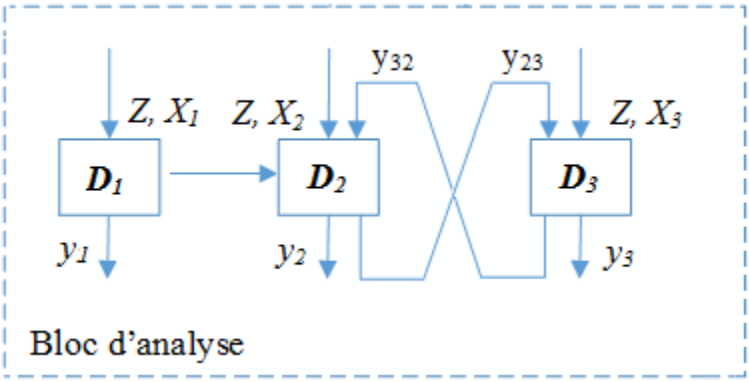


FIGURE 2.9 – Bloc d'analyse détaillé d'un système composé de trois disciplines, avec un couplage faible entre D_1 et D_2 et un couplage fort entre D_2 et D_3.

quentiellement sans la présence des hypothèses sur leurs valeurs. Un exemple de couplage fort entre 3 disciplines est mentionné dans la *Figure 2.9*.

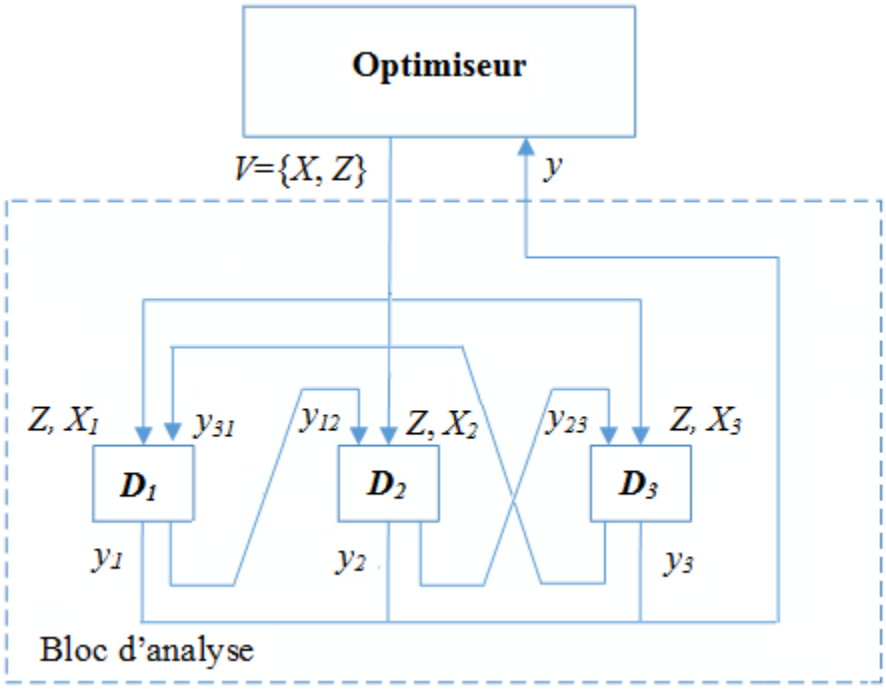


FIGURE 2.10 – Bloc d'analyse détaillé, avec couplage fort entre les disciplines D_1, D_2 et D_3.

Un exemple de couplage fort entre 3 disciplines est mentionné dans la *Figure 2.10*. Dans cet exemple, les disciplines adjacentes sont faiblement couplées, c'est-à-dire qu'au cours de l'analyse d'une solution fournie par l'optimiseur, les valeurs des variables de couplage entre deux disciplines adjacentes (D_1 avec D_2, D_2 avec D_3 ou D_3 avec D_1) peuvent être déterminées à travers des interactions séquentielles. Toutefois, le cycle d'attente des résultats rend le

couplage fort (les entrées de chaque discipline dépendent des sorties l'une de l'autre).

Dans ce qui suit, nous introduisons quelques approches d'optimisation multi-disciplinaire utilisées pour traiter et résoudre la problématique du couplage fort entre les différentes disciplines. Dans certaines approches, la résolution se fait au cours de l'analyse système et dans d'autres par l'optimiseur.

2.3.4 Stratégies d'optimisation multi-disciplinaire

La littérature introduit des notions de stratégie de résolution, de méthode de résolution, de formulation, de reformulation et de système d'équations. Dans la suite de cette partie, nous utiliserons ces termes définis de la façon suivante [45] :

- *Approche ou Stratégie de résolution* : elle permet de distinguer entre les deux cas suivants, où :
  – le système est considéré comme un seul bloc.
  – le système est décomposé en plusieurs blocs disciplinaires.
- *Méthode de résolution* : elle représente les techniques et les algorithmes d'optimisation appliqués à la stratégie de résolution choisie afin de résoudre le problème du couplage entre les différentes disciplines.
- *Formulation* : Formuler un problème d'optimisation multi-disciplinaire, c'est-à-dire définir toutes les composantes de ce problème comme les variables locales, les variables partagées, les systèmes d'équations, les contraintes et les objectifs.
- *Reformulation* : Reformuler un problème d'optimisation, c'est-à-dire modifier au moins une de ces composantes (variable, contrainte ou objectif) de la formulation courante.
- *Système d'équations* : appelé aussi les équations d'état, c'est l'ensemble d'équations nécessaires pour déterminer les valeurs des sorties du système (contrainte, objectifs, etc.) en fonction des valeurs de ses entrées (variables de conception).

Plusieurs travaux, et notamment ceux de [54–59] insistent sur l'importance des formulations utilisées pour la résolution d'un problème MDO. Alexandrov a présenté [60–62] une classification importante des différentes formulations. Cette classification est basée sur l'aspect fermeture/ouverture des contraintes de conception, des contraintes interdisciplinaires et de l'analyse disciplinaire. Une formulation est dite stricte ou fermée lorsque toutes les contraintes sont satisfaites à chaque étape d'optimisation.

Les méthodes de résolution sont détaillées dans d'autres travaux, et notamment ceux de [63–66], où la notion de stratégie de résolution (mono ou multi-niveaux) est présente.

Suite à l'analyse de ces travaux, nous distinguons deux manières de traiter les problèmes MDO : choisir une stratégie de résolution puis une méthode de résolution en fonction des caractéristiques et de la nature du problème à résoudre et, à partir de ces choix, formuler le problème en écrivant le système d'équations qui en résulte, ou encore : choisir une formulation et ensuite les stratégies et méthodes de résolution les mieux adaptées à cette formulation. Les *Figures 2.11* et *2.12* mettent en évidence ces deux façons d'aborder un problème MDO.

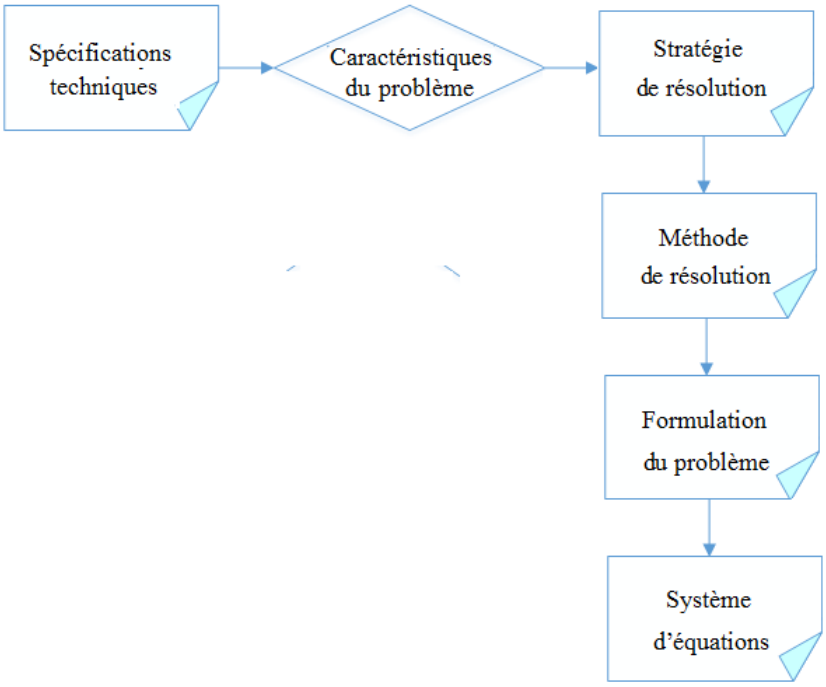


FIGURE 2.11 – Processus de résolution d'un problème MDO.

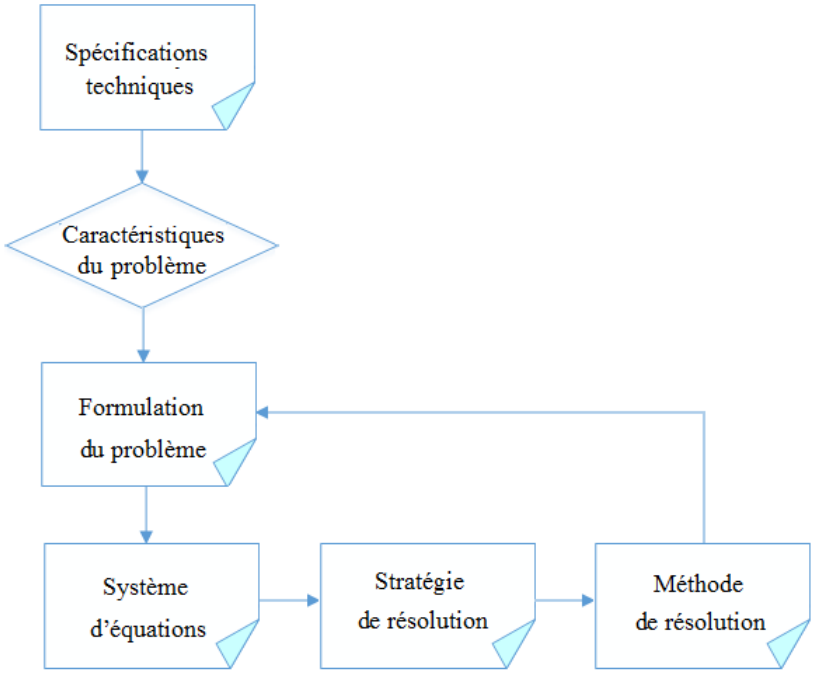


FIGURE 2.12 – Processus de résolution d'un problème MDO avec reformulation.

À travers la *Figure 2.12*, la reformulation d'un problème permet de le résoudre autrement avec un optimiseur plus adapté. En effet, certains optimiseurs sont plus efficaces que d'autres sur certains problèmes MDO. Reformuler un problème MDO, c'est redéfinir son système d'équations afin d'avoir un système équivalent, mais étant plus ou moins optimisable. Néanmoins, les formulations ayant des modèles et des paramètres multi-niveaux sont difficiles à résoudre et plus coûteuses en temps de calcul [65]. S'agissant d'une approximation des fonctions objectifs, d'une transformation des objectifs en contraintes ou bien d'une relaxation de contraintes [45], la reformulation d'un problème peut défavoriser la convergence et la robustesse de l'algorithme utilisé pour l'optimisation.

Dans la suite de cette partie, nous détaillons les approches les plus adaptées à la résolution des problèmes MDO [58, 63–67]. Ces approches sont classées suivant la stratégie d'optimisation ou de résolution : mono-niveau (globale) ou multi-niveaux (disciplinaire).

2.3.4.1 Stratégies mono-niveau

Dans les approches globales (mono-niveau), le problème MDO est examiné comme étant un ensemble de paramètres à optimiser. Dans ce cas, un seul optimiseur global peut assurer le processus d'optimisation, et toutes les disciplines sont considérées comme des outils d'évaluation ou d'analyse. Ces approches sont appliquées lorsqu'on dispose de toutes les données du problème telles que : les variables de conception (locales, partagées et de couplage), les contraintes (disciplinaires et interdisciplinaires), les objectifs et les équations d'état.

A. Multi-Discipline Feasible (MDF)

Approche MDF [67] limite les variables à optimiser aux variables de conception en écartant les variables de couplages. Une analyse multi-disciplinaire est effectuée à chaque itération de l'optimisation dans le but d'assurer la faisabilité de la solution. Cette approche fréquemment utilisée présente un inconvénient majeur, elle est caractérisée par un temps d'implémentation et d'exécution très lourd puisque l'analyse multi-disciplinaire du système global doit se faire à chaque itération. Elle se propose de centraliser les données et les connaissances.
Dans l'approche MDF, l'optimiseur ne tient pas compte des variables de couplage. Elles sont calculées par l'analyseur et par conséquent la formulation du problème MDO devient :

$$\begin{cases} Minimiser\ \ F(V); \\ Avec\ \ V = Z\ \cup\ X; \\ Sous\ \ les\ \ contraintes \\ C(V) \leq 0; \end{cases}$$

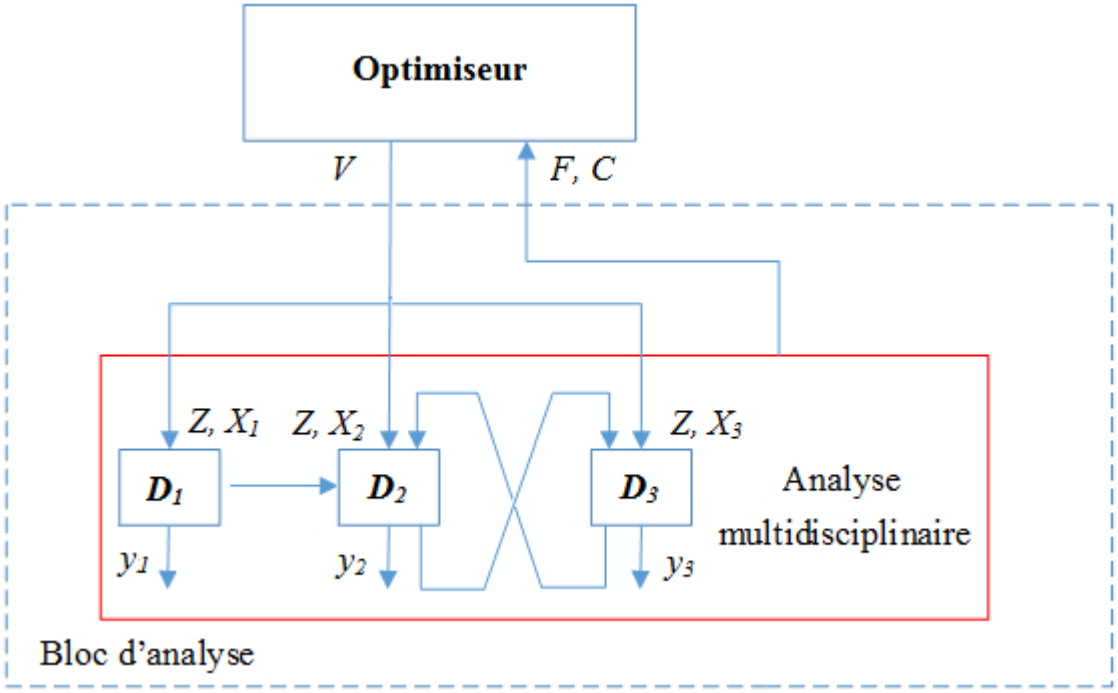


FIGURE 2.13 – Approche d'optimisation MDF.

B. All-At-Once (AAO)

À l'inverse de l'approche MDF, la méthode AAO [68] prend en considération toutes les variables de couplages en les intégrant dans l'ensemble de variables à optimiser. L'évaluation d'une solution doit se faire uniquement à la rencontre d'un optimum local ou global [57].

Dans cette méthode, les contraintes de vérification de l'égalité entre les variables de couplages calculées par le bloc d'analyse et les valeurs choisies par l'optimiseur sont ajoutées aux contraintes $C(V)$, et dans ce cas, nous parlons bien d'évaluation disciplinaire au lieu d'analyse disciplinaire. Le système d'équations (équations d'état) n'est pas forcément satisfait à chaque itération de l'optimisation. Il doit l'être à la convergence de l'algorithme.

Approche AAO est caractérisée par sa rapidité de calcul par rapport à l'approche MDF, mais son inconvénient majeur est qu'elle augmente le nombre de variables à optimiser ce qui engendre la complexité du processus d'optimisation. De plus, la convergence n'est pas assurée dans ce cas, car l'évaluation d'une solution ne se fait qu'à la rencontre d'un optimum (si l'optimum global n'est pas atteint alors la faisabilité de la solution ne peut pas être assurée). Et par conséquent, cette approche ne peut pas être la plus appropriée dans la phase de conception détaillée des systèmes compliqués. Néanmoins, elle est couramment utilisée comme un outil de vérification de solutions obtenues avec des méthodes multi-niveaux [69].

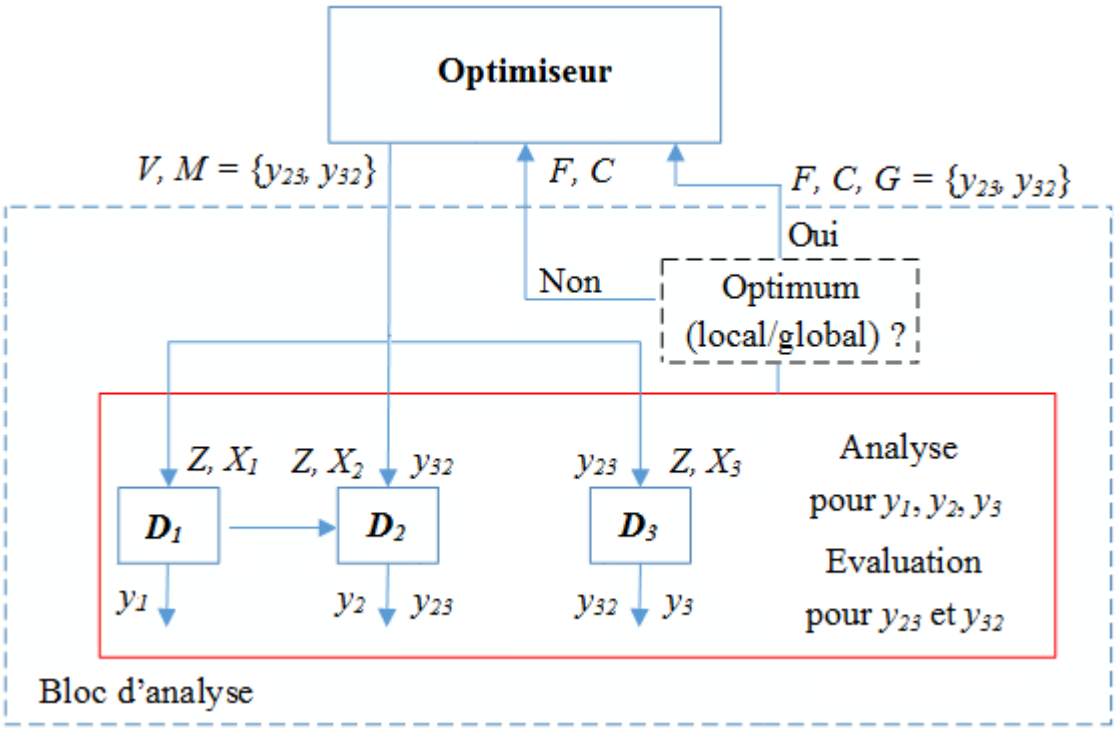


FIGURE 2.14 – Approche d'optimisation AAO.

Les variables de couplage sont rajoutées à l'ensemble des variables d'optimisation. La faisabilité disciplinaire est vérifiée à la rencontre d'un optimum local ou global, c'est-à-dire à la convergence. Des contraintes d'égalité sont rajoutées au système d'équations pour vérifier l'égalité entre les valeurs des variables de couplages proposées par l'optimiseur et celles utilisées par les évaluateurs disciplinaires. Dans ce cas, le problème d'optimisation AAO s'écrit sous la forme suivante :

$$\begin{cases} Minimiser \;\; F(V, y_{ij}); \\ Sous \;\; les \;\; contraintes \;\; disciplinaires \\ C(V, y_{ij}); \end{cases}$$

Et les contraintes de cohérence interdisciplinaires dans le cas d'un optimum sont écrites sous la forme $M - G = 0$; où G est l'ensemble des variables de couplage évaluées à partir des données fournies par l'optimiseur.

C. Individual-Discipline Feasible (IDF)

L'approche IDF [45, 68] est considérée comme une approche intermédiaire entre les approches MDF et AAO. En effet, l'approche AAO vérifie la faisabilité disciplinaire seulement à la rencontre d'un optimum local ou global, l'approche MDF requiert une faisabilité multidisciplinaire, le principe général de l'approche IDF est d'assurer la faisabilité disciplinaire à

chaque itération.

Comme dans l'approche AAO, les variables de couplages sont transformées aussi en variables à optimiser. En effet, les contraintes d'égalités rajoutées sont vérifiées à chaque convergence de l'optimiseur pour trouver une concordance interdisciplinaire (un équilibre) entre l'ensemble des variables de couplages.

Le processus de résolution de l'approche IDF est moins (réciproquement plus) complexe que celui de la méthode AAO (réciproquement MDF), et la résolution est moins (réciproquement plus) gourmande en temps de calcul et coût et que la méthode MDF (réciproquement AAO).

La *Figure 2.15* montre que l'analyse disciplinaire est effectuée à chaque itération du processus d'optimisation. La faisabilité d'une solution est assurée par les contraintes d'égalité sur les variables de couplage dont les valeurs sont données par l'optimiseur.

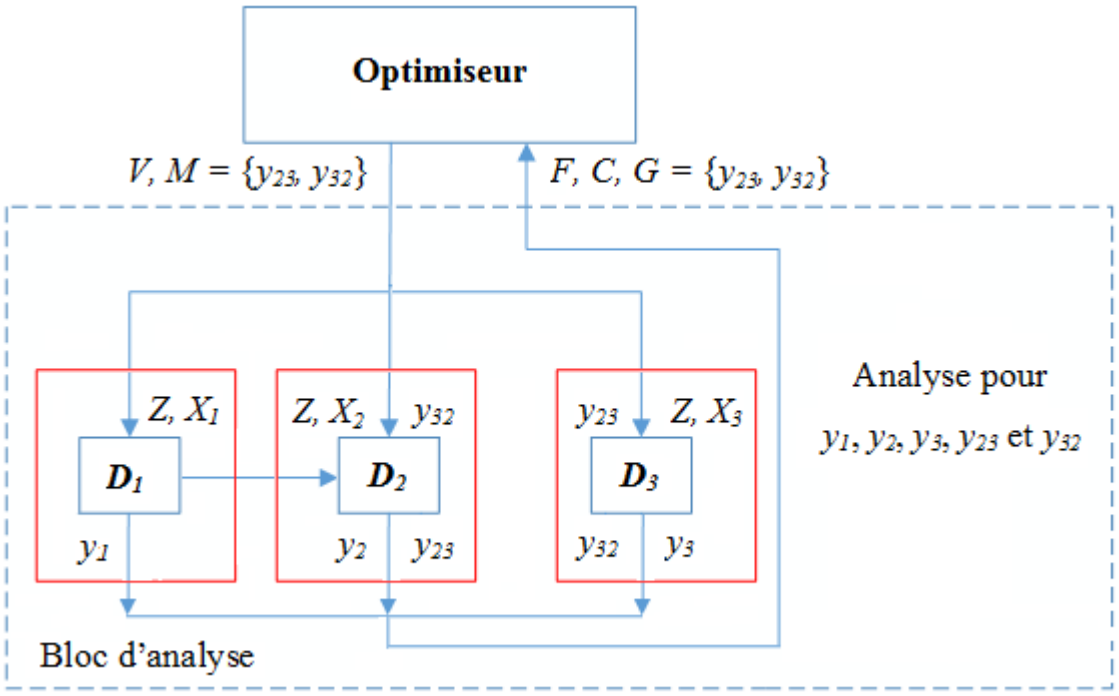


FIGURE 2.15 – Approche d'optimisation IDF.

La formulation du problème d'optimisation IDF est donnée par :

$$\begin{cases} Minimiser \ \ F(V, y_{ij}); \\ Sous \ \ les \ \ contraintes \ \ disciplinaires \\ C(V, y_{ij}); \end{cases}$$

Et les contraintes de cohérence interdisciplinaires à chaque itération sont écrites sous la forme $M - G = 0$; où G est l'ensemble des variables de couplage évaluées à partir des données fournies par l'optimiseur.

2.3.4.2 Stratégies multi-niveaux

Les approches d'optimisation multi-niveaux (ou disciplinaires) font intervenir à la fois des optimiseurs aux niveaux système et sous-systèmes. Et par conséquent, les disciplines ne sont pas considérées comme des outils d'analyse simples, mais plutôt comme des optimiseurs locaux.

Ces approches sont appliquées lorsqu'une partie du système global est considérée comme une boite noire avec des entrées et des sorties (c'est-à-dire un manque des données pour cette partie) ; ceci est dû généralement à la confidentialité inter-discipline [45]. Parfois, les approches mono-niveaux, et notamment l'approche AAO, sont utilisées comme un outil de vérification de la pertinence des solutions obtenues avec une approche multi-niveaux. La *Figure 2.16* représente le principe général de ces approches.

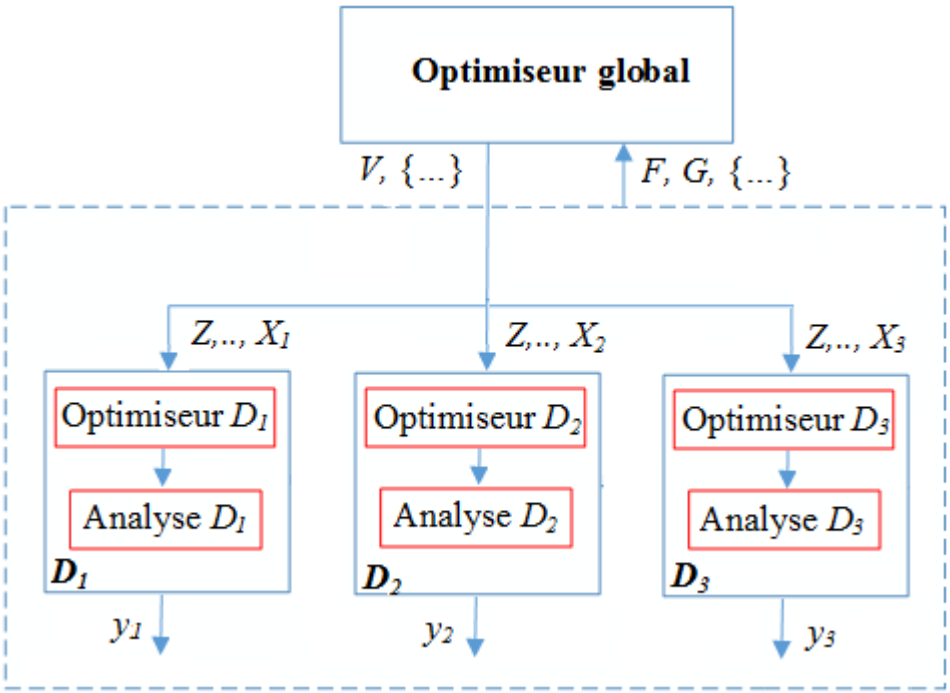


FIGURE 2.16 – Modèle général des approches multi-niveaux.

Le principe général des approches multi-niveaux est d'avoir deux types d'optimiseur, un optimiseur global du système et un optimiseur local pour chaque discipline (sous système) [70], plusieurs méthodes ont été développées dans la littérature et qui sont basées sur ce principe telles que CO, CSSO, BLISS, DIVE. Dans la suite de cette partie, nous introduisons

le principe général ainsi que les particularités de ces méthodes. Plus de détails, d'exemples et de comparaison de ces méthodes peuvent être trouvés dans [65, 66, 70–72].

A. Collaborative Optimization (CO)

Ici, le rôle de l'optimiseur global du système est de choisir les valeurs des variables partagées et ainsi que les valeurs des variables de couplage [68, 73] . Ensuite, chaque optimiseur local doit satisfaire ses contraintes internes afin de déterminer les valeurs des variables disciplinaires en se rapprochant au mieux des valeurs des variables de couplage fixées par l'optimiseur global.

B. Collaborative SubSpace Optimization (CSSO)

Dans l'approche CSSO [68, 74, 75], à chaque optimiseur local sont affectées quelques variables de couplage ainsi que les équations qui leurs sont rattachées. L'optimiseur global quant à lui est responsable du partage des contraintes dont chaque optimiseur disciplinaire a une responsabilité de satisfaire au mieux en cherchant son (ou ses) optimum.

C. Bi-Level Integrated System Synthesis (BLISS)

L'approche BLISS [68,76,77] est basée sur un modèle linéaire réduit construit à partir des valeurs des variables de sorties et de leurs gradients. L'optimisation globale du système est effectuée après l'optimisation locale de chaque discipline. Ceci étant, chaque problème ou bien sous-problème d'optimisation est formulé comme un problème d'approximation linéaire.

D. Disciplinary Interaction Variable Elimination (DIVE)

Comme pour l'approche CO, l'approche DIVE [64] définit deux niveaux d'optimisation, l'un au niveau sous-système (discipline) et l'autre au niveau système. La seule différence, c'est l'ordonnancement des différentes étapes d'optimisation. En effet, dans l'approche DIVE, on commence par les optimisations disciplinaires. Après avoir trouvé une solution, les contraintes de couplage sont satisfaites et celles qui ne sont pas vérifiées sont passées à l'optimiseur global suite à une analyse multi-disciplinaire. En fonction de ces contraintes, l'optimiseur global détermine les nouvelles valeurs des variables de couplages à imposer aux optimiseurs disciplinaires.

D'autres approches multi-niveaux d'optimisation multi-disciplinaires sont aussi développées telles que (MDOIS, MDA, DAO, etc.). Une étude détaillée de ces approches peut être trouvée dans [65, 66].

2.3.5 Synthèse et limites des approches MDO

Les méthodes MDO sont classées suivant la stratégie de résolution en deux catégories : mono-niveau, et multi-niveaux. La *Figure 2.17* permet de synthétiser les différentes approches utilisées dans chaque stratégie. Les stratégies mono-niveaux sont appliquées lorsqu'on dispose de toutes les données nécessaires à la résolution du problème telles que : les

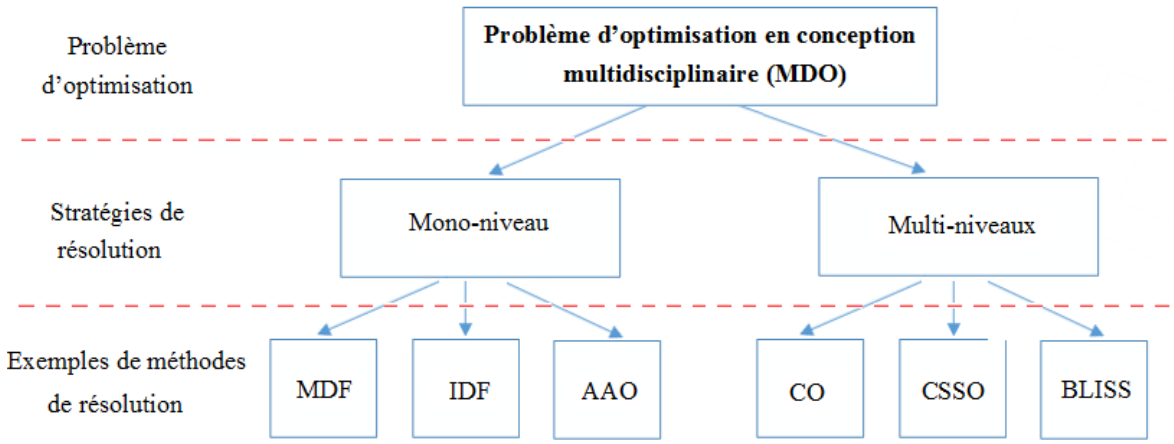


FIGURE 2.17 – Les stratégies et les méthodes de résolution d'un problème MDO.

variables, les contraintes et les objectifs. Mais, avec la présence de *boites noires* internes aux disciplines leurs applications deviennent impossibles.

La MDO propose des approches intéressantes pour la conception, car elle tient compte de problématiques métiers. En cherchant à décomposer et à intégrer des sous-problèmes d'optimisation, elle considère une partie des relations disciplinaires et surtout utilise la structure du problème pour trouver des solutions, ce que ne fait pas l'optimisation multi-objectif (MOO).

Cependant, cette considération disciplinaire se fait essentiellement à travers une décomposition hiérarchique et mathématique du problème, qui influence nécessairement la résolution et limite la compréhension globale du problème. En particulier, l'autonomie accordée aux disciplines n'est que relative, puisque la résolution se fait essentiellement par l'intégration des résultats à un niveau supérieur. Les résultats ne permettent donc pas au concepteur de découvrir l'interdépendance des contraintes au-delà du découpage préétabli, ce qui cache une partie des compromis disciplinaires et des relations entre les paramètres.

D'autres méthodes d'optimisation basées sur les problèmes de satisfaction de contraintes (CSP : Constraint Satisfaction Problem) ont été étudiées. La résolution d'un tel problème d'optimisation avec les mécanismes CSP nous permet de garder la structure globale du problème et nous donne l'occasion de trouver un optimum global par une manière déterministe [78–80].

2.4 Les problèmes de satisfaction de contraintes (CSP)

La programmation par contraintes est un paradigme de modélisation et de résolution des problèmes combinatoires de grandes tailles (problèmes de planification, problèmes d'ordonnancement, etc.). Un problème de satisfaction de contraintes CSP [81–88] est un problème défini sous une forme mathématique où l'on cherche des objets ou des états pour satisfaire un

certain nombre de propriétés ou de contraintes. L'objectif du processus de résolution est de satisfaire, à chaque itération, toutes les contraintes (un ensemble d'équations et d'inégalités) en réduisant le domaine admissible des variables. Dans ce cas, la résolution passe à l'étape suivante si et seulement si toutes les contraintes sont respectées à l'étape en cours.

2.4.1 Définition d'un problème de satisfaction de contraintes

Un problème de satisfaction de contraintes est défini par un triplet (X, D, C) tel que :

- $X = \{x_1, x_2, x_3, ..., x_n\}$ est un ensemble fini de variables que nous appelons les variables de contraintes avec n étant le nombre entier des variables dans le problème à résoudre.

- $D = \{d_1, d_2, d_3, ..., d_n\}$ est un ensemble fini de domaines de valeurs des variables de X tel que :
$$\forall\, i \in \{1, ..., n\}\ x_i \in d_i \tag{2.4}$$

- $C = \{c_1, c_2, c_3, ..., c_p\}$ est un ensemble fini de contraintes, p étant un nombre entier représentant le nombre de contraintes du problème.

$$\forall\, i \in \{1, ..., p\},\ \exists\, X_i \subseteq X\ \mid\ c_i(X_i) \tag{2.5}$$

Une contrainte est une relation mathématique (linéaire, non-linéaire, quadratique, booléenne, etc.) couvrant les valeurs d'un ensemble de variables. Les solutions d'un CSP sont des instanciations des variables satisfaisant toutes les contraintes de C.

Le processus de résolution d'un CSP peut être accéléré en terme de temps en faisant précéder la résolution par une phase de réduction des domaines des variables appelée phase de propagation de contraintes. La propriété qui sert à distinguer les valeurs qu'on garde et celles qu'on élimine est la notion de consistance (appelé aussi cohérence). Elle consiste à réduire au maximum les domaines de recherche des variables afin d'éliminer les valeurs incohérentes, c'est-à-dire les valeurs qui sont incompatibles avec l'une des contraintes du CSP.

2.4.2 Consistance

Il existe plusieurs techniques de vérification de la consistance [89, 90]. Les deux principales catégories sont :

2.4.2.1 Hull-consistance

Soient (X, D, C), un problème de satisfaction de contraintes impliquant un vecteur X de n variables, et soit $[x]$ le domaine de x.
(X,D,C) est dit Hull-consistant, si pour chaque contrainte c dans C et pour tout $i \mid (1 \leq \text{i} \leq \text{n})$, il existe deux points de $[x]$ qui satisfont c et dont les i^{mes} coordonnées sont $\underline{x_i}$ et $\overline{x_i}$, respectivement. La propriété de cohérence (Hull-consistance) réside dans la combinaison de raisonnement locale et la représentation par intervalle des domaines. Cette propriété a apporté

une amélioration décisive par rapport aux solveurs numériques traditionnels de Newton qui sont basiquement capables de contracter d'une manière globale les domaines de recherche.
Pour expliquer le processus de vérification de la consistance, nous détaillons dans cette partie un exemple sur la Hull-consistance qui contient une contrainte et deux variables. L'exemple est défini par :

$$y = x^2 \text{ avec } (x,y) \in [0\ ;\ 2] \times [0\ ;\ 2] \tag{2.6}$$

Le processus de consistance commence par :

$$x^2 \in [0\ ;\ 2]$$

Ce qui implique :

$$x \in [-1.414\ ;\ 1.414] \cap [0\ ;\ 2].$$

Et finalement,

$$x \in [0\ ;\ 1.414]$$

Le nouveau produit cartésien $(x,y) \in [0\ ;\ 1.414] \times [0\ ;\ 2]$ est Hull-consistant avec la contrainte (conforme à la cohérence) et la contrainte d'intervalle de x est réduite.
Le procédé de vérification de la Hull-consistance utilise un mécanisme de l'évaluation avant et de propagation arrière basé sur un arbre binaire. Le CSP non-Hull-consistant donné par équation (2.7) est considéré pour illustrer ce processus :

$$(x = [5\ ;\ 9], y = [3\ ;\ 8], z = [5\ ;\ 10],\ z = x + y) \tag{2.7}$$

Les *Figures 2.18* et *2.19* illustrent le mécanisme de propagation appliquée à l'exemple précédent. Le processus commence par une traversée ascendante appelée arbre d'évaluation avant. L'expression $(x+y)$ est évaluée par application de l'addition par intervalle.

$$[5\ ;\ 9] + [3\ ;\ 8] = [8\ ;\ 17]$$

La racine de l'arbre correspond à l'intersection de :

$$[5\ ;\ 10] \cap [8\ ;\ 17] = [8\ ;\ 10]$$

La propagation vers arrière est exécutée à la la fin de l'évaluation avant. La contrainte est projetée sur un arbre de parcours descendant. À partir de la racine de l'arbre, l'intervalle $[8\ ;\ 10]$ est coupé avec ses nœuds enfants. Les deux nœuds enfants deviennent $[8\ ;\ 10]$ et la Hull-consistance du domaine de z est obtenue (cohérence compatible). Ensuite, la Hull-consistance du domaine de y est calculée dans l'équation (2.8). Le même principe est appliqué pour le traitement du domaine de x dans l'équation (2.9).

$$y \in [3\ ;\ 8] \cap ([8\ ;\ 10] - [5\ ;\ 9]) = [3\ ;\ 8] \cap [-1\ ;\ 5] = [3\ ;\ 5], \tag{2.8}$$

$$x \in [5\ ;\ 9] \cap ([8\ ;\ 10] - [3\ ;\ 8]) = [5\ ;\ 9] \cap [0\ ;\ 7] = [5\ ;\ 7]. \tag{2.9}$$

La propriété de contraction est vérifiée, c'est-à-dire :

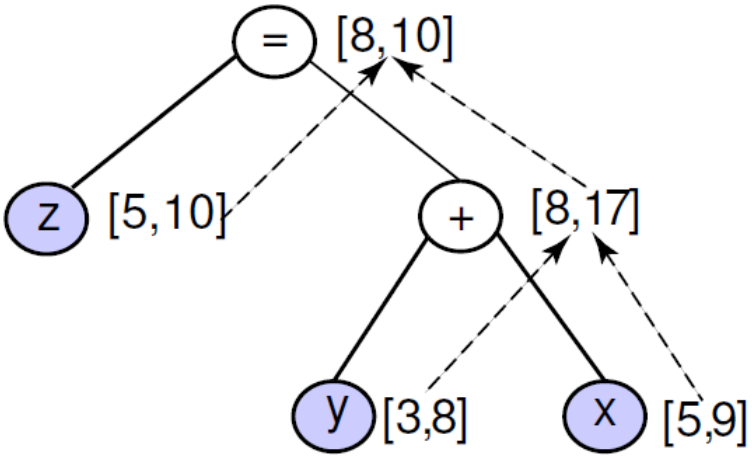


FIGURE 2.18 – Hull-consistance : évaluation avant

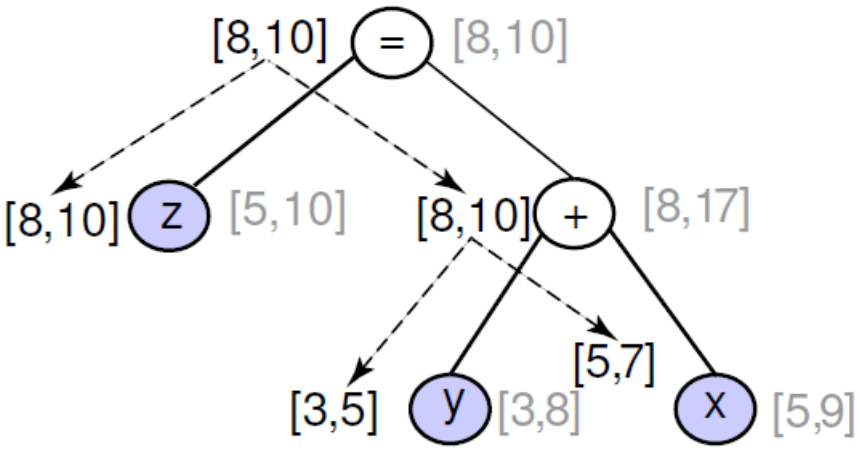


FIGURE 2.19 – Hull-consistance : propagation arrière

$$[8\ ;\ 10] \subset [5\ ;\ 10], [3\ ;\ 5] \subset [3\ ;\ 8] et [5\ ;\ 7] \subset [5\ ;\ 9]$$

Ce processus est appliqué d'une manière répétitive à chaque contrainte du CSP jusqu'à atteindre un point fixe.
Comme exemple avec deux contraintes, on considère le CSP continu suivant [3] :

$$x \in [0;10] \text{ et } y \in [-10;10] \tag{2.10}$$

Les contraintes imposées sont :

$$(C_1) : x - y = 0 \tag{2.11}$$

$$(C_2) : x + 2\,y = 3 \tag{2.12}$$

La propagation des contraintes permet de générer le processus suivant :

$$(C_1) \Rightarrow y \in [0\ ;\ 10] \tag{2.13}$$

$$(C_2) \Rightarrow x \in [0\ ;\ 10] \cap (3 - 2\ [0\ ;\ 10]), x \in [0\ ;\ 10] \cap ([-27\ ;\ 3]) \Rightarrow x \in [0\ ;\ 3] \tag{2.14}$$

$$(C_2) \Rightarrow y \in [0\ ;\ 10] \cap (\frac{3 - [0\ ;\ 3]}{2}), y \in [0\ ;\ 10] \cap (\frac{[0\ ;\ 3]}{2}) \Rightarrow y = [0\ ;\ 1.5] \tag{2.15}$$

$$(C_1) \Rightarrow x \in [0\ ;\ 3] \cap [0\ ;\ 1.5] = [0\ ;\ 1.5] \tag{2.16}$$

$$(C_2) \Rightarrow y \in [0\ ;\ 1.5] \cap (\frac{3 - [0\ ;\ 1.5]}{2}) \Rightarrow y \in [0.75\ ;\ 1.5] \tag{2.17}$$

$$(C_1) \Rightarrow x \in [0\ ;\ 1.5] \cap [0.75\ ;\ 1.5] = [0.75\ ;\ 1.5] \tag{2.18}$$

$$(C_2) \Rightarrow y \in [0.75\ ;\ 1.5] \cap (\frac{3 - [0.75\ ;\ 1.5]}{2}) \Rightarrow y \in [0.75\ ;\ 1.125] \tag{2.19}$$

$$(C_1) \Rightarrow x \in [0.75\ ;\ 1.5] \cap [0.75\ ;\ 1.125] = [0.75\ ;\ 1.125] \tag{2.20}$$

$$(C_2) \Rightarrow y \in [0.75\ ;\ 1.125] \cap (\frac{3 - [0.75\ ;\ 1.125]}{2}) \Rightarrow y \in [0.9375\ ;\ 1.125] \tag{2.21}$$

.
.
.

Le processus se poursuit jusqu'à ce que :

$$x = y = 1 \tag{2.22}$$

2.4.2.2 Box-consistance

C'est une relaxation de la Hull-consistance [87]. Le principe est de remplacer le critère de satisfaction de contraintes sur le domaine réel avec une procédure de réfutation sur le domaine de l'intervalle. Plus précisément, nous allons examiner la définition de la consistance d'arc. Une valeur $u_k \in D_k$ est consistante pour une contrainte c si,

$$\forall\, u_1 \in D_1, ..., u_k \in D_k, ..., u_n \in D_n,\ c(u_1, ..., u_k, ..., u_n) \neq 0.$$

Une déclaration équivalente à la précédente est que la gamme de c sur le domaine $(D_1 \times D_2 \times ... \times D_{k-1} \times u_k \times D_{k+1} \times ... \times D_n)$ est différente de zéro. L'idée principale est de calculer sur un ensemble de cette gamme en utilisant l'extension aux intervalles de c. Une valeur peut être éliminée si tout l'ensemble de la gamme est non nul. Alors, la définition de la box-consistance est la suivante :
Si x_k est un CSP variable et D_k est le domaine de x_k et $\forall u_k \in D_k,\ 0 \in c(D_1, ..., D_{k-1}, \{u_k\}, D_{k-1}, ..., D_n)$, alors, le domaine de x_k est Box-consistant. Le but est de trouver les valeurs extrêmes dans D_k qui sont compatibles. La mise en œuvre standard utilise une procédure de recherche dichotomique, qui exploite la propriété de monotonie de l'évaluation par intervalle.
À titre d'exemple, on considère la contrainte $y - x^2 = 0$ et les variables suivantes :

$$(x, y) \in [0\ ;\ 1] \times [0\ ;\ 6]$$

La borne gauche liée à y est cohérente (box-consistante) puisque 0 appartient à l'intervalle $0 - [0\ ;\ 1]^2 = [-1\ ;\ 0]$. Au contraire, la borne droite de y est non-consistante. Alors, le domaine de y peut être divisé pour trouver la valeur la plus cohérente à droite, comme suit :

$$[3\ ;\ 6] - [0\ ;\ 1]^2 = [2\ ;\ 6] \text{ alors } [3\ ;\ 6] \text{ est éliminé} \tag{2.23}$$

$$[0\ ;\ 3] - [0\ ;\ 1]^2 = [-1\ ;\ 3] \tag{2.24}$$

$$[1.5\ ;\ 3] - [0\ ;\ 1]^2 = [0.5\ ;\ 3] \text{ alors } [1.5\ ;\ 3] \text{ est éliminé} \tag{2.25}$$

$$[0\ ;\ 1.5] - [0\ ;\ 1]^2 = [-1\ ;\ 1.5] \tag{2.26}$$

.
.
.

$$[0\ ;\ 0.75]$$

Le dernier domaine calculé pour y est l'intervalle $[0\,;\,0.75]$. Cependant, il est clair que la recherche converge lentement. Pour accélérer la convergence, il est possible d'utiliser la méthode de Newton par intervalles [87]. L'avantage principal de la Box-consistance est la capacité de gérer le problème de l'élimination des calculs par ensemble. En effet, lorsque la variable considérée n'apparaît qu'une seule fois dans la contrainte, la Hull-consistance est équivalente à la Box-consistance et moins chère au niveau temps de calcul.

La propriété principale d'un CSP numérique dans le cas où il existe une solution d'intervalle, et qu'elle est à l'intérieur des intervalles retournés par l'algorithme de l'une de consistance (Hull- ou de Box- consistance). Dans la pratique, cette condition est nécessaire mais pas suffisante et il faut un algorithme complémentaire pour trouver les véritables intervalles de solutions.

2.4.3 Algorithmes de Branch and Prune

Les algorithmes de Branch and Prune [89] sont utilisés pour la recherche des solutions d'intervalle d'un CSP numérique comme il est montré par l'algorithme 2.1. L'objectif est d'obtenir pour chaque variable une valeur supérieure et une valeur inférieure au plus près de chaque solution du CSP numérique (X, D, C). Pour atteindre cet objectif, l'algorithme de Branch and Prune s'applique récursivement sur un opérateur de consistance (opérateur de Hull- ou de Box-consistance).

Algorithme 2.1 *Algorithme de Branch and Prune*

```
1 :  BP(CSP(X, C, D), {})
2 :  D ← Prune(C, D)
3 :  if notEmpty(D)
4 :      if OkPrecise(D)
5 :          Insert(D, L)
6 :          else
7 :          (D1, D2) ← Split(D, ChooseVariable(X))
8 :          BP(CSP(X, D1, C), L)
9 :          BP(CSP(X, D2, C), L)
10 :     end if
11 :  end if
12 :  return L
```

Le processus commence par un CSP donné (X, D, C) et la liste de solutions L est vide ($\{\emptyset\}$). À chaque étape, une variable est choisie par la fonction *ChooseVariable* et son intervalle D est divisé en deux sous-intervalles D_1 et D_2. Ensuite, l'opérateur de consistance est appliqué à la taille de chacun des deux sous-intervalles de l'ensemble des contraintes de C. Il réduit les intervalles des autres variables du CSP. Si une bonne précision est obtenue en D, c'est-à-dire une solution est atteinte, l'ensemble des intervalles résultant est ajouté à l'ensemble des intervalles de solutions, L. Dans le cas contraire, le processus continue jusqu'à ce que le fractionnement est un intervalle vide. Le test de vérification d'une bonne précision est effectué par l'intermédiaire de la fonction *OKPrecise*. À la fin de l'algorithme, le processus retourne la liste L des solutions d'intervalle du CSP numérique (X, D, C).

Les algorithmes de recherche comme les algorithmes de Branch and Prune commencent le processus en sélectionnant une variable à bissecter. L'ordre dans lequel la sélection des variables est effectuée, est défini comme l'ordre de référence des variables. Le choix d'un ordre correct peut être crucial pour avoir un processus de résolution efficace dans le cas de problèmes réels. Plusieurs heuristiques existent pour sélectionner l'ordre des variables à bissecter. Après avoir sélectionné la variable à bissecter, les algorithmes doivent sélectionner des sous-intervalles du domaine de la variable. Cette sélection peut également avoir un impact important sur la durée du processus de résolution. En pratique, l'efficacité d'un algorithme de Branch and Prune sur un problème donné est fortement liée à l'ordre des variables à découper. Cet ordre, qui dépend en général du problème, est appelé stratégie de résolution.

2.4.4 Optimisation d'un problème de satisfaction de contraintes

Le principe d'optimisation adopté pour minimiser la valeur d'une variable réelle f [89] est décrit par l'algorithme 2.2. Dans la pratique, f doit être une variable égale à une expression de contrainte représentant les critères à minimiser. Le point clé est de résoudre par dichotomie une séquence de CSP où l'ensemble des contraintes augmente d'un CSP à un autre. Chaque CSP est résolu par la procédure de Branch and Prune. À chaque étape, nous ajoutons une contrainte exprimant que le prochain CSP doit être meilleur que le précédent selon la minimisation de la variable *f*. Le processus s'arrête sur le CSP qui minimise la valeur de la variable f lorsque la précision requise ε est atteinte. Notons que la solution trouvée par l'algorithme est un optimum global sans aucune condition de différentiabilité sur la variable *f*.

Algorithme 2.2 *Algorithme de minimisation d'une valeur de fonction utilisant le CSP*

1 : **OptimCSP(**X**,** C**,** D**)**
2 : $f \in [f_{min}, f_{max}]$
3 : $CSP \leftarrow (X, C, D)$
4 : **while** $f_{max} - f_{min} > \varepsilon$
5 : $C \leftarrow C \cup \{f < \frac{f_{max}+f_{min}}{2}\}$
6 : **if** find a solution for CSP
7 : $f_{max} \leftarrow fval$
8 : **else**
9 : $C \leftarrow C - \{f < \frac{f_{max}+f_{min}}{2}\}$
10 : $f_{min} = \frac{f_{max}+f_{min}}{2}$
11 : **end if**
12 : **end while**
13 : return $[f_{min}, f_{max}]$

2.4.5 Synthèse et limites des CSP

Les approches de satisfaction de contraintes sont étudiées depuis une dizaine d'années et permettant de résoudre des problèmes combinatoires de grandes tailles. Cependant, ces approches restent majoritairement académiques et peu d'applications réelles ont été adressées. Elles présentent de nombreuses limites :

- Dans un contexte général, il n'est pas possible de décider de la cohérence des contraintes sur les nombres réels.
- La représentation des nombres réels dans les calculs numériques n'est pas exacte. Nombres à virgule flottante correspondent à un ensemble de nombres rationnels.
- Utilisation des nombres à virgule flottante peuvent conduire à des erreurs d'arrondi.

En résumant, les approches d'optimisation développées dans les sections précédentes (MOO, MDO et CSP) apparaissent comme les techniques les plus générales et les plus ef-

ficaces pour la détermination de l'ensemble optimal de Pareto avec une bonne diversité de points. Cependant, elles nécessitent un nombre important d'évaluations des fonctions coût (objectifs), ce qui rend ces approches inutilisables pour des problèmes gourmands en terme de temps de calcul. Ce nombre important d'évaluations des fonctions peut être considérablement réduit en remplaçant une grande partie de celles-ci par des approximations construites à partir d'un méta-modèle.

2.5 L'optimisation à l'aide des Méta-modèles

Le terme méta-modèle appelé aussi modèle de substitution est utilisé pour exprimer sous une forme mathématique la relation entre les variables d'entrée (ou de décision) et de sortie (les fonctions objectifs ou les contraintes) d'un modèle complexe [91]. En connaissant la valeur exacte d'une telle fonction en certain nombre de points appelés *points maîtres*, le méta-modèle permet de remplacer l'évaluation de cette fonction par une approximation. Il permet également d'effectuer une analyse de sensibilité afin d'étudier l'influence des variables d'entrée sur les variables de sortie et de donner une idée exacte sur l'allure de la fonction étudiée. Il joue ainsi un rôle important pour l'analyse des problèmes d'optimisation [92].

Plusieurs techniques ont été utilisées pour construire un méta-modèle. Dans cette partie, on présente les deux techniques les plus utilisées :

- la Méthodologie des Surfaces de Réponse polynomiale RSM (Response Surface Methodology)
- le Krigeage (Kriging)

2.5.1 La Méthodologie des Surfaces de Réponse (RSM)

La Méthodologie des Surfaces de Réponse (RSM) est considérée comme une combinaison des techniques mathématiques et statistiques nécessaires pour l'amélioration, le développement et l'optimisation des processus complexes [93]. Cette technique est largement appliquée dans le secteur industriel, particulièrement lorsque les variables d'entrée ont une grande influence sur les résultats des variables de sortie. Elle est connue sous d'autres noms, comme le *Modèle polynomial* dans [94] et le *Modèle de régression polynomiale* dans [92].
La construction du modèle RSM et les méthodes de validation sont présentées dans la suite de cette partie.

2.5.1.1 Construction du modèle RSM

La RSM consiste à représenter une fonction f comme étant la somme d'un polynôme de faible degré d'ordre un ou deux au maximum et d'un terme exprimant une erreur ε ayant une variance égal à σ^2 et une distribution normale avec une moyenne nulle $E(\varepsilon) = 0$.
Par exemple, on considère un méta-modèle polynomial d'ordre deux avec deux variables d'entrée x_1 et x_2 (appelées aussi variables de régression).

La réponse f est représentée de la façon suivante :

$$f = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_{11} x_1^2 + \alpha_{22} x_2^2 + \alpha_{12} x_1 x_2 + \varepsilon \tag{2.27}$$

On pose le changement de variable suivant :
$x_1^2 = x_3,\ x_2^2 = x_4,\ x_1 x_2 = x_5,\ \alpha_{11} = \alpha_3,\ \alpha_{22} = \alpha_4\ et\ \alpha_{12} = \alpha_5$
Alors, la relation qui exprime la sortie f en fonction des variables de régression est donnée par :

$$f = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 + \alpha_5 x_5 + \varepsilon \tag{2.28}$$

L'expression de f est appelée *modèle de régression linéaire multiple* [93] et les coefficients α_i, $i = 0, ..., 5$ sont appelés les *coefficients de régression*.
D'une façon globale, la réponse f s'écrit sous la forme suivante :

$$f = \alpha_0 + \sum_{j=1}^{n} \alpha_j x_j + \varepsilon \tag{2.29}$$

Avec n : est le nombre de variables de régression (dans ce cas, $n = 5$).
L'équation (2.29) est un modèle de régression linéaire en α quelle que soit la forme de la surface de réponse qu'il génère.
Maintenant, si l'on suppose qu'on connaît les valeurs des $m > n$ réponses de la fonction f à m valeurs différentes des variables d'entrées. Pour chaque observation i, on note f_i la valeur de la réponse et x_{ij} la j^{me} variable de régression de l'instance $X_i = (x_{ij})_{j=1,...,n}$.
En tenant compte des hypothèses que la moyenne de l'erreur est nulle $E(\varepsilon) = 0$ et que sa variance $\sigma^2 = var(\varepsilon)$, alors f_i a une moyenne égale à $E(f_i) = \alpha_0 + \sum_{j=1}^{n} \alpha_j x_{ij}$ et une variance $\sigma^2 = var(f_i)$, et par conséquent f_i s'écrit sous la forme :

$$f_i = \alpha_0 + \sum_{j=1}^{n} \alpha_j x_{ij} + \varepsilon_i,\ \ \forall i = 1, ..., m \tag{2.30}$$

Sous une écriture matricielle, l'équation (Eq (2.30)) devient :

$$Y = X\alpha + \varepsilon \tag{2.31}$$

Où :

$$Y = \begin{bmatrix} f_1 \\ f_2 \\ . \\ . \\ . \\ f_m \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & ... & x_{1n} \\ 1 & x_{21} & x_{22} & ... & x_{2n} \\ . & . & & & . \\ . & . & & & . \\ . & . & & & . \\ 1 & x_{m1} & x_{m2} & ... & x_{mn} \end{bmatrix}, \quad \alpha = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ . \\ . \\ . \\ \alpha_n \end{bmatrix} et \quad \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ . \\ . \\ . \\ \varepsilon_m \end{bmatrix}$$

La valeur approchée du vecteur des coefficients de régression, noté $\tilde{\alpha}$, peut être calculée par la minimisation de l'erreur L de la fonction des moindres carrés :

$$L = \sum_{i=1}^{n} \varepsilon_i^2 = \varepsilon^T \varepsilon = (Y - X\alpha)^T (Y - X\alpha) \tag{2.32}$$

D'où :

$$\tilde{\alpha} = (X^T X)^{-1} X^T Y \tag{2.33}$$

Une fois les coefficients de régression sont estimés, alors le méta-modèle RSM est considéré comme une fonction mathématique explicite. Cette fonction est utilisée par la suite pour déterminer la valeur approchée $\tilde{f_0}$ de la réponse f_0 en tout point X_0.
L'expression de calcul de $\tilde{f_0}$ est donnée par :

$$\tilde{f_0} = X_0^T \tilde{\alpha} \tag{2.34}$$

Avec :

$$X_0^T = [1 \;\; x_{01} \;\; x_{02} \;\; .... \;\; x_{0n}]$$

La variance de la réponse $\tilde{f_0}$ peut être déterminée par [93] :

$$var(\tilde{f_0}) = \sigma^2 X_0^T (X^T X)^{-1} X_0 \tag{2.35}$$

La variance σ^2 est calculée comme étant la *Moyenne de la Somme des Carrés de l'erreur* (*MSCE*) ou des résidus :

$$MSCE = \tilde{\sigma^2} = \frac{\sum_{i=1}^{m} (f_i - \tilde{f_i})^2}{m - (k+1)} \tag{2.36}$$

Où k est le nombre de variables ($k = n$), c.à.d. le nombre des coefficients de régression -1.

2.5.1.2 Validation du méta-modèle RSM

Afin de valider le méta-modèle RSM construit, il faut vérifier les deux aspects suivants :

- Les hypothèses proposées pour construire le méta-modèle RSM.
- La précision du méta-modèle RSM.

Validation des hypothèses proposées pour construire le méta-modèle RSM

Les hypothèses proposées pour construire un méta-modèle RSM [12, 91] sont les suivantes :

- Les réponses doivent être exprimées d'une manière additive et linéaire en fonction des variables de régression.
- $E(\varepsilon) = 0$: la moyenne de l'erreur est nulle.
- $var(\varepsilon_i) = \sigma^2$: la variance de l'erreur est constante pour toutes les instances X_i d'évaluation .
- L'erreur est distribuée normalement
- $cov(\varepsilon_i, \varepsilon_j) = 0, \forall i \neq j.$: les erreurs sont indépendantes.

Les trois premières hypothèses sont vérifiées par la courbe qui trace l'erreur standard en fonctions des réponses mesurées appelée *la courbe d'analyse des résidus* [12, 91] . En effet, si les résidus varient d'une façon aléatoire autour de zéro, alors ces hypothèses sont validées [91]. Les deux dernières hypothèses peuvent être validées, respectivement, par *la courbe normale des résidus* (la distribution normale de l'erreur) et *la courbe de la corrélation des erreurs* [91].

Validation de la précision du méta-modèle RSM

Une fois le méta-modèle RSM est construit et ses hypothèses sont vérifiées, il est très intéressant d'évaluer sa précision. La méthode de la validation croisée (cross validation) peut être utilisée pour étudier cette précision. Chaque solution *i* calculée (solution exacte) est successivement retirée du méta-modèle et le méta-modèle suivant est reconstruit en son absence. La différence entre la solution exacte f_i et celle $\tilde{f_{-i}}$ donne une idée de l'erreur de prédiction du méta-modèle [93, 95] :

$$e_{-i} = f_i - \tilde{f_{-i}} \tag{2.37}$$

Si l'erreur de prédiction, donnée par l'équation (2.37), est nulle pour tous les points considérés alors le méta-modèle est performant. Dans ce cas, les résultats de la validation croisée sont visualisés par une courbe qui représente la variation la solution approximée $\tilde{f_{-i}}$ en fonction de la solution exacte f_i (*Figure 2.20*).

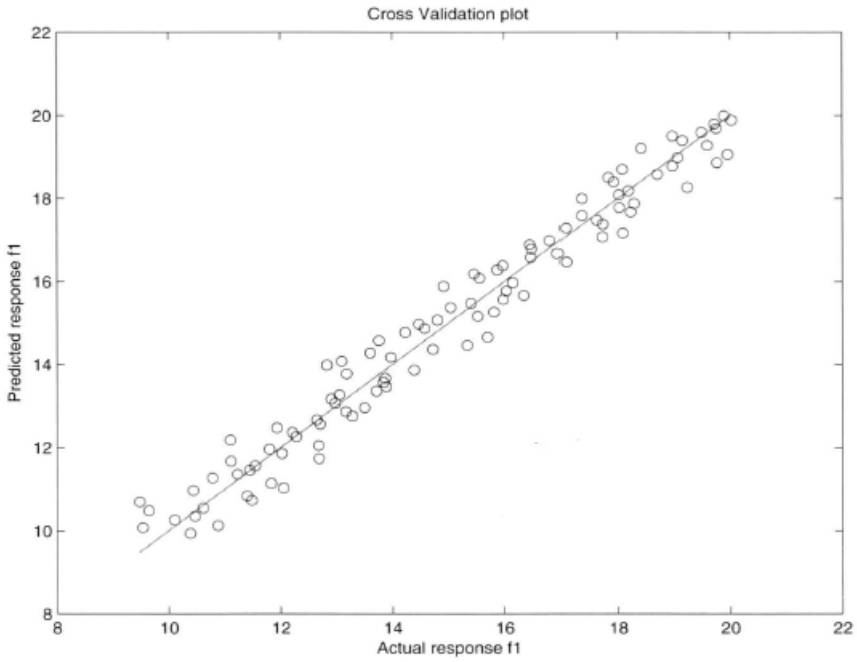


FIGURE 2.20 – Courbe de validation croisée [12]

Il existe un autre outil statistique très performant pour évaluer la précision du méta-modèle RSM, c'est la technique d'analyse de la variance (ANOVA :ANalysis Of VAriance) [95]. Dans la suite de cette section, un aperçu rapide de cette technique est donné, le modèle mathématique détaillé à été développé par [95].
L'évaluation de l'erreur est basée sur les trois termes suivants :

- SCT : la Somme des Carrés Totaux ou la somme des carrés des réponses mesurées :

$$SCT = \sum_{i=1}^{m} (f_i - \bar{f})^2 \tag{2.38}$$

Avec

$$\bar{f} = \frac{1}{m} \sum_{i=1}^{m} f_i$$

- SCR : la Somme des Carrés résultants de la Régression ou la somme des carrés des réponses prédites :

$$SRC = \sum_{i=1}^{m} (\tilde{f}_i - \bar{f})^2 \tag{2.39}$$

- SCE : la Somme des Carrés de l'Erreur ou des résidus :

$$SCE = \sum_{i=1}^{m} (\tilde{f}_i - \tilde{f}_i)^2 \tag{2.40}$$

À partir de l'analyse de la variance :

$$SCT = SCR + SCE \tag{2.41}$$

Si le méta-modèle est précis, alors la variabilité due à l'erreur est petite par rapport à celle due à la régression. Afin d'établir la précision du méta-modèle, on peut évaluer le coefficient de détermination ou coefficient de corrélation multiple R^2, donné par l'équation (2.42), qui donne une idée sur la performance de l'approximation. R^2 représente le rapport entre la SCR (la variance due à la régression) et SCT (la variance totale). Ce rapport varie entre 0 et 1 et s'il est très proche de 1, alors le méta-modèle est très performant (précis).

$$R^2 = \frac{SCR}{SCT} = 1 - \frac{SCE}{SCT} \tag{2.42}$$

Il existe un autre critère qui peut être utilisé pour évaluer la précision du méta-modèle, c'est le critère ajusté R_A^2 :

$$R_A^2 = 1 - \frac{m-1}{m-q}(1 - R^2) \tag{2.43}$$

Où

q est le nombre de coefficients de régression ($q = n + 1$). Plus R_A^2 est très proche de 1, plus le méta-modèle est performant.

On peut utiliser aussi la racine carrée de la moyenne de l'erreur quadratique calculée par la technique de la validation croisée $RMSE_{CV}$ (Cross validation Root Mean Squared Error) :

$$RMSE_{CV} = \sqrt{\sum_{i=1}^{m} \frac{(f_i - \tilde{f_{-i}})^2}{m}} \tag{2.44}$$

Dans ce cas, le méta-modèle est précis si la valeur du $RMSE_{CV}$ est très faible.

Le méta-modèle RSM est une technique d'approximation rapide, simple et bien établi. En revanche, ce méta-modèle perd de son efficacité pour des problèmes caractérisés par un nombre important de variables (n>10) et il est difficile de choisir l'ordre polynomiale à utiliser pour les résoudre [12].

2.5.2 Le Krigeage

Le mot krigeage dérive du nom de l'ingénieur Daniel Gerhardus Krige du Sud-Africain. Dans les années cinquante, Krige a proposé le principe fondamental de cette technique statistique pour trouver la dispersion de l'or à Witwatersrand [96]. Dans les années soixante, la théorie de cette méthode, appelée Krigeage a été mise en forme par le mathématicien français Georges Matheron [97]. Matheron a défini le krigeage comme étant une méthode d'interpolation spatiale d'une fonction f à partir d'un ensemble de m mesures $(X_i, f(X_i))_{i=1,\ldots,m}$.

2.5.2.1 Construction du modèle de krigeage

Le modèle du Krigeage est construit par la démarche suivante [98, 99] :
On suppose qu'au point X_i la fonction mesurée $f(X_i)$ s'écrit sous la forme :

$$f(X_i) = \sum_h \alpha_h \varphi_h(X_i) + \varepsilon_i \tag{2.45}$$

Où

- φ_h : Une fonction de régression.
- α_h : Le coefficient inconnu à estimer de la fonction de régression.
- ε_i : Une erreur distribuée normalement avec une variance égal à σ^2 et une moyenne nulle.

Si la fonction f est continue, alors son erreur ε est aussi continue. De plus, si l'on suppose qu'on a deux points X_i et X_j qui sont proches, alors les erreurs $\varepsilon(X_i)$ et $\varepsilon(X_j)$ doivent être aussi proches, c'est à dire qu'elles sont reliées ou corrélées. Cette corrélation est d'autant plus faible que les deux points X_i et X_j sont éloignés et d'autant plus importante qu'ils sont proches. Dans le cas du krigeage, nous ne supposons pas que les erreurs sont indépendantes, mais plutôt qu'elles sont liées (corrélées) à la distance entre les points [99] :

$$d(X_i, X_j) = \sum_{h=1}^{n} \Theta_h |x_i^h - x_j^h|^{p_h} \qquad (\Theta_h \geq 0, \;\; p_h \in [1,2]) \tag{2.46}$$

La corrélation entre les erreurs des points X_i et X_j est donnée par :

$$Corr[\varepsilon(X_i), \varepsilon(X_j)] = \exp(-d(X_i, X_j)) \tag{2.47}$$

À partir de cette corrélation, on peut remplacer les termes de régression dans l'équation (2.45) par un terme constant μ, appelé la moyenne du méta-modèle, et par conséquent le krigeage peut être écrit sous la forme suivante :

$$f(X_i) = \mu + \varepsilon(X_i) \tag{2.48}$$

L'équation (2.48) est appelée aussi modèle d'un processus stochastique DACE, une abréviation de titre de l'article «*Design and Analysis of Computer Experiments*», qui a popularisé ce modèle dans son application aux expériences numériques [100].
Le krigeage est caractérisé par $2n+2$ paramètres. La valeur de ces paramètres est estimée en maximisant la fonction de probabilité (Likelihood).

Soient $Y = (f_1, ..., f_m)^T$ un vecteur des m mesures, R la matrice de taille m^2 dont les composants R_{ij} sont donnés par l'équation (2.47) et 1 un vecteur unitaire de taille m.
La fonction de probabilité est donnée par :

$$\frac{1}{(2\pi)^{\frac{m}{2}}(\sigma^2)^{\frac{m}{2}}|R|^{\frac{1}{2}}} \exp[\frac{-(Y-1\mu)^T R^{-1}(Y-1\mu)}{2\sigma^2}] \tag{2.49}$$

Il est plus pratique de maximiser le log de la fonction de probabilité (*log –Likelihood*) qui élimine les termes constants :

$$-\frac{m}{2}\log(\sigma^2) - \frac{1}{2}\log(|R|) - \frac{(Y-1\mu)^T R^{-1}(Y-1\mu)}{2\sigma^2} \tag{2.50}$$

En considérant que les dérivés de l'équation (2.50) par rapport à σ^2 et μ sont nulles, il est possible de déterminer les valeurs optimales de σ^2 et μ en fonction de R :

$$\hat{\sigma}^2 = \frac{(Y-1\hat{\mu})^T R^{-1}(Y-1\hat{\mu})}{m} \tag{2.51}$$

$$\hat{\mu} = \frac{1^T R^{-1} Y}{1^T R^{-1} 1} \tag{2.52}$$

On injecte les deux équations précédentes (2.51) et (2.52) dans l'équation (2.50), on peut déterminer la fonction réduite de log de la fonction de probabilité *log-likelihood réduite*. Cette fonction s'écrit :

$$-\frac{m}{2}\log(\hat{\sigma}^2) - \frac{1}{2}\log(|R|) \tag{2.53}$$

Dans cette équation, on n'a pas tenu compte des termes constants.

La fonction précédente dépend seulement de R, et par conséquent des paramètres de corrélations $\hat{\Theta}_h$ et $\hat{p_h}$ ($h = 1, ..., n$). En pratique, la détermination de ces paramètres est obtenue par la maximisation de la fonction *log-likelihood réduite*. Une fois ces paramètres sont déterminés, les deux équations (2.51) et (2.52) sont utilisées pour calculer les deux valeurs de $\hat{\sigma}^2$ et $\hat{\mu}$.

Pour estimer la valeur approchée $\tilde{f}$ au point X^*, on ajoute ce point (X^*, $\tilde{f}$) comme une $(m+a)^{ime}$ mesure à l'ensemble des points mesurés (observés). Par la suite, la fonction de probabilité augmentée est calculée en utilisant les paramètres obtenus en maximisant la fonction de probabilité.
Soient $\tilde{Y} = (Y^T \tilde{f})$, le vecteur des fonctions mesurées, augmenté par (X^*, $\tilde{f}$) et r le vecteur de corrélation de $\varepsilon(X^*)$ avec $\varepsilon(X_i)$:

$$r = \begin{bmatrix} Corr[\varepsilon(X^*), \varepsilon(X_1)] \\ . \\ . \\ . \\ . \\ Corr[\varepsilon(X^*), \varepsilon(X_m)] \end{bmatrix} \tag{2.54}$$

La matrice de corrélation du système augmenté est donnée par :

$$\tilde{R} = \begin{bmatrix} R & r \\ r^T & 1 \end{bmatrix} \tag{2.55}$$

La fonction du système augmenté *log-Likelihood* s'écrit sous la forme suivante :

$$[\frac{-1}{\hat{\sigma}2(1-r^T R^{-1} r)}](\tilde{f}-\hat{\mu})^2 + [\frac{r^T R^{-1}(Y-1\hat{\mu})}{\hat{\sigma}^2(1-r^T R^{-1} r)}](\tilde{f}-\hat{\mu}) + termes \ \ sans \ \ \tilde{f} \tag{2.56}$$

L'expression précédente est une fonction quadratique en $\tilde{f}$. La valeur de $\tilde{f}$ qui maximise cette expression peut être déterminée en considérant que sa dérivée par rapport $\tilde{f}$ est nulle :

$$[\frac{-1}{\hat{\sigma}2(1-r^T R^{-1} r)}](\tilde{f}-\hat{\mu}) + [\frac{r^T R^{-1}(Y-1\hat{\mu})}{\hat{\sigma}^2(1-r^T R^{-1} r)}] = 0 \tag{2.57}$$

Finalement, le modèle de Krigeage est donné par :

$$\tilde{f}(X^*) = \hat{\mu} + r^T R^{-1}(Y-1\hat{\mu}) \tag{2.58}$$

L'avantage du modèle de Krigeage est sa possibilité de déterminer (estimer) l'erreur d'interpolation :

$$\hat{s}(X^*) = \sqrt{\hat{\sigma}^2[1 - r^T R^{-1} r + \frac{(1-r^T R^{-1} r)^2}{1^T R^{-1} 1}]} \tag{2.59}$$

2.5.2.2 Validation du modèle de Krigeage

Comme dans le cas de la méthode des Surfaces de Réponse, la validation du modèle de Krigeage, repose sur les deux aspects suivants :

- Validation des hypothèses proposées pour la construction du méta-modèle.
- Evaluation de l'erreur.

Validation des hypothèses du Krigeage

Les hypothèses proposées pour la construction du méta-modèle krigeage sont :

- La moyenne de l'erreur est nulle.
- L'erreur est distribuée normalement.
- La variance de l'erreur est constante, elle est égale à σ^2 .

Les trois hypothèses peuvent être validées par la méthode de la validation croisée (équation (2.37) et *Figure 2.19*), permettant de tracer les courbes de la distribution normale de l'erreur, d'analyse des résidus et de la corrélation des erreurs [12].

Validation de la précision du Krigeage

Après la validation des hypothèses de la construction du modèle de Krigeage, il est très important d'étudier sa précision. Les techniques utilisées pour évaluer la précision sont :

- La méthode de la validation croisée (comme dans le cas de la RSM).
- En utilisant le coefficient de corrélation multiple R^2(équation (2.42)), R_A^2(équation (2.43)) et $RMSE_{CV}$ (équation (2.44)). Si les deux coefficients R^2 et R_A^2 sont proches de 1 et si la $RMSE_{CV}$ est très proche de 0 (tends vers 0), alors le modèle de krigeage est précis.

Le modèle de Krigeage est une technique d'approximation déterministe et flexible. Toutefois, l'inconvénient majeur de cette technique est lié à la complexité du modèle à construire (construction coûteuse en terme de temps de calcul) et à la répartition des points mesurés (la matrice de corrélation peut être singulière dans le cas où les points maîtres sont proches [101]). Enfin, le modèle de krigeage perd de son efficacité pour un nombre élevé de variables (n>10) [12].

2.5.3 Synthèse et limites des méta-modèles

Dans cette partie, nous avons présenté les deux méta-modèles les plus employés pour la minimisation de temps de calcul liés à l'optimisation des problèmes multi-disciplinaires : la Méthodologie des Surfaces de Réponse RSM et le Krigeage. On a vu que l'utilisation de chacun de ces deux méta-modèles se heurte à des difficultés. Pour la RSM, le problème concerne l'aspect stochastique du méta-modèle, le nombre de variables du problème et l'ordre polynomial à utiliser. L'inconvénient du krigeage est lié à la répartition des points maîtres (la matrice de corrélation peut devenir singulière si les points mesurés sont proches) et à la complexité du modèle à construire (modèle coûteux en terme de coûts de calcul).

2.6 Conclusion

Les méthodes traditionnellement utilisées pour résoudre la conception préliminaire d'un système mécatronique sont basées sur des méthodes d'optimisation globales ou hiérarchiques.

En tant que méthode globale multi-objectif (MOO), les algorithmes génétiques ont montré leur efficacité à traiter ce problème, parce qu'ils sont robustes aux discontinuités et assez indépendants des caractéristiques de la fonction globale à optimiser. Le principal inconvénient de cette méthode d'optimisation globale est qu'elle ne fournit qu'une vue limitée sur les relations d'interdépendances liant l'ensemble des paramètres [102] et aucune sur les compromis disciplinaires. Aussi, dans le cas où le problème est sur-contraint le concepteur n'a aucune aide ni pour reformuler son problème, ni pour agir sur le système. De plus, la plupart de ces approches étant stochastiques, elles deviennent plus difficilement utilisables, lorsque le nombre d'objectifs et/ou de degré de liberté augmentent.

Les méthodes d'optimisation multi-disciplinaires (MDO) [103] sont des approches intéressantes, car elles cherchent à décomposer et à intégrer des sous-problèmes d'optimisation. Pour cela, elles considèrent les disciplines en tant qu'éléments agissant les uns avec les autres. Cependant, elles résolvent le problème en utilisant une décomposition et une intégration hiérarchique des sous-problèmes, ce qui influence les résultats [104] et limite la compréhension globale du problème. En particulier, les résultats n'aident pas le concepteur à découvrir l'interdépendance des contraintes au-delà du découpage préétabli, ce qui cache une partie des compromis disciplinaires et des relations entre les paramètres.

Les approches de satisfaction de contraintes, ont montré leurs intérêts dans l'optimisation des problèmes complexes tels que les problèmes de planification et d'ordonnancement. Mais la formulation particulière de ces approches les rend difficilement transposables à des problèmes continus et non-linéaires.

La détermination des solutions optimales avec l'une de ces approches nécessite un grand nombre d'évaluations des fonctions objectifs ce qui provoque l'augmentation du coût de calcul et parfois empêche la convergence des algorithmes utilisés à l'intérieur de ces approches. Une technique consiste à remplacer une grande partie de ce nombre d'évaluations par des approximations basées sur des méta-modèles afin de minimiser le coût de calcul. Cette technique perd de son efficacité pour des problèmes caractérisés par un nombre important de variables (n>10).

Afin de surmonter les limites des approches classiques d'optimisation, l'objectif du chapitre suivant est de développer une nouvelle approche basée sur les paradigmes multi-agents pour la conception optimale des systèmes mécatroniques.

Chapitre 3 : Approche multi-agents pour l'optimisation de la conception mécatronique

Chapitre 3

Approche multi-agents pour l'optimisation de la conception mécatronique

3.1 Introduction

La conception d'un système mécatronique est un problème d'optimisation multidisciplinaire et multi-objectif. Les approches d'optimisation actuellement utilisées pour l'optimisation des systèmes multi-disciplinaires sont coûteuses en temps de calcul, difficiles à mettre en œuvre et non-flexibles avec la phase de conception préliminaire et plus précisément dans le cas où les objectifs et les contraintes de conception changent fréquemment. D'où la nécessité de chercher des nouvelles techniques plus simples à mettre en œuvre, moins coûteuses et qui permettent d'adapter dynamiquement une solution suite à un changement des spécifications.

L'objectif de ce chapitre est de développer une approche multi-agents de conception qui, se basant sur les connaissances disciplinaires et par un comportement coopératif, permet de trouver collectivement une solution optimale qui satisfait les contraintes et les performances demandées.

Ce chapitre introduit, tout d'abord, les notions d'agents et les systèmes multi-agents, et détaille dans la section deux les différentes questions que soulèvent la problématique liée à la structure organisationnelle des systèmes multi-agents, en particulier : la coopération, la négociation et la communication. La section trois traite une description de l'approche multi-agents pour l'optimisation de la conception mécatronique. La section quatre présente la mise en œuvre de l'approche multi-agents avec la plateforme JADE.

3.2 Les notions d'agents et de systèmes multi-agents

3.2.1 Notion d'agent

Plusieurs définitions attribuées à la notion d'agent existent dans la littérature. Cependant, la définition du terme agent proposée par Ferber et Perrot [105] est largement utilisée dans la communauté scientifique et plus précisément dans la communauté francophone. Ferber et Perrot définissent un agent comme étant une entité physique (hard) ou logicielle (soft) située dans un environnement (réel ou virtuel) qui :

- est capable d'agir dans son environnement,
- peut communiquer avec d'autres agents de la même plateforme ou des différentes plateformes,
- est capable de percevoir son environnement d'une manière limitée,
- possède des ressources propres,
- possède des compétences
- offre des services,
- dispose d'une vue partielle de son environnement,
- peut éventuellement se reproduire,
- se comporte de façon à satisfaire ses objectifs en tenant compte des compétences et des ressources dont il dispose.

La définition qui a été donnée plus tard par Jennings et al. [106] est : « un agent est un système informatique, situé dans un environnement et qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu ».
Afin de mieux saisir la notion d'agent, nous pouvons relever plusieurs propriétés qui peuvent être attribuées aux agents [107] :

- *Autonomie* : capacité d'un agent à agir seul pour atteindre ses objectifs en ne subissant aucun contrôle sur son état interne, ni sur les actions qu'il réalise.
- *Capacité sociale* : capacité d'un agent d'interagir avec d'autres agents et probablement avec des humains grâce à des langages de communication.
- *Réactivité* : capacité d'un agent de percevoir son environnement et de répondre d'une façon opportune aux différents changements qui se produisent dans celui-ci.
- *Adaptabilité* : capacité d'un agent à apprendre et à s'améliorer avec l'expérience.
- *Pro-activité* : les agents ne doivent pas réagir seulement à des stimuli provenant de leur environnement, ils doivent être aussi capables de montrer des comportements dirigés par des objectifs internes et ceci en prenant des initiatives.
- *Continu dans le temps* : c'est un processus qui est continuellement en exécution.

3.2.2 Les systèmes multi-agents

Historiquement, les systèmes multi-agents SMA(s) se positionnent au carrefour de l'intelligence artificielle (autonomie de décision), de la programmation (logiciels) et des systèmes répartis (décentralisation). Le domaine des SMA a vu le jour pendant les années 80 sous le nom d'Intelligence Artificielle Décentralisée [108] ou SMA en Europe et d'Intelligence Artificielle Distribuée [109] aux États-Unis.

L'intelligence artificielle IA [110] est une discipline informatique pour modéliser et simuler les comportements intelligents tels que la prise de décision, la perception, l'apprentissage, la compréhension, etc. Elle s'attache à la construction de programmes informatiques pour exécuter des tâches complexes en s'appuyant sur une concentration et une centralisation de l'intelligence au sein d'un système unique. Mais, l'IA a rencontré beaucoup de difficultés dues en réalité à la nécessité d'intégrer au sein de la même base, les compétences, l'expertise et les connaissances individuelles qui communiquent et collaborent à la validation d'un objectif commun.

La nécessité de passer du comportement individuel au comportement collectif, a poussé la communauté de l'intelligence classique à se tourner vers des systèmes plus coopérants et plus autonomes afin de pouvoir résoudre les difficultés de l'IA. En effet, selon Müller [111], l'IA s'appuie sur la métaphore du penseur isolé alors que les SMA s'appuient sur la métaphore de l'organisation collective. Sur le plan technologique, ce domaine cherche à comprendre comment intégrer des ressources computationnelles, cognitives et même humaines dans un système intégré permettant de résoudre des problèmes complexes pour lesquels il n'existe que des solutions partielles et locales (diagnostic de pannes dans des réseaux de distribution, logistique, conception concourante, etc). Sur le plan scientifique, il s'agit d'essayer de comprendre comment un système complexe peut produire des compétences et performances

globales qui excèdent les compétences et les performances des entités qui y participent par le jeu des interactions entre ces entités et avec leur environnement.

Dans ce contexte, plusieurs définitions des SMA sont proposées par la communauté agent. Une des définitions les plus utilisées par la communauté francophone est celle-ci développée par Ferber [112] qui définit un SMA comme :

- Un environnement E, c'est un espace disposant généralement d'une métrique.
- Un ensemble d'objets Ob. Ces objets sont situés et passifs.
  • Situé : il est possible, à un moment donné, d'associer à un objet une position dans E.
  • Passif : un objet peut être perçu, crée, détruit et modifié par les agents.
- Un ensemble d'agents A (A $\subset$ Ob), qui sont des objets particuliers (représentent les entités actives du système).
- Une organisation O structurant les interactions entre les agents et définissant les fonctions remplies par chaque agent.
- Un ensemble d'opérations Op permettant aux agents A de percevoir, produire, consommer, transformer et manipuler des objets d'Ob dans E.

Comme il est indiqué dans la *Figure 3.1*, un SMA est composé d'un environnement d'objets passifs qui sont manipulés par les agents, représentant les entités actives du système, d'interactions entre les agents et d'opérations qui permettent les perceptions et les différentes actions.

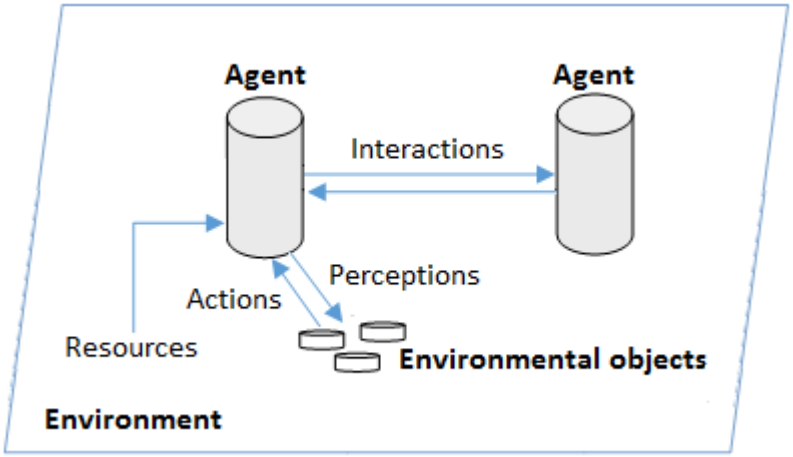


FIGURE 3.1 – Représentation d'un SMA [13].

Un SMA est un système idéal pour représenter des problèmes possédant de multiples perspectives, de multiples résolveurs et de multiples méthodes de résolution. Ce système possède les avantages traditionnels de la résolution concurrente et distribuée de problèmes tels que la fiabilité et la modularité. De plus, il hérite des bénéfices envisageables de l'IA tels que la facilité de maintenance, le traitement symbolique, la portabilité, la réutilisation

et l'avantage de faire intervenir des schémas d'interaction sophistiqués. Les types courants d'interaction incluent la coopération (travailler ensemble pour faciliter la résolution d'un problème) ; la coordination (organiser la résolution d'un problème pour éviter les interactions nuisibles et exploiter les interactions bénéfiques) ; et la négociation (parvenir à une solution finale acceptable pour toutes les parties concernées).
Malgré tous ces avantages, il existe des défis inhérents à la conception et à l'implémentation d'un SMA [113–115] :

- Comment formuler, décrire et décomposer les problèmes ?
- Comment synthétiser les résultats ?
- Comment permettre aux agents de communiquer et d'interagir ?
- Comment assurer que les agents agissent de manière cohérente
  i) en gérant les effets non-locaux de leurs décisions locales,
  ii) en évitant les interactions nuisibles,
  iii) et en prenant leurs décisions ou actions ?
- Comment trouver le meilleur compromis entre le traitement local à l'intérieur d'un agent et le traitement distribué entre plusieurs agents ?
- Comment gérer la répartition des ressources entre les agents ?
- Comment éviter les comportements chaotiques ou oscillatoires du système global ?
- Comment concevoir les méthodologies et les plates-formes de développement pour les SMA ?

Les SMA sont à l'intersection de plusieurs domaines scientifiques : intelligence artificielle, vie artificielle, informatique répartie et génie logiciel. Ils s'inspirent aussi d'études issues d'autres disciplines connexes notamment la psychologie sociale, la sociologie et les sciences cognitives. C'est ainsi qu'on les trouve parfois à la base des :

- systèmes distribués [105, 116],
- programmation orientée agents [117, 118],
- réseaux de télécommunications et protocoles de communication [116, 119, 120],
- systèmes pour la compréhension du langage naturel [121–128],
- interface personnes-machines [129, 130],
- bases de connaissances distribuées coopératives [131],
- robotique cognitive et coopération entre robots [132–134],
- applications distribuées comme le contrôle aérien, les réseaux d'énergie, le contrôle de trafic routier, le web, l'Internet, etc [135, 136].

3.3 Organisation dans les systèmes multi-agents

Dans un SMA, les agents sont caractérisés par une vision locale de leur environnement, ils sont amenés à coopérer pour atteindre l'objectif global du système. Et par conséquent, il est nécessaire de définir une structure organisationnelle à l'intérieur du SMA afin d'établir :

- la communication entre les agents,

- la coordination et la planification des tâches (actions) des agents,
- la négociation entre les agents pour la détection et la résolution des conflits.

3.3.1 Communication

Les agents communiquent entre eux pour pouvoir interagir, échanger des données et des informations et coordonner leurs activités. Ils peuvent interagir de deux manières, soit en accomplissant des actions linguistiques (en communiquant entre eux), soit en accomplissant des actions non-linguistiques qui modifient leur environnement. Les travaux antérieurs traitant la communication entre un groupe d'agents, montrent qu'il existe deux types de communication : les communications directes et indirectes [11, 13].

La communication indirecte

La communication indirecte est une démarche de communication, où le partage d'information passe généralement par l'environnement [11]. Ainsi, pour assurer la communication, les agents doivent agir sur leur environnement ; ce qui modifie la partie visible d'un ou de plusieurs agents à qui l'information doit être transférée. Ce mode de communication implique donc que les agents doivent être implémentés dans un environnement physique commun, ce qui ne permet pas la communication avec d'autres agents particuliers.

La communication directe

La communication directe est une démarche de communication, où l'agent s'adresse individuellement aux autres par l'envoi de messages [11]. Ce mode de communication complètement adapté à des applications conceptuellement distribuées. Dans ces applications, les agents utilisent des actes de langage et ont une connaissance de leur voisinage. Ce type de communication est basé sur les trois éléments suivants :

- *Le langage de communication* : nécessaire pour structurer l'échange des messages entre un groupe d'agents. Les protocoles de communication les plus connus et les plus utilisés pour le développement des SMA sont KQML (Knowledge Query and Manipulation Language) et FIPA-ACL (Foundation for Intelligent Physical Agent-Agent Communication Language) qui s'est inspiré de KQML. Ces deux protocoles sont basés sur la théorie des actes de langage [137, 138] : les messages échangés entre les agents sont considérés comme des actions ou des actes communicatifs.

- *L'ontologie* : est une spécification d'objets, de concepts et de relations dans un domaine d'intérêt. L'intérêt d'une ontologie quand elle est partagée par des agents pour représenter leur connaissance est qu'ils ont les moyens de comprendre les « mots » utilisés dans une communication [139].

- *Les supports de communication* : sont des mécanismes utilisés pour stocker et rechercher des messages. Ces mécanismes sont programmés et développés dans les plateformes multi-agents telles que JADE [140], Madkit [141], Mason [142], etc.

3.3.2 Coordination

La coordination est un processus dans lequel les agents se sont engagés en vue d'assurer une communauté d'agents individuels agissant avec cohérence et harmonie [13]. En effet, les agents ont besoin de la coordination pour empêcher les comportements chaotiques, pour être coordonnés de la même manière, puisqu'aucun agent ne possède une vue globale sur le système et parce qu'ils possèdent des capacités et expertises différentes, mais complémentaires.

Il est clair que la coordination s'avère plus facile dans les situations de routine que dans les situations non familières. En effet, dans la routine, les agents peuvent être parfaitement coordonnés, car on pourrait savoir ce qu'ils sont en train de faire et prévoir ce qu'ils vont faire. Les situations peuvent être de routine, familière ou non familière. Dans les situations familières, les agents peuvent être coordonnés selon des lois sociales, ce qui n'est pas possible dans les situations non familières.

La coordination comprend aussi l'allocation des tâches, qui consiste à affecter des responsabilités et des ressources nécessaires à la résolution de problèmes à un agent. Le créateur du système d'agents peut allouer toutes les tâches d'avance en engendrant ainsi une organisation de résolution des problèmes qui est non adaptable. Par contre, on peut avoir une allocation dynamique et flexible des tâches.

La planification fait également partie de la coordination. Pour un agent, elle constitue un processus de construction d'une séquence d'actions en tenant compte seulement des objectifs, des capacités et des contraintes environnementales. La planification a pour rôle d'éviter les conflits. Elle peut être centralisée ou distribuée.

3.3.3 Négociation

La négociation est un processus de coordination et de résolution de conflits, qui a pour objectif d'atteindre un accord final accepté par un groupe d'agents. Ce processus de négociation peut induire des échanges de données, des échanges d'informations, des relaxations des buts initiaux, des concessions mutuelles, des menaces ou des mensonges. Il existe deux types de négociation : les négociations compétitives et coopératives [13].

La négociation compétitive

La négociation compétitive est valable dans la situation où des agents de différents intérêts tentent de faire un choix de groupe avec des alternatives bien définies. Dans ce cas, les agents sont plutôt compétitifs et non coopératifs.

La négociation coopérative

La négociation coopérative est utilisée dans la situation où les agents ont un objectif unique global considéré pour le système. Dans ce cas, les agents sont dits collaboratifs, il

s'agit ainsi d'un système distribué qui a été conçu pour réaliser son objectif global.

En conclusion, pour pouvoir résoudre avec cohérence les systèmes complexes, les agents doivent communiquer entre eux, coordonner leurs activités et négocier dans les cas des conflits (situation où les agents tombent en désaccord à cause des différences entre leurs domaines d'expertise). La communication est nécessaire pour faciliter l'échange de données et des informations. La coordination est nécessaire pour l'allocation des ressources et pour déterminer la structure organisationnelle d'un groupe d'agents. La négociation est nécessaire pour la détection et la résolution des conflits.

3.4 Description de l'approche multi-agents pour l'optimisation de la conception mécatronique

Notre approche est basée sur une plateforme multi-agents qui contient des agents purement logiciels, mais qui communiquent avec des agents physiques (concepteurs, ingénieurs, etc.) et des logiciels de conception (modélisation et simulation) et d'optimisation. Pour chaque problème d'optimisation mécatronique, on a choisi que la plateforme multi-agents soit composée de deux types d'agents, des agents de conception $DA(s)$ (Design Agents en anglais) et un agent de coordination *CA* (Coordinating Agent). Ce choix est justifié par le fait que dans tout problème d'optimisation, on a des objectifs globaux du système et des objectifs locaux liés à des composants ou des sous-systèmes. L'objectif de *CA* est de participer à la recherche des solutions optimales globales du problème d'optimisation mécatronique. Alors que chaque *DA* sera affecté à une partition locale du problème d'optimisation pour participer à la recherche des solutions optimales locales.

Chaque agent logiciel (*CA* ou *DA*) est associé à une équipe d'ingénierie et une plate-forme de développement (*Figure 3.2*).

- L'équipe d'ingénierie peut être composée d'ingénieurs et des experts dans les différents domaines de l'ingénierie système.

- La plate-forme de développement peut être composée des différents outils pour la modélisation, la simulation et l'optimisation.

Chaque agent logiciel peut partager, communiquer ou coordonner avec les agents physiques à travers le modèle d'analyse, le modèle de connaissance et le modèle de coordination.

- Le modèle d'analyse est utilisé pour évaluer la conception affectée à une partition (modèle CAO, modèle des éléments finis, etc.).

- Le modèle de connaissance est basé sur des règles de modélisation et de contrôle et des informations de contexte de l'agent.

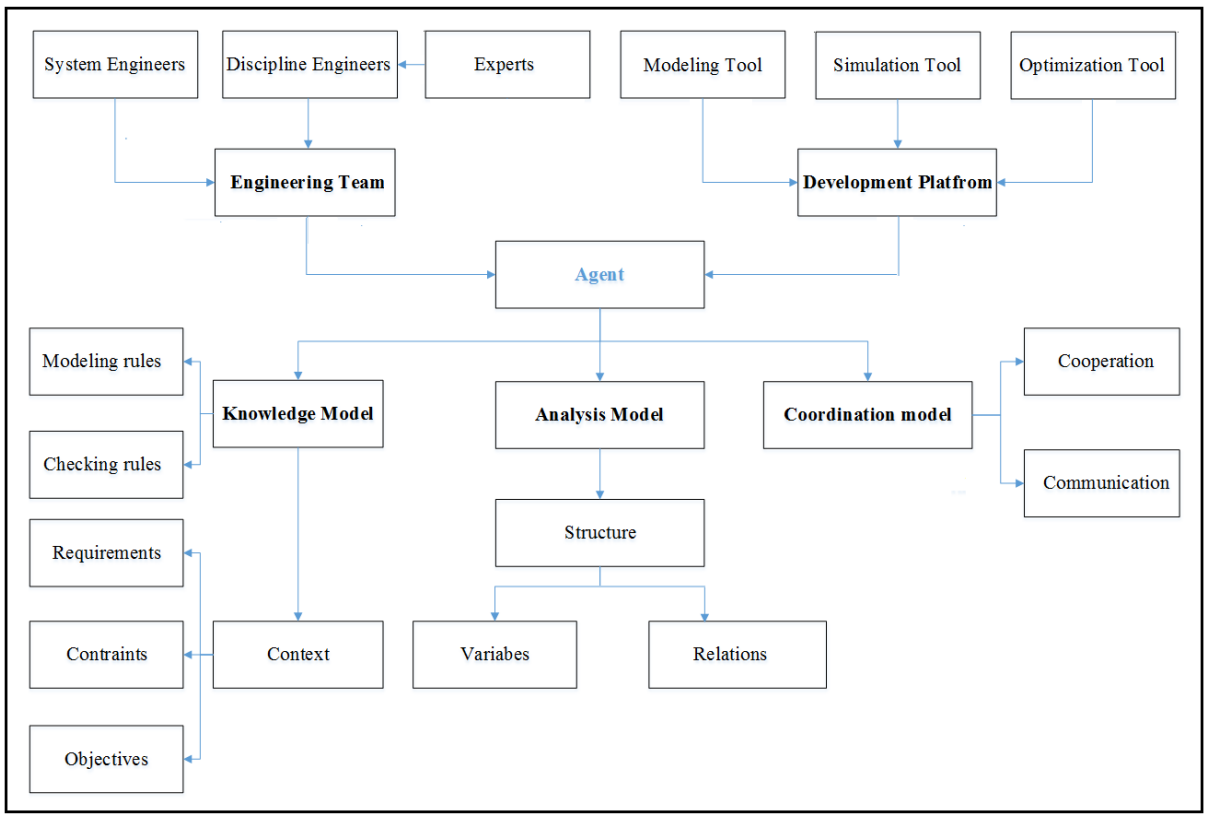


FIGURE 3.2 – La définition d'un agent pour l'optimisation de la conception mécatronique.

• Les règles de modélisation aident les concepteurs par exemple dans la définition des modèles mathématiques qui peuvent être utilisés dans la modélisation du système mécatronique.
• Les règles de contrôle sont utilisées pour vérifier la cohérence du modèle d'analyse.
• Le contexte de l'agent est basé sur des informations par rapport aux exigences de conception, les contraintes et les objectifs d'optimisation.

- Le modèle de coordination contient des informations pour la coopération et la communication entre les agents tels que les variables de couplages et les variables de conceptions partagées, etc.

3.4.1 Le processus de conception pour l'optimisation d'un système mécatronique

Pour mieux expliquer notre approche, nous détaillons dans cette partie les principales tâches réalisées au cours du processus d'optimisation pour faciliter la conception collaborative distribuée (*Figure 3.3*). Nous indiquons ainsi les étapes dans les quelles les agents *CA* et *DA* interviennent et les tâches à réaliser.

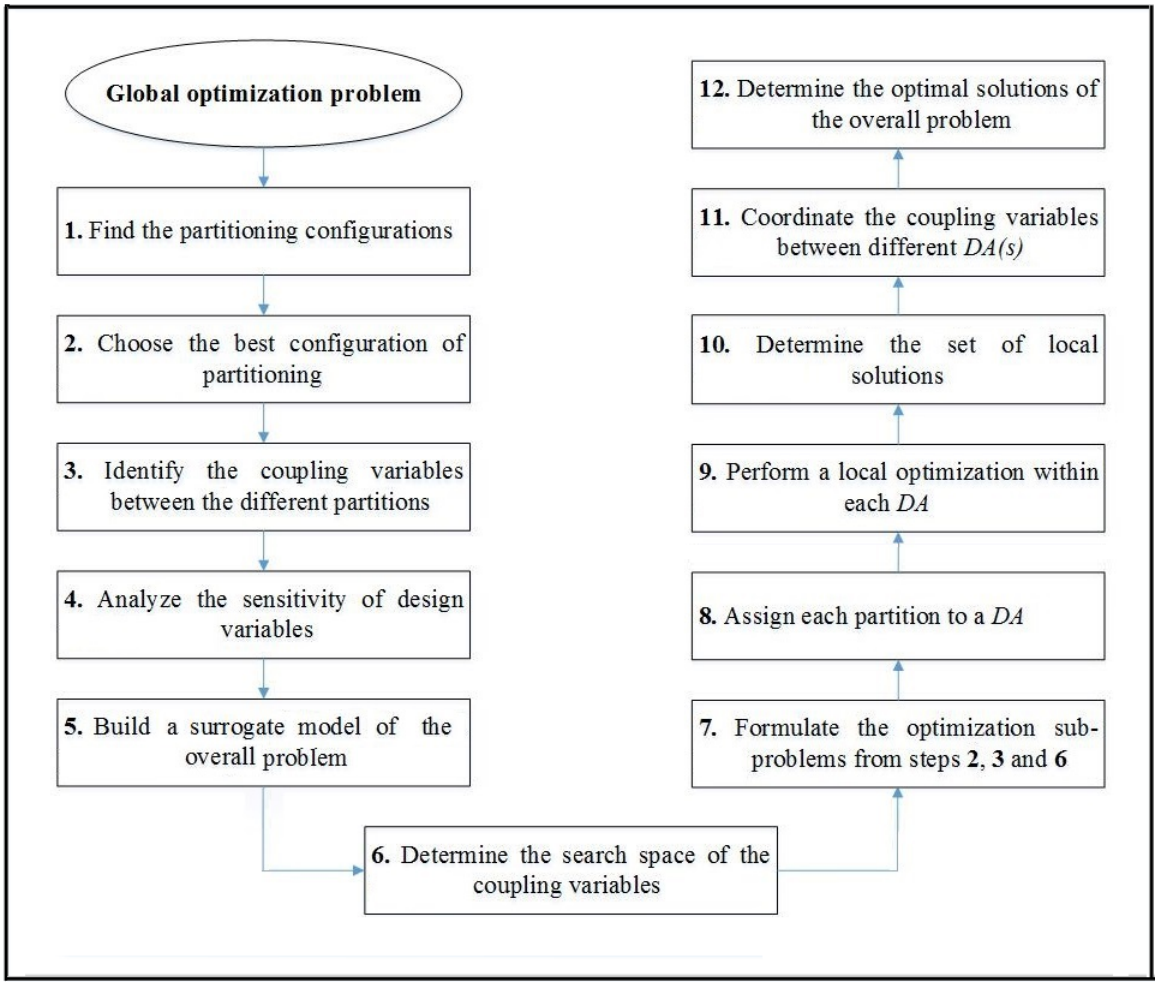


FIGURE 3.3 – Le processus de conception pour l'optimisation d'un système mécatronique.

La première tâche du processus de conception est de définir le nombre optimal de $DA(s)$ nécessaire pour l'optimisation et la répartition optimale des objectifs, des contraintes et des variables de conception entre ses différents agents. Pour effectuer cette tâche, il est nécessaire d'exécuter les étapes suivantes :

1 La première étape consiste à trouver les configurations possibles pour partitionner le problème d'optimisation. Plusieurs techniques peuvent être utilisées pour assurer cette étape. La technique la plus simple, c'est que le partitionnement soit effectué manuellement en étudiant le degré de couplage entre les fonctions objectifs, les contraintes et les variables de conception ou suivant des approches qui existent déjà dans la littérature telles que : la théorie des graphes [143], la théorie des jeux [144], la coordination de Lagrange augmentée [145, 146], etc.

2 Une fois les configurations trouvées, la deuxième étape du processus de conception est de trouver la meilleure configuration à utiliser pour résoudre le problème d'optimisa-

tion globale. Pour le choix de la meilleure configuration, nous proposons d'utiliser les deux critères suivants : le temps de calcul global C_1 et le nombre de partitions C_2. Ces deux critères doivent être minimisés.

Avec

$$C_1 = t_{global} = max\ \ t_{local} + max\ \ t_{echange} \tag{3.1}$$

- t_{global} = temps de calcul global de la configuration à choisir.
- t_{local} = temps de calcul local de chaque partition (peut être évalué pour une seule itération).
- $t_{echange}$ = temps de transfert des données (quantité d'informations à échanger) entre deux partitions (dépend du nombre de variables de couplage, nombre de variables partagées, des outils utilisés, etc.)

L'agent logiciel *CA* peut intervenir dans cette étape afin de nous aider à bien choisir la meilleure configuration. Pour cela nous proposons d'utiliser le langage SysML. En effet, on peut représenter la configuration d'optimisation avec un diagramme de blocs internes [Internal Block Diagram (IBD)]. Chaque Bloc peut représenter une partition. Les informations nécessaires pour décrire la partition, telles que les temps d'exécution, peuvent ainsi être attribuées aux propriétés du bloc qui définit la partition. Les interfaces (ports SysML) permettent de spécifier les variables d'échanges et peuvent contenir d'autres informations sur le temps d'échange à travers les ports.

Un *CA*, qui peut être décrit par un programme Java, peut lire les informations définies dans l'IBD, via un format XML. Ce qui permet d'évaluer le critère C_1.

3 La troisième étape consiste à déterminer les variables de couplages entre les partitions de la configuration choisie. Ces variables de couplages sont utilisées par la suite pour faciliter le processus de coordination entre les différentes partitions afin de trouver la conception optimale du problème global.

Une fois les variables de couplages identifiées, le concepteur vérifie si l'on dispose des intervalles de variation de ces variables. Si oui, le processus de conception passe directement à l'étape 7. Si non, le concepteur doit déterminer les variables dont les domaines de validité sont inconnus pour effectuer les étapes de 4 à 6.

4 Dans la quatrième étape, une analyse de sensibilité des variables de conception est nécessaire pour identifier celles qui n'ont pas une grande influence sur le comportement des partitions identifiées dans l'étape précédente. Ces variables doivent être éliminées afin de maximiser la précision et de minimiser le temps de calcul du méta-modèle à construire.

5 Une fois l'analyse de sensibilité est effectuée, le rôle de l'agent physique est de construire

un méta-modèle (surrogate model) ayant comme entrées les variables de conception et comme sorties les fonctions objectifs, les contraintes et les variables des couplages identifiées dans la troisième étape.

Les deux techniques utilisées ici pour construire un méta-modèle sont : la Méthode des Surfaces de Réponse polynomiale RSM (Response Surface Methodology) et le Krigeage (Kriging). Les modèles mathématiques de RSM et le krigeage sont détaillés dans le deuxième chapitre.

6 Après la construction du méta-modèle, le rôle du concepteur est de déterminer les domaines de validité des variables de couplages à partir de l'optimisation du méta-modèle.

7 L'étape suivante consiste à formuler les sous-problèmes d'optimisation des différentes partitions de la configuration choisie à partir des étapes précédentes. La formulation de chaque problème d'optimisation nécessite de :

- Choisir une partition de la configuration retenue à partir de la deuxième étape (spécifier les variables de conception, les objectifs et les contraintes locales).
- Déterminer les variables de couplages qui sont en relations avec la partition choisie en fonction de la troisième étape.
- Spécifier le domaine de validité de chaque variable de couplage, à partir de la troisième ou la sixième étape.

8 Après la formulation des sous-problèmes d'optimisation de la configuration choisie, le rôle de *CA* est d'affecter chaque partition (sous-problème) à un *DA*. En effet, à partir du fichier XML déjà construit dans la deuxième étape, *CA* peut transformer les données de chaque partition en des messages contenant les formulations d'optimisation de chaque *DA*.

Après la réception des messages de formulation de la part de *CA*, chaque *DA* informe le concepteur associé de la formulation proposée pour effectuer une optimisation locale à l'intérieur de sa partition spécifique et déterminer les solutions optimales locales par rapport à ces objectifs internes afin de satisfaire les exigences demandées (étapes 9 et 10 dans le processus de conception). L'optimisation locale de chaque *DA* nécessite de :

- Spécifier un modèle d'analyse pour évaluer les contraintes et les objectifs locaux.
- Allouer un algorithme d'optimisation multi-objectif pour déterminer les solutions locales.

Une fois les optimisations locales effectuées et les solutions générées, le rôle de l'ingénieur système, avec l'aide de *CA*, est de trouver la meilleure stratégie pour lier les variables de coordination (variables de couplages ainsi que les variables partagées) entre les différents $DA(s)$ et trouver les solutions globales qui respectent toutes les performances demandées (étapes 11 et 12). Ceci est expliqué dans la partie suivante.

3.4.2 Processus de coordination pour la détermination des solutions optimales du système global

La détermination des solutions optimales du problème global est basée sur le processus de coordination développé dans la *Figure 3.4*.

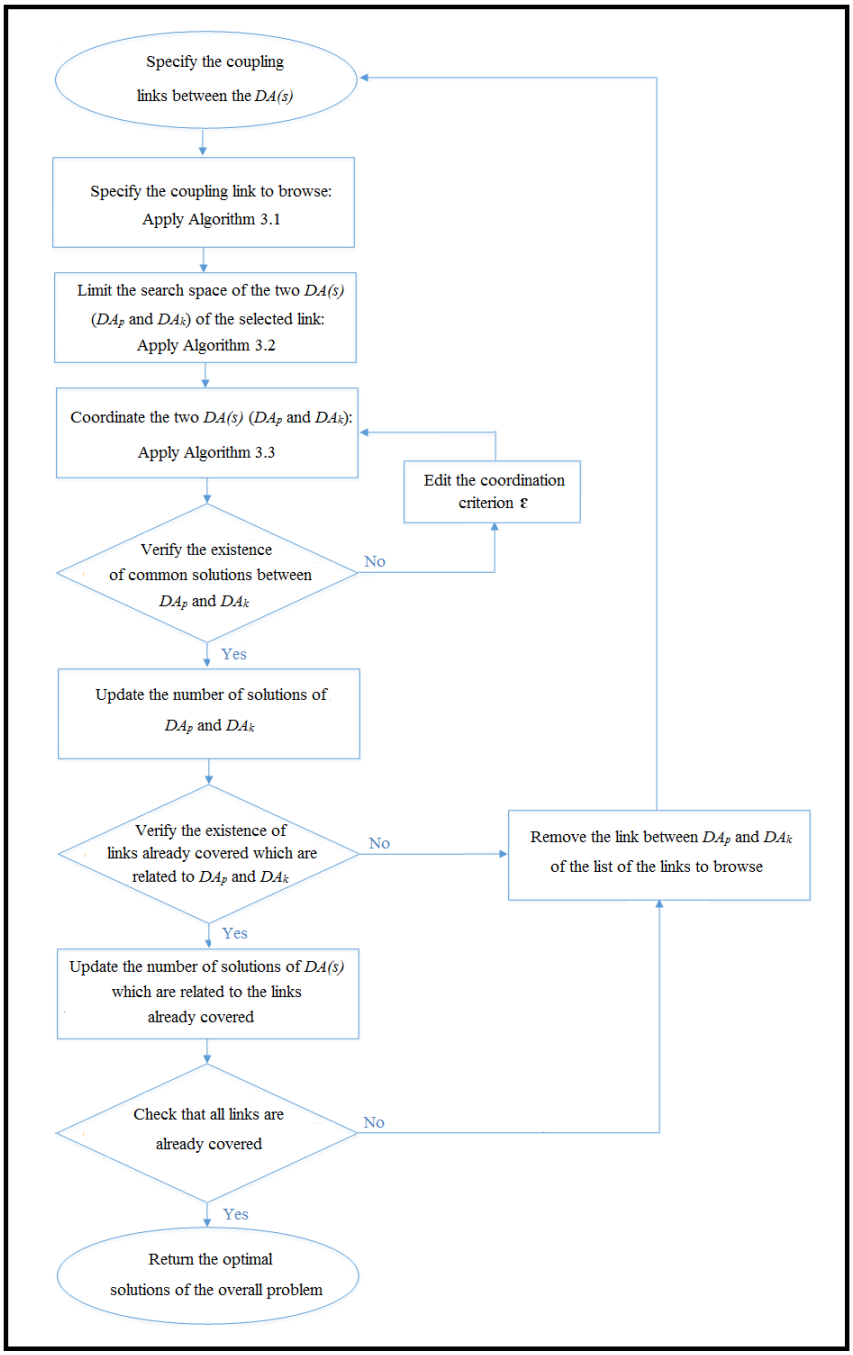


FIGURE 3.4 – Processus de coordination pour la détermination des solutions optimales du système global.

Une fois les optimisations locales effectuées, le rôle de *CA* est de spécifier l'ordre à suivre pour coordonner la recherche des solutions optimales globales. Une métrique importante qui peut être utilisée ici est le nombre d'évaluation N_i nécessaire pour déterminer toutes les solutions communes entre deux $DA(s)$ qui sont liés par un lien de couplage. L'algorithme 3.1 est appliqué pour déterminer le nombre d'évaluations minimal N_i du lien à parcourir et par conséquent de spécifier les deux $DA(s)$ (DA_p et DA_k) à coordonner.

Algorithme 3.1 *Algorithme de spécification du lien à parcourir*

1 : Let $N_{min} = N_1 = N_{1p} * N_{1k}$
2 : **for** $i = 2$ to n
3 : $Ni = N_{ip} * N_{ik}$
4 : **if** $N_i \leq N_1$ **then**
6 : $N_{min} = N_1$
7 : **end if**
8 : **end for**

- n : le nombre de liens entre les $DA(s)$,
- i : le ième lien entre les deux agents DA_p et DA_k,
- N_i : le nombre d'évaluations entre les deux agents DA_p et DA_k qui correspond au lien i,
- N_{ip}, N_{ik} : le nombre des solutions de DA_p et DA_k respectivement.

La *Figure 3.5* représente un exemple de trois $DA(s)$ qui sont liés par deux liens de couplage. Le nombre d'évaluations est égal à 50 le long du $lien_1$ et 70 le long du $lien_2$ et par conséquent le processus de coordination commence par les deux premiers agents DA_1 et DA_2 et se termine par les agents DA_1 et DA_3.

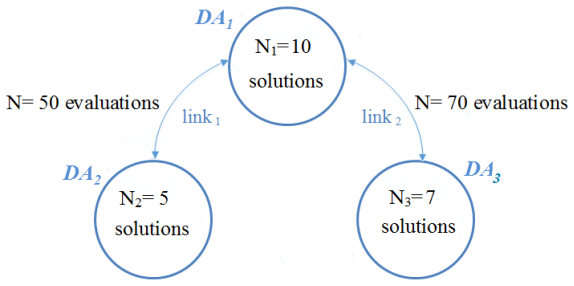


FIGURE 3.5 – Exemple de calcul du nombre d'évaluations pour déterminer les deux $DA(s)$ à coordonner.

Avant de déterminer tous les points communs entre DA_p et DA_k qui sont en relations avec le lien spécifié par l'alghorithme 3.1, il est nécessaire de limiter l'espace de recherche de ces deux $DA(s)$. Comme il est indiqué dans l'algorithme 3.2, la limitation de l'espace de recherche consiste à déterminer les bornes *min* et *max* de chaque variable de coordination pour chaque *DA* et par la suite de déterminer le *min* de *max* et *max* de *min* entre les bornes

trouvées de DA_p et DA_k.

Algorithme 3.2 *Algorithme de limitation de l'espace de recherche des solutions optimales*

Détermination du minimum et du maximum de chaque variable de coordination de chaque *DA*

1 : **for** $j = 1$ to m
2 : X_{jmin}=$X_j[1]$
3 : X_{jmax}=$X_j[1]$
4 : **for** $i = 2$ to n
5 : **if** $X_j[i] \leq X_j[1]$ **then**
6 : $X_{jmin} = X_j[i]$
7 : **else**
8 : $X_{jmax} = X_j[i]$
9 : **end if**
10 : $X_j = [X_{jmin}, X_{jmax}]$
11 : **end for**
12 : **end for**

Détermination du minimum des maxima et du maximum des minima de chaque variable de coordination de DA_p et DA_k

13 : Let $X^p = [X^p_{jmin}, X^p_{jmax}]_{j=1,\ldots,m}$ and $X^k = [X^k_{jmin}, X^k_{jmax}]_{j=1,\ldots,m}$
14 : **for** $j = 1$ to m
15 : $X^c_{jmin} = max(X^p_{jmin}, X^k_{jmin})$
16 : $X^c_{jmax} = min(X^p_{jmax}, X^k_{jmax})$
17 : $X^c_j = [X^c_{jmin}, X^c_{jmax}]$
18 : **end for**
19 : return $X^c = X^c_{j=1,..,m}$

- j : l'indice de la jème variable de coordination,
- i : la ième solution sur le front de Pareto,
- m : le nombre de variables de coordination entre deux agents (variables de couplage et variables partagées),
- n : le nombre de points optimal (solutions) le long du front de Pareto,
- X_j : l'espace de recherche de la jème variable de coordination le long du front de Pareto,
- X_{jmin} : la valeur minimale de la jème variable de coordination,
- X_{jmax} : la valeur maximale de la jème variable de coordination,
- $X_j[i]$: la valeur de la jème variable de coordination de la ième solution,
- X^p, X^k : les vecteurs de variation des variables de coordination le long des fronts de Pareto de DA_p et DA_k respectivement,
- X^p_{jmin}, X^k_{jmin} : les bornes inférieures de la jème variable de coordination le long des fronts de DA_p et DA_k respectivement,
- X^p_{jmax}, X^k_{jmax} : les bornes supérieures de la jème variable de coordination le long des fronts de DA_p et DA_k respectivement,
- X^c_{jmin} : le maximum des deux valeurs minimales de la jème variable de coordination le

long des deux fronts,
- X^c_{jmax} : le minimum des deux valeurs maximales de la jème variable de coordination le long des deux fronts,
- X^c : vecteur de coordination (vecteur de limitation de l'espace de recherche des variables de coordination).

Après la limitation de l'espace de recherche, le *CA* cherche toutes les solutions communes entre DA_p et DA_k et met à jour le nombre des solutions de chaque *DA* par l'implémentation de l'algorithme 3.3. Une fois les solutions communes entre DA_p et DA_k déterminées, *CA* vérifie s'il existe des liens déjà parcourus qui sont en relation directe ou indirecte avec DA_p et DA_k. Le nombre de solutions des $DA(s)$ qui sont liés aux liens déjà parcourus doit être aussi mis à jour et le lien entre DA_p et DA_k doit être supprimé de la liste des liens à parcourir pour commencer du nouveau le processus de coordination.

Algorithme 3.3 *Algorithme de détermination des solutions optimales entre* DA_p *et* DA_k

```
1 :  L = {}
2 :  for i = 1 to n
3 :       for j = 1 to m
4 :            if ||X_p[i] − X_k[j]||^2 ≤ ε then
5 :            X_c = (X_c[i], X_c[j]) ← (X_p[i], X_k[j])
6 :            end if
7 :            return X_c
8 :       end for
9 :  end for
10 :  return L = {X_c}
```

- n, m : le nombre de solutions optimales le long des fronts de Pareto de DA_p et DA_k après la limitation de l'espace de recherche,
- $X_p[i]$: le vecteur des variables de coordination de la ième point du front de DA_p,
- $X_k[j]$: le vecteur des variables de coordination de la jème point du front DA_k,
- $X_c[i]$: les valeurs des variables de conception du point i,
- $X_c[j]$: les valeurs des variables de conception du point j,
- X_c : la solution optimale du problème global,
- L : la liste des solutions optimales,
- ε : le critère de coordination.

La *Figure 3.6* représente un problème composé de cinq $DA(s)$ qui sont liés par quatre liens de couplage. Supposons que les liens 2, 3 et 4 sont déjà parcourus et que le nombre de solutions est égal à 7 pour chaque *DA* qui sont en relations avec ces liens (c-à-d. DA_2, DA_3, DA_4 et DA_5). Donc il reste un seul lien à parcourir, le $lien_1$ entre DA_1 et DA_2.

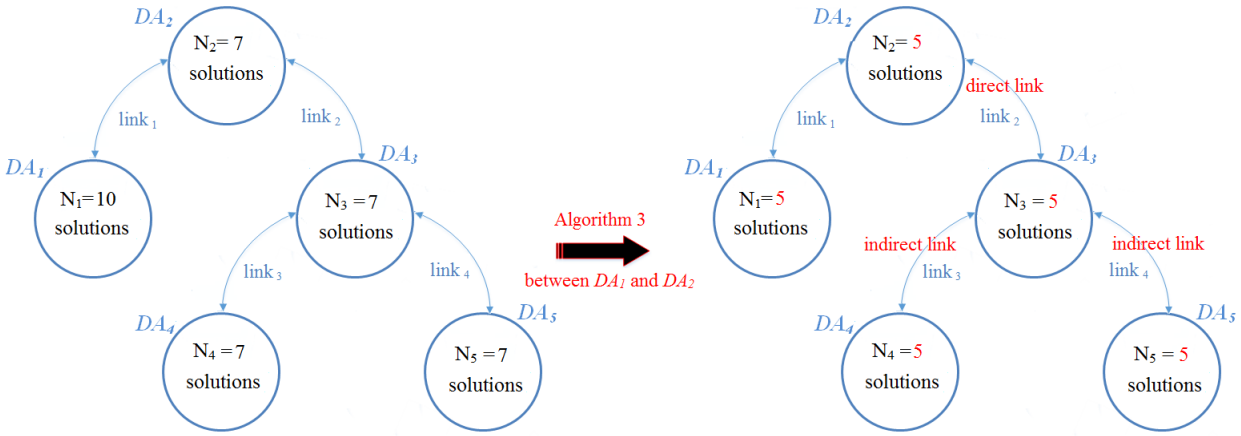


FIGURE 3.6 – Exemple de coordination avec les deux types de lien : direct et indirect.

Supposons qu'après l'implémentation de l'algorithme 3.3 le nombre des solutions communes entre DA_1 et DA_2 devient égal à 5. Dans ce cas, le $lien_2$ est en relation directe avec le $lien_1$ et les liens 4 et 5 sont en relation indirecte avec le $lien_1$ (à travers le $lien_2$) et par conséquent le nombre de solutions de DA_3, DA_4 et DA_5 devient aussi égal à 5.

Ce processus est appliqué pour tous les $DA(s)$ qui sont liés par des liens de couplage. Le processus s'arrête lorsque la liste des liens à parcourir est vide et dans ce cas le nombre de solutions global est égal au nombre de solutions communes entre les deux $DA(s)$ qui sont liés par le dernier lien parcouru.

3.5 La mise en œuvre des technologies de système multi-agents

3.5.1 Plateformes de développement des SMA

De nombreuses plates-formes [1] existent pour le développement des systèmes multi-agents. Le choix d'une plate-forme dépend de sa facilité de mise en œuvre des différents types (modèles) d'agent et de sa facilité de prise en main. En effet, les caractéristiques internes des agents sont plus ou moins faciles à mettre en œuvre suivant la plate-forme de développement utilisée [147]. Parmi les plateformes les plus connues, on trouve :

- CORMAS : (COmmon Resources Multi-Agent System) : est un framework de développement de SMA, open-source et basé sur le langage de programmation orientée objet SmallTalk. Il est centré sur des problématiques de recherche en sciences du développement et de négociation entre acteurs.

1. http ://fr.slideshare.net/mohamedyoussfi9/systmes-multi-agents-concepts-et-mise-en-oeuvre-avec-le-middleware-jade

- DoMIS : est un outil permettant la conception de SMA (orientés « pilotage opérationnel des systèmes complexes »). Il est utilisé pour l'analyse décisionnelle des systèmes complexes.

- JACK : est un langage de programmation et un environnement de développement pour les agents cognitifs, développé par la société Agent Software comme une extension orientée agent du langage Java.

- Jadex : est une plate-forme agent développée en Java par l'université de Hambourg qui se veut modulaire, compatible avec de nombreux standards et capable de développer des agents.

- Janus : est une plateforme multi-agents modulaire écrite en Java. Elle permet de créer des SMA avec ou sans une approche organisationnelle basée sur le modèle Capacité-Rôle-Interaction-Organisation (CRIO).

- Madkit : est une plateforme multi-agents modulaire écrite en Java et construite autour du modèle organisationnel Agent/Groupe/Rôle.

- JADE : (Java Agent DEvelopment) est un framework de développement de SMA, open-source et basé sur le langage Java. Il offre en particulier un support avancé de la norme FIPA-ACL, ainsi que des outils de validation syntaxique des messages entre les agents basés sur les ontologies.
- etc.

3.5.2 Mise en œuvre des agents en utilisant la plateforme JADE

Dans notre cas, nous avons choisi la plate-forme JADE [140] comme un environnement de développement de SMA. Ce cadre est fourni gratuitement par TILabs [2] et s'exécute entièrement sur l'environnement d'exécution Java.

JADE est fondée sur la spécification de FIPA [3]. Elle comprend tous les composants qui permettent la gestion de la dite plate-forme selon FIPA. Ces composants sont : Agent Management References Model (AMRF), Message Transport System (MTS), Agent Managment System (AMS) et Directory Facilitator (DF).

Une application JADE est une plateforme déployée sur une ou plusieurs machines. Elle héberge un ensemble d'agents, identifiés de manière unique, pouvant communiquer de manière bidirectionnelle avec les autres agents. Chaque agent s'exécute dans un conteneur secondaire (Container) qui lui fournit son environnement d'exécution ; il peut migrer à l'intérieur de la plateforme. Chaque plate-forme doit avoir un conteneur principal (mainContainer)

2. http ://jade.tilab.com/

3. Foundation for Intelligent Physical Agents : une organisation en 1996 dont l'objectif est de produire des standards pour l'interopération d'agents logiciels hétérogènes

qui enregistre les autres conteneurs.

La *Figure 3.7* représente une application JADE développée sur 3 machines. Les agents CA, DA_1 et DA_2 sont déployés respectivement dans le mainContainer, container 1 et container 2.

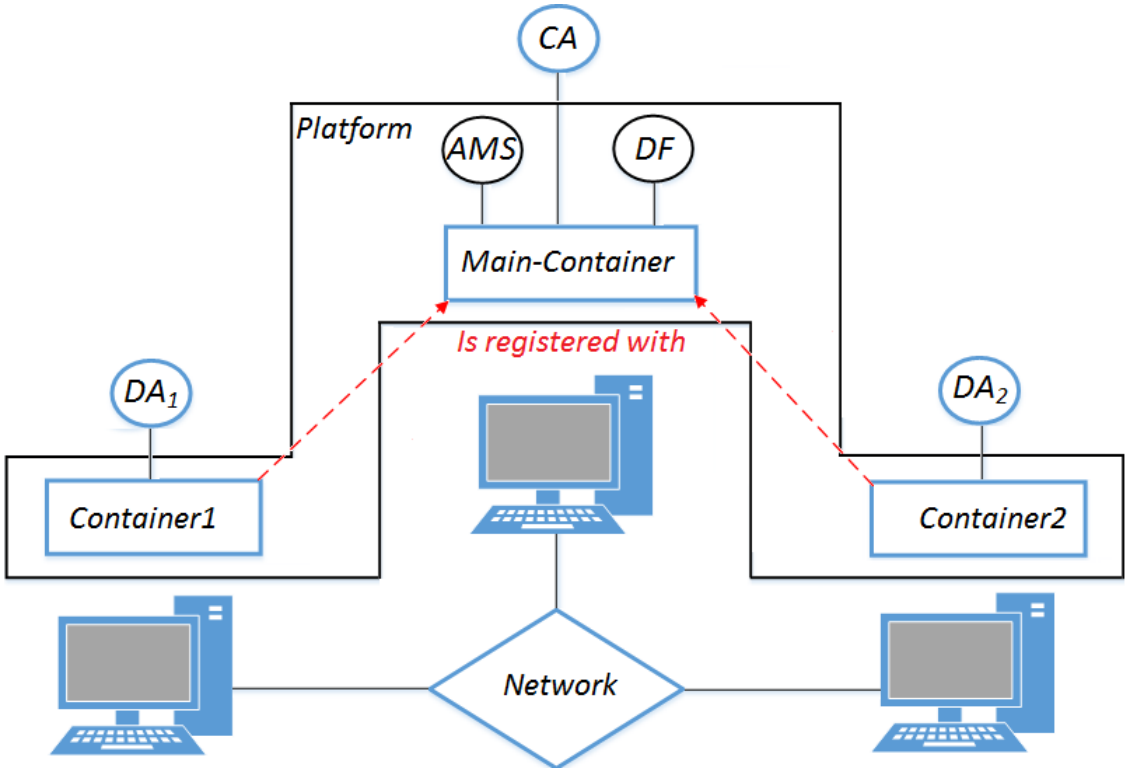


FIGURE 3.7 – Exemple d'une application JADE deployée sur trois machines

Le cadre du système de conception à base d'agents est illustré à la *Figure 3.8*. Agent Platform (PA) est l'environnement logiciel dans lequel plusieurs agents peuvent être créés, déployés, enregistrés, situés et communiqués dans JADE. Chaque agent doit être inscrit sur les descriptions désignées par AMRF (fournit une infrastructure standard avec des fonctions bien définies) pour fournir des services spécifiques. AMS est un répertoire de registre qui attribue un identifiant unique pour chaque agent (Service de Pages Blanches : référence automatiquement les agents suivant leur nom dès leur entrée dans le système). DF est un répertoire de registre qui offre le service de pages jaunes (référence à leur demande les agents suivant leur(s) service(s)). Les services d'un agent peuvent être interrogés par les informations enregistrées dans le DF. MTS coordonne la communication entre les agents de la même AP ou différentes AP(s). *CA* réside dans un conteneur principal, et chaque $DA(s)$ est déployé dans un conteneur secondaire. Ces conteneurs peuvent être placés séparément dans différentes machines et distingués par les adresses IP.

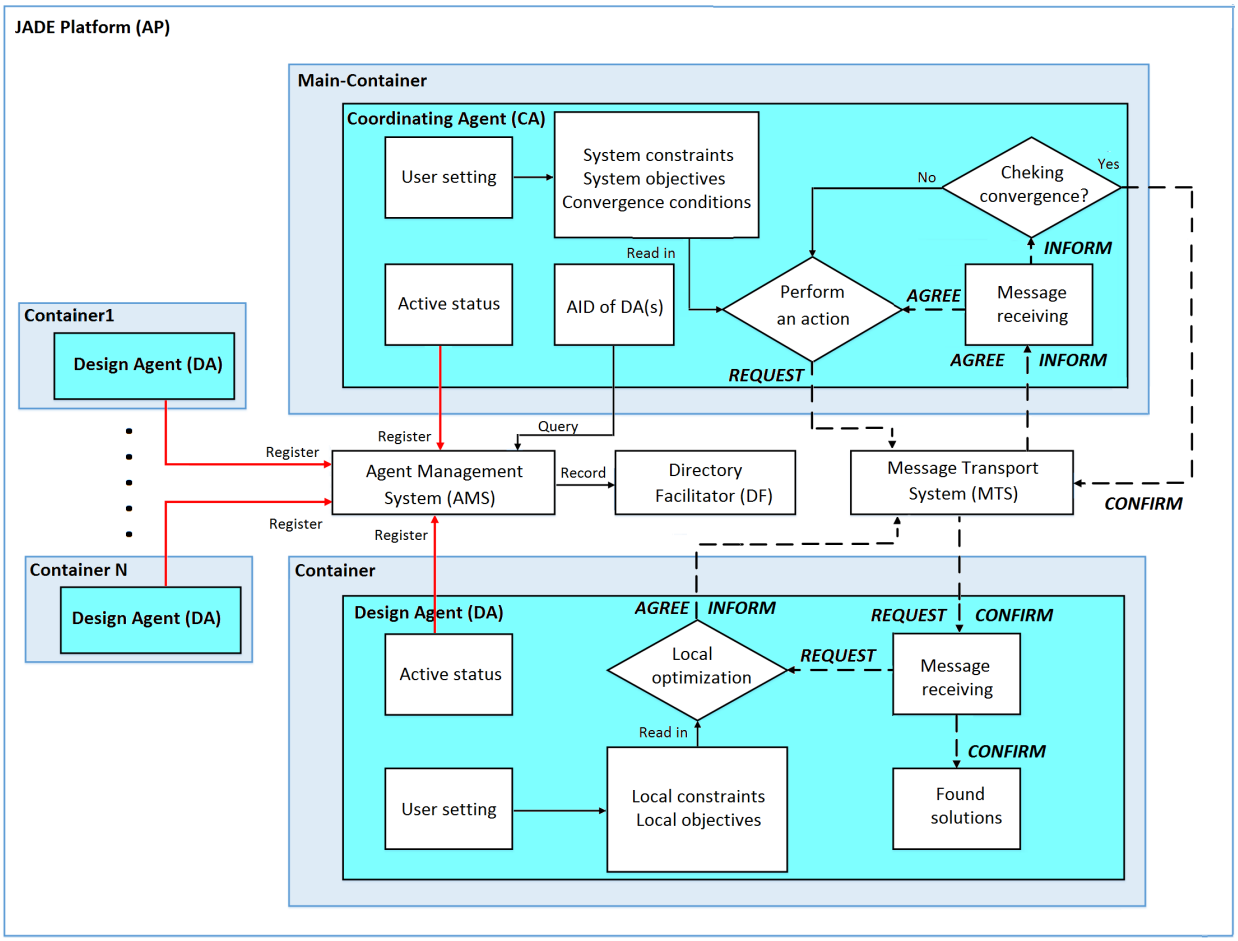


FIGURE 3.8 – La mise en œuvre des agents avec la plate-forme JADE

3.5.3 Modèle d'exécution d'un agent sur JADE

Pour qu'un agent JADE exécute un comportement, nous avons tout d'abord besoin de définir ces tâches (*Figure 3.9*) :

- Initialisation : Implémenter la méthode setup()[obligatoire] pour ajouter une liste de comportements actifs.

- Sélection : Sélectionner le comportement b à exécuter dans la liste des comportements actifs de l'agent.

- Exécution : Exécuter le comportement b. Chaque comportement doit implémenter les deux méthodes suivantes :

  • Exécution de la méthode action() qui désigne les opérations à exécuter par le comportement b.

• Exécution de la méthode done() pour tester la fin du comportement.
Si le comportement est terminé, retrait de la liste de comportements actifs, sinon insertion du comportement dans la liste des comportements actifs.

- Implémentation : Implémenter la méthode takeDown() [optionnelle] lors de la fin d'exécution de l'agent pour demander au DF de supprimer les services qui ont été inscrits par l'agent.

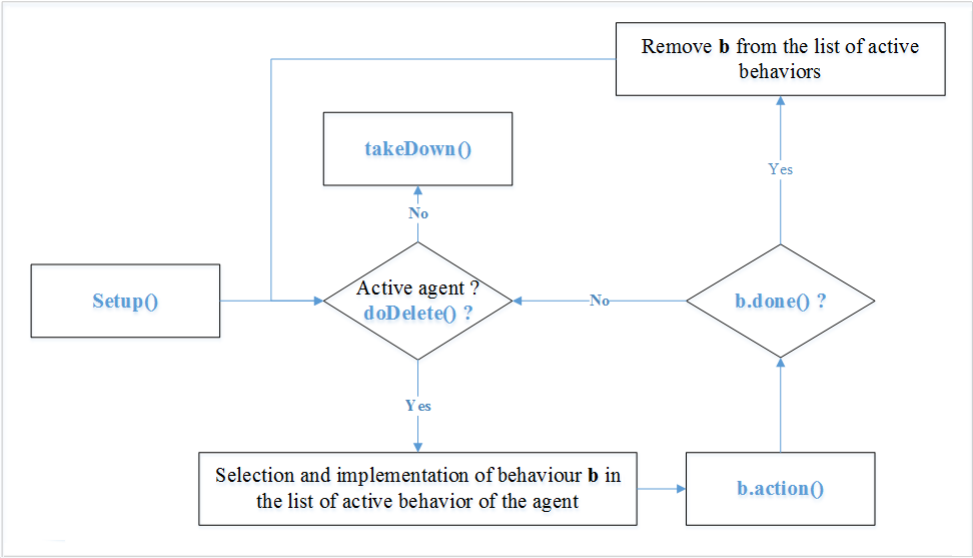


FIGURE 3.9 – Modèle d'exécution d'un agent sur JADE

3.5.4 Cycle de vie d'un agent

Durant l'exécution d'un agent, son état peut être modifié :

- INITIATED : l'agent est lancé mais non enregistré auprès de l'AMS, aucun nom, aucune adresse.

- ACTIVE : l'agent est répertorié auprès de l'AMS et peut accéder aux services.

- SUSPENDED : tous les comportements de l'agent sont suspendus.

- TRANSIT : l'agent migre vers une autre plateforme.
- WAITING : tous les comportements de l'agent sont temporairement interrompus.

- DELETED : l'exécution de l'agent est terminée et n'est plus répertoriée au sein de l'AMS.

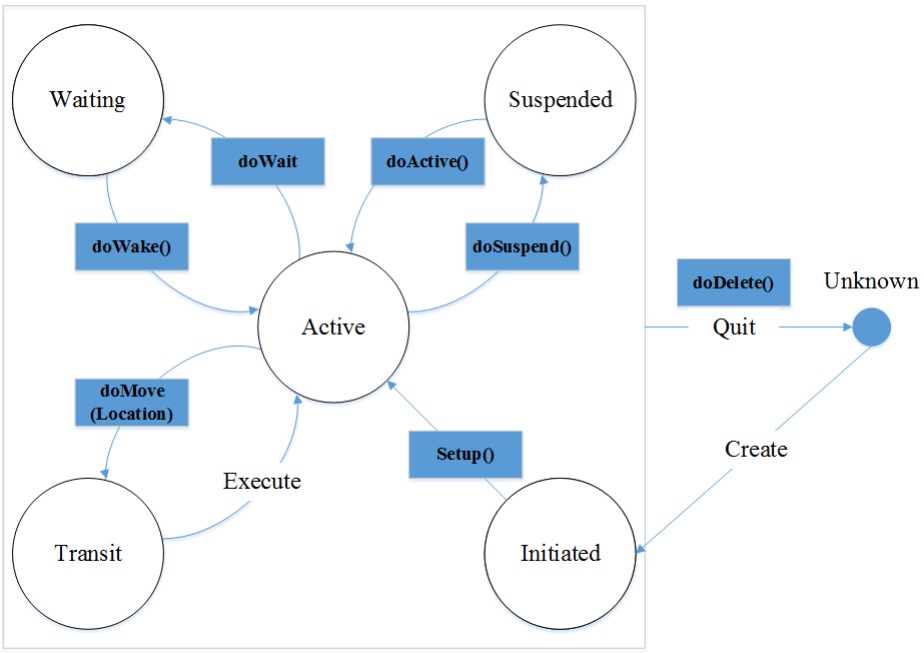


FIGURE 3.10 – Cycle de vie d'un agent

3.5.5 Le processus de coordination : Workflow de communication

Les protocoles de coordination spécifient l'enchaînement des messages entre les agents. Ils définissent les messages à envoyer en précisant : le performatif, le contenu, le(s) destinataire(s) et établissent les réponses à ses messages. Le processus de coordination proposé est réalisé dans le système à base d'agents de la manière suivante :

- 1. Initialisation : tous les $DA(s)$ nécessaires à la coordination sont enregistrés dans AMS. Les informations de chaque *DA* sont stockées dans DF.

- 2. Spécification : les objectifs globaux, les contraintes globales et les conditions de convergence du système sont enregistrés dans *CA*.

- 3. Information : *CA* précise les AIDs des DA(s) impliqués dans la coordination en interrogeant AMS et invite tous les $DA(s)$ à rejoindre le processus de coordination.

- 4. Coordination : le processus de coordination commence pour la détermination des solutions optimales qui respectent toutes les exigences demandées des différents $DA(s)$ (*Figure 3.11* représente un workflow de communication entre deux $DA(s)$) :

  4.1. *CA* informe les $DA(s)$ à travers MTS que le workflow de communication commence. *CA* envoie une demande en envoyant *ACLMessage.Request* à chaque *DA* pour effectuer une optimisation locale et le contenu du message est la formulation du sous-problème d'optimisation de l'agent correspondant.

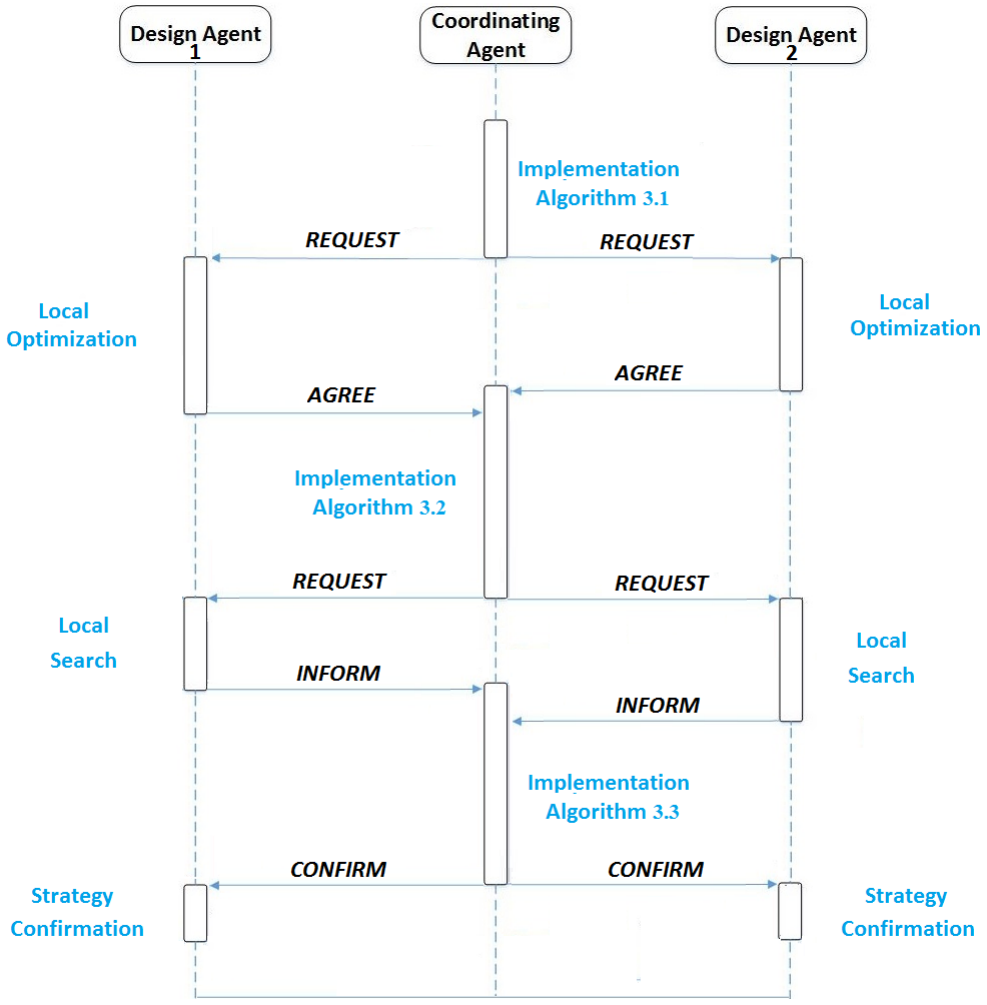


FIGURE 3.11 – Workflow de communication entre *CA* et *DA*

4.2. Chaque *DA* reçoit la demande pour effectuer une optimisation locale et déterminer les solutions optimales par rapport à ces objectifs internes afin de satisfaire les exigences demandées. Une fois l'optimisation est effectuée et le front de Pareto est généré *ACLMessage.AGREE* est envoyé à *CA*. Le contenu du message est l'intervalle de variation de l'ensemble des solutions trouvé.

4.3. *CA* attend l'arrivée de tous les messages des $DA(s)$. *CA* limite l'espace de recherche des solutions optimales par l'implémentation de l'algorithme 3.2. Après la limitation de l'espace de recherche, *CA* demande de nouveau à chaque *DA*, en envoyant *ACLMessage.REQUEST*, de déterminer la liste des points qui respecte l'espace demandé.

4.4. Chaque *DA* reçoit la demande pour effectuer une recherche locale et détermine la liste des points demandés. Une fois la recherche est terminée *ACLMessage.INFORM* est envoyé à *CA* et le contenu du message est l'ensemble des solutions trouvé.

4.5. *CA* attend l'arrivée de tous les messages des $DA(s)$. *CA* coordonne les solutions envoyées par les différents $DA(s)$ en implémentant l'algorithme 3.3 afin de trouver les solutions communes entre ces derniers. Tous les $DA(s)$ accomplissent le workflow de communication en recevant *ACLMessage.CONFIRM* de *CA* pour les informer de la confirmation de la stratégie de coordination.

3.6 Conclusion

Dans ce chapitre, nous avons développé une nouvelle approche basée sur les technologies multi-agents pour la conception optimale des systèmes mécatroniques. Cette approche est composée par des agents purement informatiques qui participent à la recherche des solutions optimales (locales ou globales) et qui communiquent avec des agents physiques qui sont en relation directe avec les logiciels de conception. Elle permet aux agents de suivre un processus de conception afin de faciliter la conception collaborative des systèmes distribués.

Dans le chapitre suivant, nous validerons cette approche à travers un cas de test pour la conception préliminaire d'un véhicule électrique.

Chapitre 4 : Étude de cas : conception préliminaire d'un véhicule électrique

Chapitre 4

Étude de cas : conception préliminaire d'un véhicule électrique

4.1 Introduction

Avec le développement considérable de l'univers de la mécanique, les types de moyens de transport sont de plus en plus nombreux. La voiture demeure toutefois le moyen de locomotion le plus convoité. On peut distinguer deux grands types de véhicule : les voitures habituelles qui sont dotées d'un moteur à explosion et celles qui sont électriques [148].

La voiture électrique n'a pas encore supplanté la voiture à essence. Ce type de véhicule présente différents inconvénients [148, 149], notamment en ce qui concerne le coût, l'autonomie, l'approvisionnement et la vitesse. En effet, la production des différentes pièces de la voiture étant beaucoup plus compliquée, le coût devient obligatoirement plus élevé et sa batterie ne disposant que de quelques heures d'autonomie, une recharge doit être effectuée environ tous les 100 kilomètres.

Ces inconvénients ne sont pas pour autant définitifs. L'augmentation du prix du carburant et les normes de plus en plus strictes d'émissions exigent que les nouvelles technologies développées répondent à ces besoins [149]. En même temps, l'industrie automobile a besoin de satisfaire sa clientèle en gardant toujours haut ces standards de performances. Pour cela, les constructeurs reviennent à la charge pour trouver un moyen d'y remédier, en ayant l'espoir que la voiture électrique puisse conquérir dans un avenir proche le cœur des consommateurs [149].

La conception préliminaire des véhicules électriques peut être réalisée en utilisant des outils de modélisation multi-domaines tels que VHDL-AMS [150], Matlab / Simulink [7, 40], Modelica [39], etc.

Plusieurs études ont été effectuées pour optimiser les caractéristiques de la source de tension [151] et des moteurs électriques [152, 153] afin d'améliorer les capacités de la voiture électrique et d'obtenir un véhicule plus fiable.

Dans ce chapitre, nous proposons un nouveau modèle pour la conception préliminaire d'un véhicule électrique. Le modèle proposé est basé sur la combinaison de Modelica avec ModelCenter. Modelica [39] a été utilisé pour modéliser et vérifier le modèle proposé. ModelCenter [154] a été utilisé pour optimiser les variables de conception des différents composants du modèle afin de respecter les exigences de performance demandées.

Le modèle développé est utilisé par la suite comme cas d'application afin de démontrer la validité et l'efficacité de l'approche multi-agents proposée dans le chapitre précédent pour faciliter la conception collaborative des systèmes distribués.

Ce chapitre est organisé comme suit : après l'introduction, la section 2 présente la formulation mathématique du véhicule électrique, la section 3 décrit la modélisation du véhicule électrique avec Modelica, la section 4 donne les résultats de simulation afin de vérifier le modèle proposé, la section 5 traite l'estimation de la quantité d'énergie de la batterie nécessaire pour parcourir une certaine distance, la section 6 fournit une étude de cas de l'approche multi-agents pour faciliter la conception collaborative distribuée du véhicule électrique et la section 7 donne les remarques de conclusion.

4.2 Modélisation Mathématique

4.2.1 Architecture du véhicule électrique

Afin d'être en mesure de modéliser un véhicule électrique, il est très important de disposer d'une architecture appropriée. Différentes architectures du véhicule électrique existent [151, 155–158] selon le type de la machine électrique (machine à courant continu ou alternatif), la nature de l'onduleur (monophasé ou triphasé) et les caractéristiques de la batterie (haute ou basse tension).

Dans ce chapitre, l'architecture de la *Figure 4.1* est choisie [151]. Cette architecture comporte : une source de tension continue (batterie), un onduleur de tension triphasée, une machine électrique et un système de transmission.

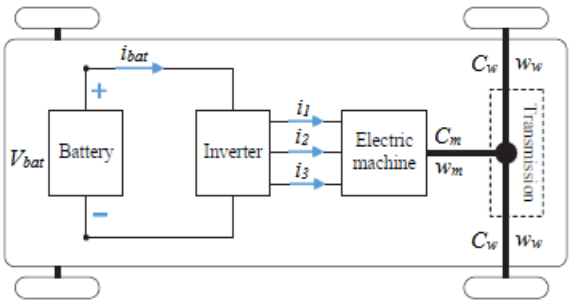


FIGURE 4.1 – Architecture du véhicule électrique.

4.2.2 Développement mathématique du véhicule électrique

4.2.2.1 Modèle des forces

La première étape de la modélisation du véhicule consiste à décrire un modèle des forces. Les efforts résistants qui vont s'opposer au couple fourni par la machine électrique sont les forces dues à la gravité, la résistance au roulement, l'aérodynamique et l'effet d'inertie. Ces forces sont présentées dans la *Figure 4.2*.

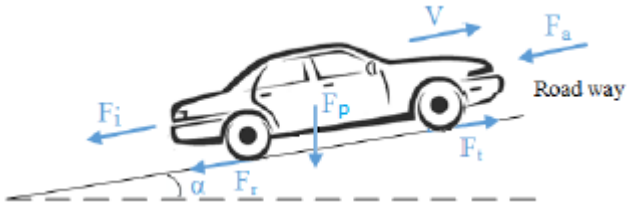


FIGURE 4.2 – Forces appliquées sur le véhicule.

L'effort total de résistance à l'avancement que doit vaincre le système de motorisation, afin d'accélérer le véhicule est donné par [151] :

$$F_t = F_a + F_r + F_p + F_i \tag{4.1}$$

Avec

$$\begin{cases} F_a = \frac{1}{2}.C_d.\rho.A_{front}.V^2 \\ F_r = C_{ro}.M.g.cos(\alpha) \\ F_p = M.g.sin(\alpha) \\ F_i = M.\frac{dV}{dt} \end{cases} \tag{4.2}$$

Où

- F_t : la force de traction,
- F_a : la force aérodynamique,
- F_r : la force de résistance au roulement,
- F_p : la force pour vaincre une certaine pente,
- F_i : la force liée à l'accélération,
- C_d : le coefficient d'entraînement aérodynamique,
- ρ : la masse volumique de l'air,
- A_{front} : la surface frontale du véhicule,
- V : la vitesse du véhicule,
- C_{ro} : le coefficient de résistance au roulement,
- M : la masse du véhicule,
- g : l'accélération de la pesanteur,

- α : l'angle d'inclinaison de la pente.

Remarques :

- Une pente de 100 % correspond à un angle d'inclinaison de 90° (pente infinie).
- Prise en compte de la vitesse du vent V_{vent} [159] : pour prendre en compte l'influence de la vitesse du vent sur le déplacement du véhicule, nous remplaçons la vitesse V, qui est en fait une vitesse relative déduite de la vitesse de rotation de la machine au rapport de réduction près, par la vitesse réelle V_{reel}, avec :

$$V_{reel} = V - V_{vent} \tag{4.3}$$

Convention :

- $V_{vent} > 0$: si le vent s'oppose au déplacement du véhicule.
- $V_{vent} < 0$: si le vent et le véhicule se déplacent dans le même sens.

4.2.2.2 Système de transmission

Le couple, la vitesse angulaire et la puissance du système de transmission sont donnés par les équations suivantes [151] :

$$\begin{cases} C_t = F_t.r_w \\ C_w = \frac{C_t}{2} \\ w_w = \frac{V}{r_w} \\ P_t = F_t.V \end{cases} \tag{4.4}$$

Où

- C_t : le couple de traction,
- C_w : le couple de chaque roue motrice,
- r_w : le rayon de la roue,
- w_w : la vitesse angulaire des roues,
- P_t : la puissance de traction.

La chaîne de traction étudiée est munie d'un réducteur mécanique reliant l'actionneur à l'arbre de transmission des roues. Ce réducteur est caractérisé par un rapport de réduction R_g. Dans ces conditions, la vitesse de rotation du moteur d'entraînement et le couple qu'il doit fournir sont liés aux grandeurs côté transmission par les relations [157] :

$$\begin{cases} w_m = w_w.R_g \\ C_m = \frac{C_t}{R_g} \\ P_m = C_M.w_m \end{cases} \tag{4.5}$$

Où

- C_m : le couple mécanique de la machine électrique,
- w_m : la vitesse angulaire de l'arbre de la machine électrique,
- R_g : le rapport de réduction,
- P_m : la puissance mécanique de la machine électrique.

4.2.2.3 Machine électrique

Le moteur synchrone à aimants permanents (MSAP) semble la solution la plus adaptée pour une traction automobile grâce à ses performances techniques et en particulier, sa compacité et son rendement [160]. Il a été retenu par Toyota dans son modèle Prius pour les raisons suivantes [161] :

- bon rendement,
- bonnes performances dynamiques grâce à la faiblesse des inductances statoriques due à la largeur importante de l'entrefer apparent,
- champ magnétique important dans l'entrefer,
- pas de source de tension continue pour l'excitation.

La modélisation de la machine électrique est divisée en deux parties ; une partie électrique et une partie mécanique. La partie électrique de la MSAP est donnée par [157] :

$$\begin{cases} U_1 = R.i_1 + L_1.\frac{di_1}{dt} + M_{12}.\frac{di_2}{dt} + M_{13}.\frac{di_3}{dt} + e_1 \\ U_2 = R.i_2 + L_2.\frac{di_2}{dt} + M_{21}.\frac{di_1}{dt} + M_{31}.\frac{di_3}{dt} + e_2 \\ U_3 = R.i_3 + L_3.\frac{di_3}{dt} + M_{31}.\frac{di_1}{dt} + M_{32}.\frac{di_2}{dt} + e_3 \end{cases} \tag{4.6}$$

Où

- U_i : la tension aux bornes de la phase i,
- i_i : le courant dans la phase i,
- L_i : l'inductance propre de la phase i,
- M_{ij} : l'inductance mutuelle entre la phase i et la phase j,
- e_i : la f.e.m. induite par le champs rotorique dans la phase i.

Les MSAP présentent lors de leur rotation des f.e.m sinusoïdales, dont les expressions sont de la forme [157] :

$$\begin{cases} e_1 = K.\frac{w_e}{p}.sin(we.t) \\ e_2 = K.\frac{w_e}{p}.sin(we.t - \frac{2.\Pi}{3}) \\ e_3 = K.\frac{w_e}{p}.sin(we.t - \frac{4.\Pi}{3}) \end{cases} \tag{4.7}$$

Où

- k : le coefficient de f.e.m incluant notamment le nombre de spires par phase ainsi que l'amplitude du champ rotorique,

- w_e : la vitesse angulaire électrique,
- p : le nombre de paire de pôles.

La partie mécanique de la machine synchrone à aimant permanent MSAP peut être modélisée de la manière suivante [157] :

$$J.\frac{w_m}{dt} = C_m - B.w_m \tag{4.8}$$

Où

- J : le moment d'inertie de l'arbre moteur,
- B : le coefficient de frottement visqueux.

Le couplage entre la partie électrique et la partie mécanique est donné par [157] :

$$w_e = p.w_m \tag{4.9}$$

4.2.2.4 Onduleur

Un onduleur triphasé est constitué de trois cellules de commutation dont les commandes décalées entre elles d'1/3 de période permettent de reconstituer un système triphasé de tensions et de courants.

Dans notre cas, nous avons choisi un onduleur de type DC/AC pour transformer la tension continue à la sortie de la batterie V_{bat} en une tension triphasée V_1, V_2 et V_3 comme entrée du moteur électrique en changeant l'état des six interrupteurs. Le schéma de circuit électrique de l'onduleur peut être vu dans la *Figure 4.3*.

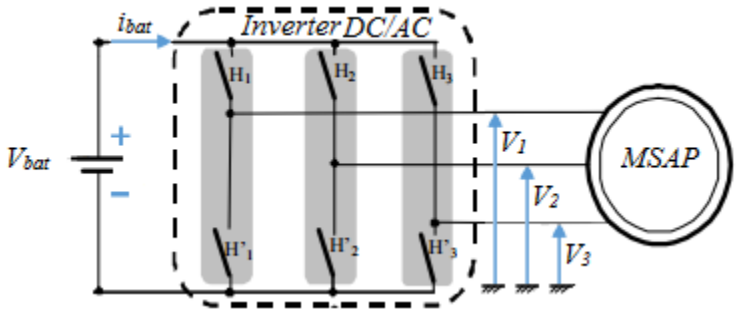


FIGURE 4.3 – Schéma de circuit électrique de l'onduleur.

Les tensions moyennes phase-neutre à la sortie de l'onduleur sont des tensions sinusoïdales alternatives, de même fréquence, de même valeur efficace et régulièrement déphasées

de $\frac{2.\Pi}{3}$. Ces tensions sont données par les expressions suivantes [157] :

$$\begin{cases} V_1 = V.sin(w_e.t) \\ V_2 = V.sin(w_e.t - \frac{2.\Pi}{3}) \\ V_3 = V.sin(w_e.t - \frac{4.\Pi}{3}) \end{cases} \tag{4.10}$$

Avec

$$V = m_a.\frac{V_{bat}}{2} \tag{4.11}$$

Où

- V_i : la tension moyenne entre la phase i et le neutre,
- V : la valeur efficace des tensions moyennes phase-neutre,
- V_{bat} : la tension de la batterie,
- m_a : le facteur de modulation.

La valeur de m_a est comprise entre 0 et 1. Par exemple, selon la technique de modélisation m_a peut prendre deux valeurs maximales [162] :

- m_{amax}=0.907 : pour la technique de modulation de largeur d'impulsion vectorielle SVPWM (Space-Vector Pulse Width Modulation).
- m_{amax}=0.7854 : pour la technique de modulation de largeur d'impulsion sinus-triangle STPWM (Sine-Triangle Pulse Width Modulation).

L'efficacité de l'onduleur est donnée par [159] :

$$\eta = \frac{P_{ond}}{P_{bat}} \tag{4.12}$$

Avec

$$\begin{cases} P_{ond} = V_1.i_1 + V_2.i_2 + V_3.i_3 \\ P_{bat} = V_{bat}.i_{bat} \end{cases} \tag{4.13}$$

Où

- P_{ond} : la puissance à la sortie de l'onduleur,
- P_{bat} : la puissance de la batterie,
- V_{bat} : la tension de la batterie,
- i_{bat} : le courant de la batterie.

4.2.2.5 Batterie

La batterie est la source d'énergie du véhicule. Elle est constituée de l'association en série et/ou parallèle de cellules élémentaires. Le schéma équivalent de la batterie est présenté à la

Figure 4.4.

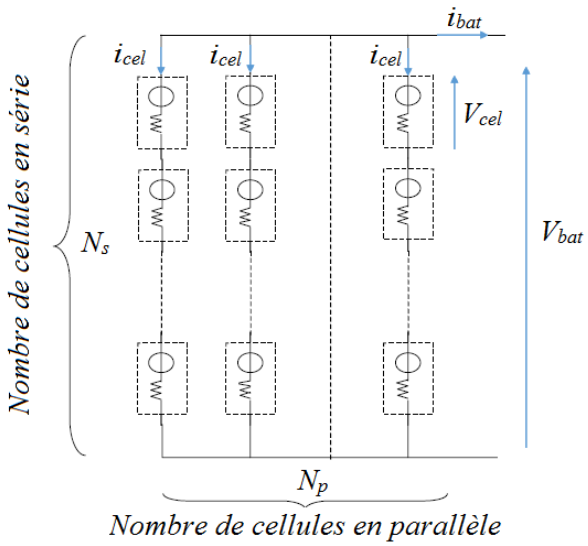


FIGURE 4.4 – Schéma équivalent de la batterie.

La tension totale, le courant et la masse de la batterie peuvent être calculées par les expressions suivantes [157] :

$$\begin{cases} V_{bat} = N_s.v_{cel} \\ i_{bat} = N_p.i_{cel} \\ M_{bat} = N_s.N_p.M_{cel} \end{cases} \tag{4.14}$$

Où

- N_s : le nombre de cellules en série,
- N_p : le nombre de cellules en parallèle,
- V_{cel} : la tension d'une cellule,
- i_{cel} : le courant dans une cellule,
- M_{cel} : la masse d'une cellule,
- M_{bat} : la masse de la batterie.

Afin d'évaluer le comportement de la batterie, il est nécessaire de déterminer la variation de son état de charge SoC (State-of-charge en anglais). Le SoC correspond à la quantité de charge, notée C, pouvant être restituée par la batterie par rapport à sa capacité nominale, notée C_0 et correspondant à $SoC = 100\%$ (batterie pleinement chargée) [159] :

$$\begin{cases} SoC = SoC_{init} - 100.\frac{C}{C_0} \\ C_N = \frac{1}{3600}.\int i_{bat}.dt \end{cases} \tag{4.15}$$

Les capacités C et C_0 sont exprimées en A.h.

4.3 Modélisation du véhicule électrique avec Modelica

Après le développement mathématique du véhicule électrique, le système a été modélisé en utilisant le langage orienté objet Modelica. Comme il est indiqué dans la *Figure 4.5*, ce système est composé par les éléments suivants :

- une batterie (source de tension continue),
- un onduleur (convertisseur DC-AC),
- une machine électrique (moteur électrique synchrone),
- un modèle de transmission,
- une force de résistance,
- un système de contrôle.

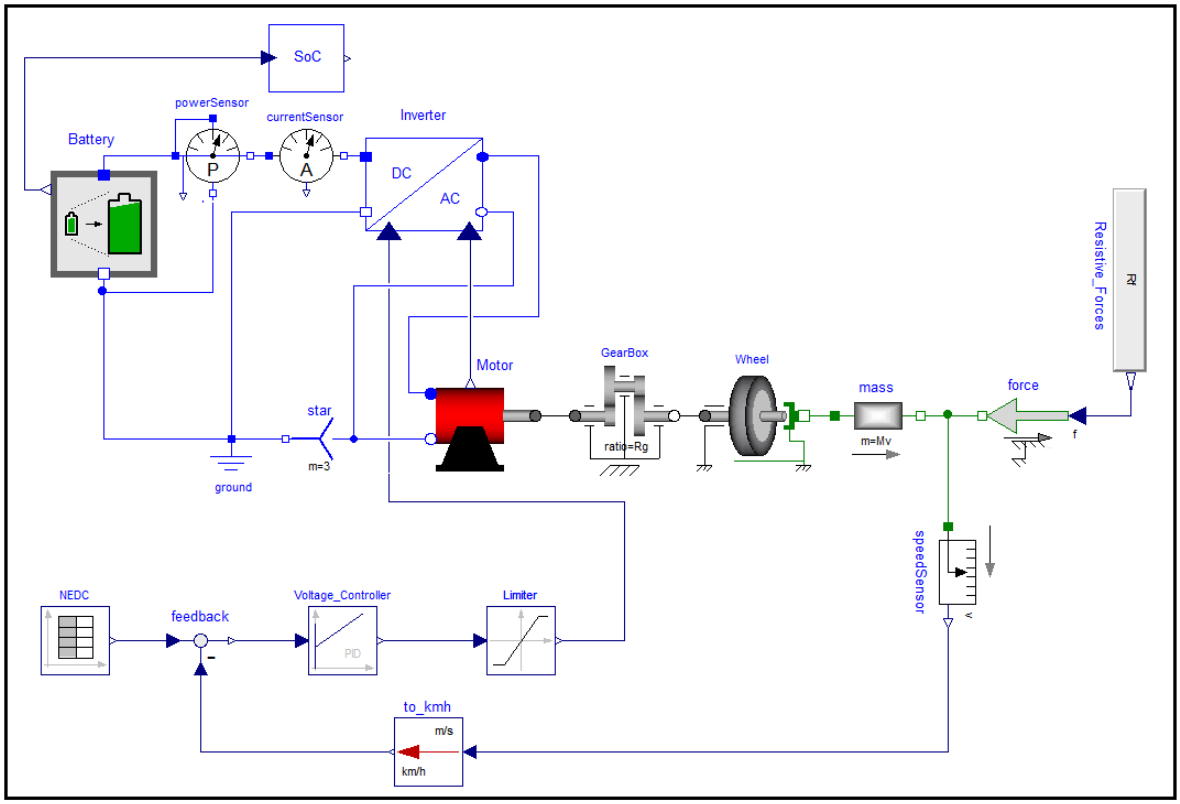


FIGURE 4.5 – Modèle du véhicule électrique développé avec Modelica.

Un capteur de courant *currentSensor*, un capteur de puissance *powerSensor*, un capteur de vitesse *speedSensor* et un composant *SoC* sont utilisés pour mesurer respectivement le courant électrique généré par la batterie, la puissance électrique requise par le moteur électrique, la vitesse du véhicule et l'état de charge de la batterie.

4.3.1 Batterie

4.3.1.1 Principaux types de batteries

Il existe différents types de batteries. Si leur fonctionnement est similaire, les matériaux utilisés réagissent différemment et ont donc des performances différentes. Les types de batteries les plus utilisées par l'industrie automobile sont au nombre de trois : batteries au plomb, batteries au nickel et batterie au lithium [163, 164].

Batteries au plomb

Une batterie dite au plomb (ou plomb/acide) [164] est une batterie qui utilise une anode et une cathode en plomb. Elle présente l'avantage de délivrer un courant de forte puissance permettant d'alimenter des moteurs eux aussi puissants. En revanche, elle a une densité énergétique faible et offre donc une faible autonomie à un véhicule. De plus, la durée de vie est plutôt limitée pour ce type de batterie (autour de 600 cycles).

Le principal avantage, qu'elle est simple à produire et le plomb est un métal commun : le coût est donc faible. Cependant, le plomb est toxique et polluant même s'il se recycle plutôt facilement.

Batteries au nickel

Les batteries Nickel-cadmium (Ni-Cd) [164] sont très courantes dans l'industrie en général et ont équipé un certain nombre d'anciens projets de véhicules électriques. Elles sont légèrement plus performantes que les batteries au plomb (jusqu'à 80 Wh/kg) et relativement bon marché. Cependant, elles possèdent un effet mémoire important ce qui rend leur utilisation contraignante. De plus, le cadmium est très polluant et difficile à recycler.

Les batteries Nickel métal-hydrure (Ni-Mh) stockent plus d'énergie (jusqu'à 110 Wh/kg) et ne contiennent pas d'élément très polluant. En revanche, leur durée de vie est limitée et elles se déchargent rapidement même quand elles ne sont pas utilisées (auto-décharge).

Enfin, les batteries Nickel-Zinc (Nickel Metal-Zinc) ont des performances comparables à la technologie Nickel-Cadmium. En revanche, elles sont beaucoup moins polluantes, mais plus chères.

Batteries au lithium

L'appellation batterie au lithium [164] regroupe de nombreuses technologies qui utilisent du lithium sous différentes formes. Ces technologies sont aujourd'hui les plus performantes, mais restent aussi très chères. Elles semblent toutefois s'imposer dans l'industrie automobile.

Les batteries lithium-ion, tout d'abord, utilisent le lithium sous forme d'ions insérés dans l'électrolyte. Mais là encore il existe différents types, basés sur des matériaux différents :

- les batteries lithium-ion classiques : elles sont très performantes (150 à 200 Wh/kg), mais chères. En effet, ces accumulateurs utilisent du cobalt ou du manganèse qui sont rares et peuvent provoquer des réactions instables. Ils requièrent alors des systèmes de contrôle coûteux. Elles ont une durée de vie plutôt élevée (autour de 1000 cycles) et surtout aucun effet mémoire.

- les batteries lithium-ion fer phosphate sont moins performantes (100 Wh/kg) mais beaucoup plus sûres et moins coûteuses. Leur durée de vie peut atteindre les 2000 cycles.

Les batteries lithium polymère sont très proches des batteries lithium-ion, sauf qu'elles utilisent un électrolyte solide (gélifié). Cela permet de produire des accumulateurs avec des formes très diverses et qui sont plus sûres que les batteries lithium-ion classiques. Ils sont, en revanche, moins performants et plus chers à produire.

4.3.1.2 Modélisation de la batterie avec Modelica

L'objectif de cette partie est de créer un modèle capable de simuler le comportement réel d'une batterie, compte tenu de sa décharge et aussi de la diminution conséquente de l'état de charge. Dans notre cas, le choix du modèle de la batterie est basé sur une cellule de type AMP 20 Lithium Ion [165] en raison des avantages qu'elle confère par rapport aux autres technologies :

- densité énergétique plus importante que les autres,
- un temps de recharge moins important,
- un faible risque de pollution.

Un simple modèle de la batterie est indiqué dans la *Figure 4.6*.

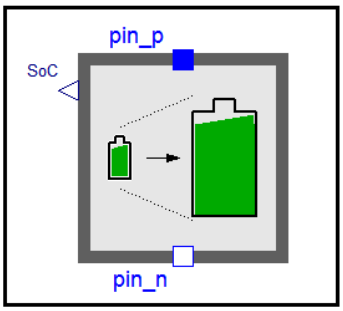


FIGURE 4.6 – Vue externe de la batterie.

Allant profondément dans l'architecture interne du modèle comme il est monté dans la *Figure 4.7*, le comportement électrique de la batterie est dicté seulement par des blocs qui se trouvent dans la bibliothèque Modelica. Avant de décrire le rôle de chaque bloc, il est important de spécifier que ce modèle fonctionne en cycle et que son rôle principal est de fournir

une tension entre les deux ports de la batterie *pin_p* et *pin_n*.

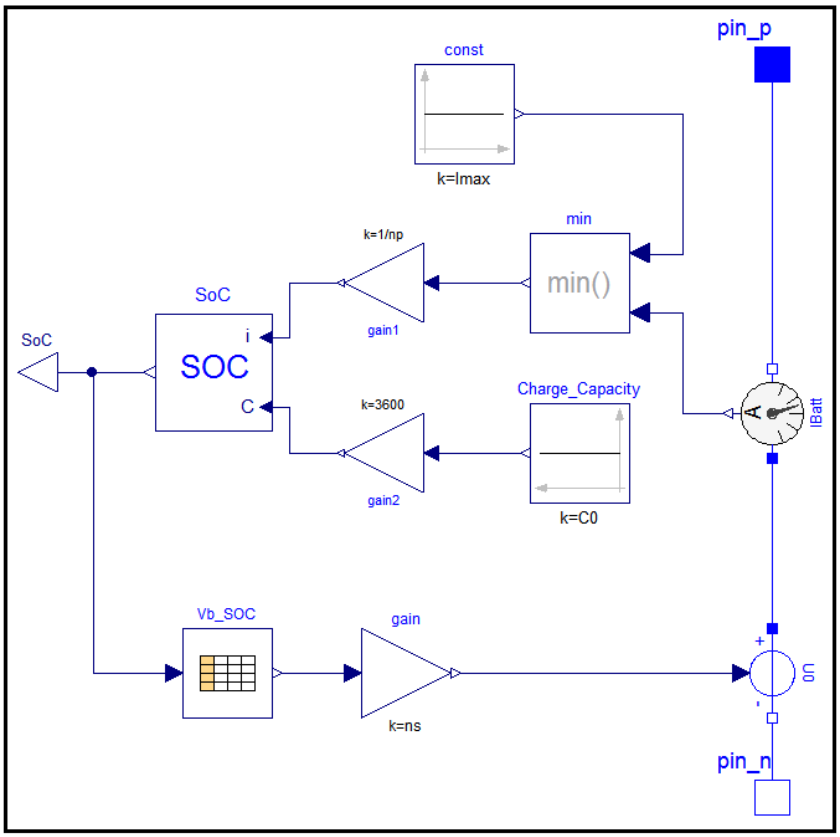


FIGURE 4.7 – Vue interne de la batterie.

La tension U_0 génère un courant mesuré par l'ampèremètre *currentSensor*. Ce courant est comparé avec le courant admissible I_{max}. Le minimum de ces deux courants est divisé par le nombre de cellules en parallèle N_p pour obtenir le courant dans chaque cellule (représenté par le *gain1*). Le courant d'une cellule i circule par la suite dans le bloc *SoC*. L'autre entrée de ce bloc est la capacité de la batterie C_0 exprimée en $A.h$ (le *gain2* est nécessaire pour effectuer une conversation d'unité de $A.s$ en $A.h$).
Le bloc *SoC* calcule l'état de charge de la batterie par l'équation :

$$SoC = SoC_{int} - \int \frac{i(t)}{C_0} dt \tag{4.16}$$

où SoC_{int} est l'état de charge initial de la batterie.

La valeur instantanée de l'état de charge est une entrée d'une table V_b qui fournit la tension d'une cellule de la batterie en fonction de l'état de charge. Cette valeur, multipliée par le nombre de cellules en série N_s (représenté par le *gain3*), est utilisée comme référence pour générer la tension de la batterie. Pour évaluer le comportement des différentes technologies

des batteries, le concepteur peut modifier la table V_b à la sortie du block *SoC*.

Un niveau de modélisation plus fin de ce modèle pourrait être l'insertion d'un tableau pour modéliser la variation de la résistance interne en fonction de l'état de charge et de la température de la batterie.

4.3.2 Onduleur

Le modèle de l'onduleur a été développé en utilisant un code de calcul pour effectuer une conversion de courant continu en courant alternatif sur la base de l'équilibre entre la puissance d'entrée, qui provient de la batterie et la puissance de sortie, qui sera remise à un moteur à courant alternatif. Comme on peut le voir sur la *Figure 4.8*, le modèle est composé de deux ports d'entrée (p,n), deux ports de sortie (P,N) et de deux connecteurs (a,we).

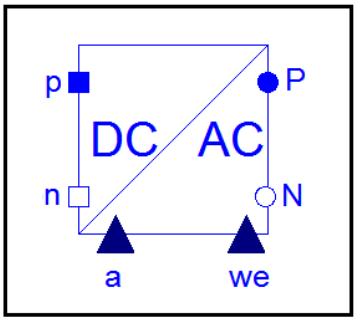


FIGURE 4.8 – Modèle externe de l'onduleur.

Les ports négatifs n et N sont reliés à la masse, les ports positifs p et P assurent la liaison de l'onduleur avec la batterie et le moteur électrique, respectivement. Le premier connecteur a représente le signal d'entrée du contrôle utilisé pour régler l'amplitude de la tension triphasée. Le deuxième connecteur *we* représente la pulsation électrique requise par le moteur afin de fonctionner correctement.

Passant maintenant à l'intérieur du modèle, comme il est indiqué dans la *Figure 4.9*, on peut voir qu'il y a un seul paramètre modifiable, qui est le rendement de l'onduleur *nu*. Ainsi, le concepteur peut modifier sa valeur selon les spécifications du fabricant de l'appareil choisi.

L'aspect le plus important qui a été pris en compte lors du processus de modélisation de l'onduleur est d'assurer l'équilibre entre les puissances d'entrée et de sortie. Cette relation est donnée par l'équation (4.2) :

$$Pi = -nu * Pb \tag{4.17}$$

Les deux puissances électriques *Pb* et *Pi* sont calculées par le produit entre la tension et le courant à l'entrée et à la sortie de l'onduleur, respectivement.

```
model Inverter

  //Inverter Efficiency
  parameter Real nu=0.98;

  // Battery Voltage
  Modelica.SIunits.Voltage Vb;

  // Effective Voltage
  Modelica.SIunits.Voltage Veff;

  //Battery Power
  Modelica.SIunits.Power Pb;

  //Output Power from Inverter
  Modelica.SIunits.Power Pi;
equation
  // Effective Voltage Calculation
  Vb = p.v - n.v;
  Veff = a*Vb/2;
  //Balance of Current
  p.i + n.i = 0;
  P.pin.i + N.pin.i = fill(0,3);
  //Balance of Power
  Pi = -nu*Pb;
  Pb = (p.v-n.v) * p.i;
  Pi = (P.pin.v-N.pin.v) * P.pin.i;
  //Output Voltage from Inverter
  P.pin[1].v - N.pin[1].v = a*Vb/2*sin(we*time);
  P.pin[2].v - N.pin[2].v = a*Vb/2*sin(we*time-2
                        *Modelica.Constants.pi/3);
  P.pin[3].v - N.pin[3].v = a*Vb/2*sin(we*time+2
                        *Modelica.Constants.pi/3);
end Inverter;
```

FIGURE 4.9 – Code Modelica de l'onduleur.

$$Pb = (p.v - n.v) * p.i \tag{4.18}$$

$$Pi = (P.pin.v - N.pin.v) * P.pin.i \tag{4.19}$$

Les tensions à la sortie de l'onduleur sont des tensions sinusoïdales déphasées de $2\pi/3$ l'une par rapport à l'autre. À titre d'exemple, l'expression de la tension de la première phase est donnée par l'équation suivante :

$$P.pin[1].v - N.pin[1].v = Veff * sin(we * t) \tag{4.20}$$

La tension efficace $Veff$ est calculée par l'amplification de la tension de la batterie comme suit :

$$Veff = a * \frac{Vb}{2} \tag{4.21}$$

Néanmoins, la tension efficace n'est pas la seule variable importante dans l'équation (1). Il y a aussi les variables a et we. La variable a est sans dimension qui découle du système de contrôle de l'onduleur afin d'amplifier ou réduire l'amplitude de la tension triphasée. La fréquence angulaire we dérive de la vitesse angulaire mécanique de l'arbre moteur et permet de changer la forme des ondes de signaux de tension en fonction de la demande de couple moteur.

4.3.3 Moteur électrique synchrone

L'idée de cette partie est de modéliser le moteur sous une forme mathématique en écrivant les équations fondamentales qui génèrent ses comportements mécaniques et électriques.

Le modèle du moteur représente la transition entre les niveaux électriques et mécaniques. Comme on le voit sur la *Figure 4.10*, le modèle se compose de deux ports d'entrée (P,N) et un connecteur de sortie $flange\_a$. Les ports P et N sont liés à l'onduleur et la masse, respectivement. Le connecteur mécanique $flange\_a$ est responsable de la liaison avec le réducteur mécanique, qui délivre deux sorties : l'angle ($flange\_a.phi$) et le couple ($flange\_a.tau$) de l'arbre moteur. La variable we (la troisième sortie du modèle) représente la fréquence angulaire électrique demandée par le moteur pour atteindre les exigences du cycle de conduite. Il est important de souligner que ce connecteur de sortie est lié avec le paramètre we du modèle de l'onduleur, décrit et expliqué précédemment.

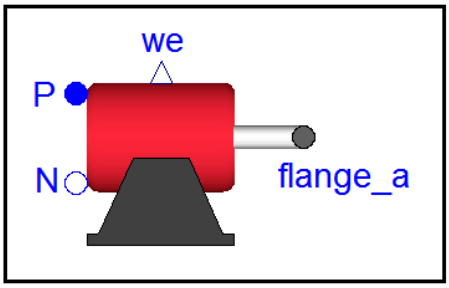


FIGURE 4.10 – Vue externe du modèle de la machine électrique.

Par conséquent, la modélisation mathématique du moteur a été développée en fonction des paramètres mentionnés ci-dessus, des équations du circuit électrique du moteur et du connecteur mécanique à la sortie du moteur. La logique de la modélisation et la formulation des équations sont entièrement expliquées et détaillées dans les paragraphes suivants.

Tout d'abord, afin de donner au concepteur la possibilité de modifier les spécifications du moteur qu'il veut insérer sur son banc d'essai virtuel, trois paramètres de performance

modifiables ont été créés :

- La puissance maximale demandée par le moteur *Pmax*.
- La vitesse angulaire maximale du moteur *wmax*.
- Le couple maximal du moteur *Cmax*.

La *Figure 4.11* représente les premières lignes du code de calcul qui régit le comportement du moteur électrique. En plus de ces variables, on a créé aussi six autres variables nécessaires pour simuler les caractéristiques des composants internes du moteur :

- La résistance interne du moteur *R*.
- L'inductance interne du moteur *L*.
- L'inductance mutuelle *M*.
- L'inertie de l'arbre moteur *J*.
- La constante de la force électromotrice *Emf*.
- Le nombre de paires de pôles *p*.

```
parameter Modelica.SIunits.Power Pmax;            "Maximum power"
parameter Modelica.SIunits.AngularVelocity wmax; "Maximum angular velocity"
parameter Modelica.SIunits.Torque Cmax;           "Maximum torque"

//Internal characteristics of the motor

parameter Modelica.SIunits.Resistance R;          "Internal resistance"
parameter Modelica.SIunits.Inductance L;          "Internal inductance"
parameter Modelica.SIunits.Inductance M;          "Mutual inductance"
parameter Real p;                                 "Number of pairs of poles"
parameter Modelica.SIunits.MagneticFlux Emf;      "Constant of the electromotive force"
parameter Modelica.SIunits.Inertia J;             "Shaft inertia";
```

FIGURE 4.11 – Déclaration des variables de performance des composants internes du moteur.

L'étape suivante a été réalisée dans le but de faciliter la compréhension de ce code et de réduire la taille des principales équations. La *Figure 4.12* montre la liste des paramètres de calcul utilisés. Comme il est indiqué, trois vecteurs ont été créés pour modéliser les valeurs instantanées du courant triphasé *i*, la tension en circuit ouvert *V* et la tension électromécanique *E*, puis, quatre variables ont été utilisées pour accomplir la transition entre les niveaux électrique et mécanique : le couple moteur *C*, la vitesse de rotation mécanique *wm*, la vitesse angulaire électrique *we* et la puissance électrique *Pe*.

Après la déclaration de tous ces paramètres et ces variables, l'étape suivante consiste à écrire le système d'équations (mécanique, électrique et électromécanique) caractéristique du moteur électrique. Ainsi, afin de modéliser le comportement réel du modèle, les équations suivantes sont prises en considération :

```
//Electric Circuit

Modelica.SIunits.Current i[3];       "Three-phase current "
Modelica.SIunits.Voltage v[3];       "Open circuit voltage"
Modelica.SIunits.Voltage E[3];       "Electromechanical voltage"

//transition between the electrical and mechanical levels

Modelica.SIunits.Torque C;             "Mechanical Torque"
Modelica.SIunits.AngularVelocity we;   "Electric angular velocity"
Modelica.SIunits.AngularVelocity wm;   "Mechanical rotation speed"
Modelica.SIunits.Power Pe;             "Electric Power"
```

FIGURE 4.12 – Paramètres utiles pour le code de calcul du moteur.

$$v = P.pin.v - N.pin.v \tag{4.22}$$

$$i = P.pin.i \tag{4.23}$$

$$P.pin.i + N.pin.i = fill(0,3) = [0\ \ 0\ \ 0] \tag{4.24}$$

Les équations (4.22) et (4.23) sont utilisées pour calculer la tension et le courant, respectivement (en fonction des ports d'entrée P et N du moteur), l'équation (4.24) est nécessaire pour assurer la cohérence entre le nombre d'inconnues et le nombre d'équations du modèle (le nombre d'inconnues doit être égal au nombre d'équations). Ces équations sont modélisées par les premières lignes du code dans la *Figure 4.13*. Ensuite, les équations de la force élec-

```
//Electric Equations
 v = P.pin.v - N.pin.v;
 P.pin.i + N.pin.i = fill(0,3);
 i = P.pin.i;

//Synchrounous Machine Equations

 E[1]=Emf/p*we*sin(we*time);
 E[2]=Emf/p*we*sin(we*time - 2*pi/3);
 E[3]=Emf/p*we*sin(we*time + 2*pi/3);

 v[1] = R*i[1] + L * der(i[1]) + M * der(i[2])
              + M * der(i[3]) + E[1];
 v[2] = R*i[2] + L * der(i[2]) + M * der(i[3])
              + M * der(i[1]) + E[2];
 v[3] = R*i[3] + L * der(i[3]) + M * der(i[1])
              + M * der(i[2]) + E[3];

//Electric Power
 Pe = v*i;
```

FIGURE 4.13 – Les équations caractéristiques du moteur électrique.

tromagnétique E sont considérées. De ce fait, afin de tenir compte des trois phases, chaque

valeur de E est liée à la valeur de son courant respectif. Cette corrélation peut être représentée par l'équation (4.10), dans laquelle l'indice j appartient à $\{0, 1, 2\}$.

$$E[j+1] = Emf/p * we * sin(we * time + (2 * \pi * j)/3) \tag{4.25}$$

Les équations du circuit électrique ouvert qui gouvernent le comportement du moteur sont rédigées selon le schéma représenté sur la *Figure 4.14*. Comme on peut le voir, le circuit est composé d'une résistance R, qui représente les pertes internes de la machine, une inductance L et une inductance mutuelle M. Une fois le calcul de E est terminé, les tensions aux bornes des différentes phases peuvent être calculées. Par exemple, l'expression de la tension de la première phase est donnée par l'équation suivante :

$$v[1] = R * i[1] + L * \frac{di[1]}{dt} + M * \frac{di[2]}{dt} + M * \frac{di[3]}{dt} + E[1] \tag{4.26}$$

La puissance électrique totale *Pe* est donnée par l'équation suivante :

$$Pe = v * i \tag{4.27}$$

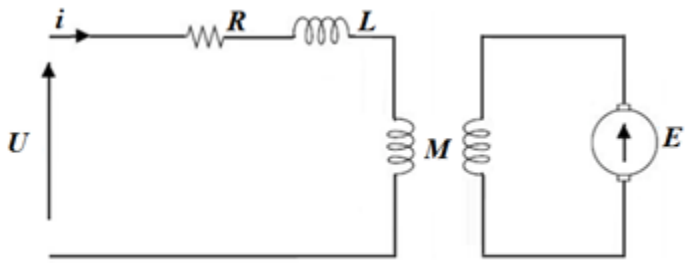


FIGURE 4.14 – Schéma du circuit électrique d'un moteur.

Enfin, une fois toutes les relations mathématiques responsables à la modélisation du comportement interne du moteur sont complètement décrites, l'accent est maintenant dirigé vers la spécification des valeurs des paramètres de sortie du modèle.

Dans ce but, d'une part le couple moteur est calculé en divisant la puissance électrique *Pe* (qui est égale à la puissance mécanique) par la vitesse mécanique de l'arbre moteur *wm*. Afin de respecter les contraintes de la machine, le résultat est comparé au couple maximum *Cmax* admis par le moteur et la valeur minimale entre les deux est considérée comme le couple mécanique nécessaire *C*. D'autre part la vitesse mécanique de l'arbre moteur est calculée de la même manière, mais sa valeur est obtenue par la comparaison entre la rotation maximale du moteur *wmax* et la dérivée de l'angle de l'arbre moteur (la dérivée de $flange\_a.phi$).

Par la suite, la pulsation électrique de l'onduleur *we* est calculée en divisant la vitesse mécanique de l'arbre par le nombre de paires de pôles p. Toutes les dernières relations mentionnées ci-dessus sont présentées sur la *Figure 4.15*.

```
//Mechanical Torque
 C = if abs(wm)>0 then min(min(Pmax,Pe)/wm,Cmax) else Cmax;

//Mechanical Angular Velocity
 wm = min( der(flange_a.phi),  wmax);

//Mechanical Equation
 J*der(wm) = C - B*wm;

//Conversion from electric angle to mechanical angle
 we = p*wm;
```

FIGURE 4.15 – Spécifications des sorties du moteur.

4.3.4 Système de Transmission

Afin de simuler le comportement réel d'un véhicule, trois composants mécaniques de la bibliothèque Modelica sont insérés dans le modèle global et connectés à la sortie du moteur. Comme il est indiqué sur la *Figure 4.16*, ces composants sont :

- Gear Box,
- Wheel,
- Mass.

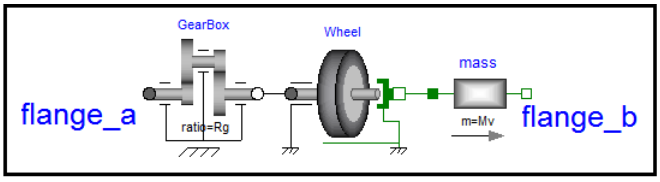


FIGURE 4.16 – Modèle de transmission du véhicule électrique.

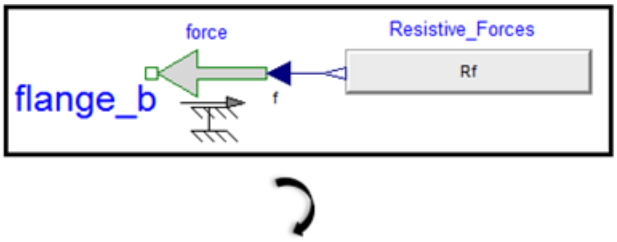
Pour chaque modèle du véhicule, il existe des valeurs distinctes des paramètres liés à ces éléments. Ainsi, le concepteur doit spécifier toutes les caractéristiques du système de transmission avant de lancer la simulation.

Concernant le Gear Box, il y a un seul paramètre qui doit être déterminé, qui est le rapport de la transmission Rg. Les deux autres composants (Wheel et Mass) sont nécessaires pour transformer le mouvement de rotation de l'arbre moteur en mouvement de translation du véhicule. Pour ce faire, le concepteur doit définir ici deux paramètres :

- le rayon des roues rw,
- la masse du véhicule Mv.

4.3.5 Effort résistant

Cette partie est constituée de deux blocs, comme la montre la *Figure 4.17*. Le premier bloc *Resistive_Forces* est constitué d'une expression mathématique qui calcule l'effort résistant en fonction de la vitesse instantanée du véhicule et le deuxième bloc *force* reçoit le résultat généré pour le transformer en une force, qui doit être appliquée sur la dernière partie du système de transmission.




```
model Resistive_Forces

 //Resistive forces
 Modelica.SIunits.Force Rf;
 //Parameters of the vehicle:
 parameter Modelica.SIunits.Mass Mv;          "Vehicle Mass"
 parameter Modelica.SIunits.Radius rw;        "Wheel radius"
 parameter Modelica.SIunits.Area Afront;      "Frontal Area"
 parameter Real Cd;                           "Aerodynamic drag coefficient"
 parameter Modelica.SIunits.Density Rho;      "Air density"
 parameter Modelica.SIunits.Acceleration g;  "Gravitational acceleration constant"
 //Parameters of the road:
 parameter Real Cro;                          "Rolling coefficient"
 parameter Modelica.SIunits.Angle alpha;      "Grade angle"
equation
  Rf=-(1/2*Cd*Rho*A*V.v^2 + Cro*M*g*cos(alpha)+M*g*sin(alpha))
end Resistive_Forces;
```

FIGURE 4.17 – Effort résistant appliqué sur le véhicule électrique.

4.3.6 Système de Contrôle

Le système de contrôle est composé de deux entrées (la vitesse de référence et la vitesse calculée) et un paramètre de contrôle (coefficient de contrôle de la tension de l'onduleur), comme la montre la *Figure 4.18*.

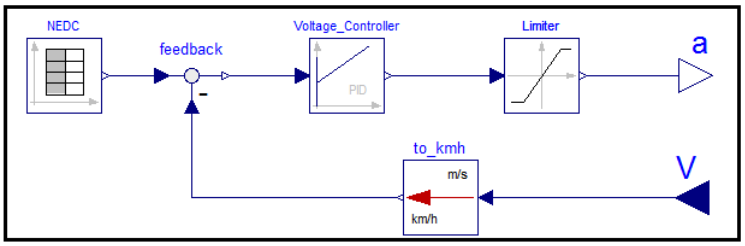


FIGURE 4.18 – Système de contrôle du véhicule électrique.

Le système fonctionne comme suit : d'abord, le bloc feedback calcule la différence entre la vitesse de référence et la vitesse calculée à la sortie du block to_km/h de façon à transmettre le résultat au $VoltageController$. Le $VoltageController$ est responsable à la lecture et le calcul de la réponse de $feedback$ afin de donner la valeur la plus appropriée pour le paramètre de sortie a. Néanmoins, si la sortie a n'est pas limitée, dans ce cas le système de contrôle est risque d'être instable. Par conséquent, il est nécessaire d'ajouter un autre bloc pour limiter la valeur de a (dans notre cas c'est le bloc $Limiter$).

Les caractéristiques internes de système de contrôle sont indiquées dans le *Tableau 4.1*.

Paramètres	*Valeur*
Ti	0.5
Td	0.5

Tableau 4.1 – Paramètres du système de contrôle

Pour vérifier le modèle d'analyse qui sera utilisé plus tard dans le processus d'optimisation, l'objectif de la partie suivante, il est nécessaire de comparer la vitesse à la sortie du véhicule électrique avec un cycle de conduite standard fourni en entrée.

4.4 Les résultats de simulation

Dans le cadre de cette étude, un cycle de conduite qui exprime l'évolution de la vitesse du véhicule en fonction du temps est utilisé comme référence pour comparer le modèle développé dans la partie précédente. Il permet d'évaluer aussi la variation des différents paramètres du véhicule tels que le courant de la batterie, la puissance de la batterie, l'effort de traction, le couple moteur et la vitesse de rotation de l'arbre moteur, etc.

Pour les besoins de la simulation et pour reproduire un trajet routier avec différentes conditions de conduites, on a recours au cycle de vitesse européen normalisé NEDC (New European Driving Cycle) [166].

Dans cette étude, les paramètres utilisés dans le modèle de véhicule électrique sont donnés dans le *Tableau 4.2*. L'objectif est de simuler le modèle proposé. Les valeurs par défaut du nombre de cellules en série N_s, le nombre de cellules en parallèles N_p, la constante de la force électromotrice Emf et le rapport de transmission R_g sont respectivement 50, 5, 1 et 6.

Paramètres	*Description*	*Valeur*	*Unité*
M	Masse du véhicule	1540	Kg
ρ	Masse volumique de l'air	1.2	$kg.m^{-3}$
A_{front}	Surface frontale du véhicule	1.8	m^2
C_{ro}	Coefficient de résistance au roulement	0.2	–
C_d	Coefficient d'entraînement aérodynamique	0.013	–
α	Angle d'inclinaison de la pente	0	rad
g	Accélération de pesanteur	9.81	$m.s^{-2}$
r_w	Rayon de la roue	0.28	m
R_g	Rapport de réduction	$1 \leq R_g \leq 12$	–
Emf	Constante de la f.e.m	$0.1 \leq Emf \leq 2$	$N.m.A^{-1}$
η	Efficacité de l'onduleur	0.98	–
N_s	Nombre de cellules en série	$1 \leq N_s \leq 100$	–
N_p	Nombre de cellules en parallèle	$1 \leq N_p \leq 10$	–
SoC_{init}	Etat de charge initial	0.8	–
V	Vitesse de véhicule	Variable	km/h
SoC	Etat de charge	Variable	–
P	Puissance électrique	Variable	kW

Tableau 4.2 – Paramètres du véhicule électrique

Les résultats de simulation du véhicule selon le cycle NEDC sont donnés par les *Figures 4.19*, *4.20* et *4.21*.

La *Figure 4.19* représente la variation de la vitesse durant le cycle NEDC et la vitesse du véhicule mesurée à la sortie du composant to_km/h. Cette figure confirme que la vitesse mesurée (courbe rouge) suit le profil de la route imposée (courbe bleue) avec une erreur < 2%.

La connaissance de l'état de charge *SOC* de la batterie est un élément déterminant par rapport au comportement du système complet. La *Figure 4.20* montre la variation de la *SOC* pendant le cycle NEDC. Le *SoC* initial est égal à 80 % de la charge totale de la batterie. La plage de fonctionnement de *SOC* est comprise entre 67 % et 80 %, cela signifie que la batterie a perdu 16.25 % de sa capacité pendant le cycle NEDC. Avec un état de charge minimal SOC_{min} = 20 % et les mêmes valeurs de N_s, N_p, Emf et R_g le véhicule électrique peut répéter 4.6 fois le cycle NEDC.

La *Figure 4.21* montre la variation de la puissance électrique pendant le cycle de conduite

NEDC.

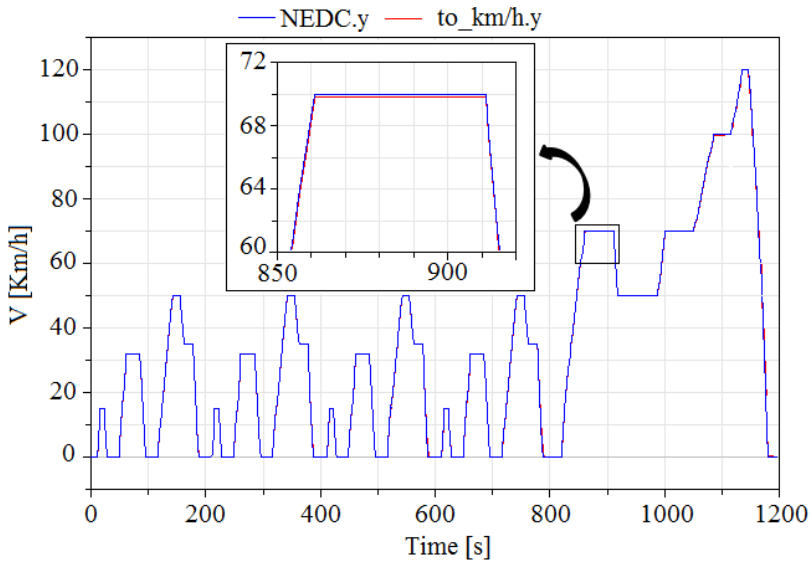


FIGURE 4.19 – Comparaison entre la vitesse imposée par le cycle NEDC et la vitesse mesurée à la sortie du modèle.

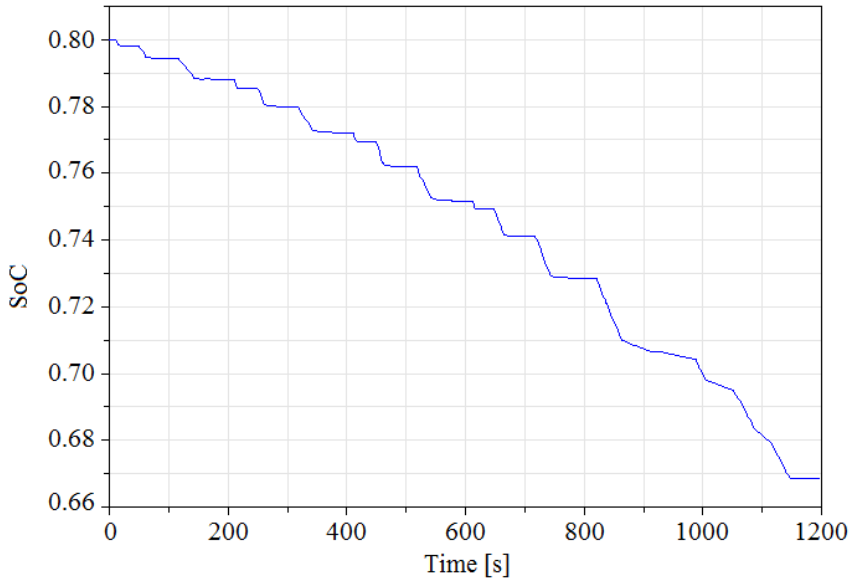


FIGURE 4.20 – Variation de l'état de charge de la batterie durant le cycle NEDC.

En modifiant l'entrée par une vitesse constante, on peut simuler différents autres cas de test de performance, tels que la vitesse maximale du véhicule, la vitesse d'accélération à t=10 secondes et la vitesse du véhicule en pente.

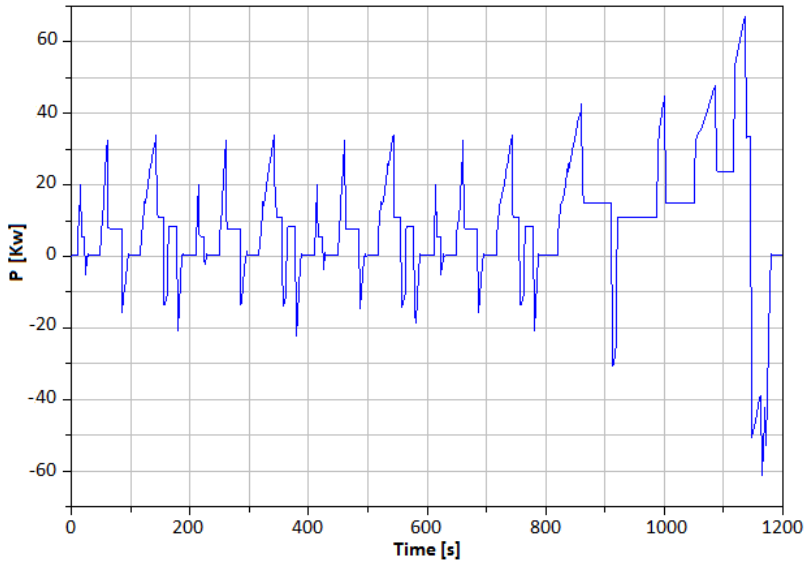


FIGURE 4.21 – Variation de la puissance électrique durant le cycle NEDC.

Par exemple, la puissance maximale donnée par la simulation d'un test de vitesse maximale égal à 120 km/h pendant 250 secondes est de l'ordre de 64 kW, l'état de charge de la batterie égale à 73 % et la masse de la batterie est égale à 198.4 kg . Ces résultats doivent être optimisés pour réduire la puissance consommée par le moteur électrique et la masse de la batterie et de maximiser l'état de charge de la batterie à la fin du cycle de conduite.

4.5 Estimation de l'énergie requise par la batterie

L'objectif de cette partie est d'estimer la quantité d'énergie de la batterie nécessaire pour parcourir une certaine distance.
Les exigences suivantes sont prises en compte :

- La vitesse imposée à l'entrée est une constante.
- La route est horizontale $\Longrightarrow \alpha$=0.
- La vitesse de vent est nulle.
- La batterie totalement chargée (SoC_{init}=80%).

Le modèle du véhicule électrique développé dans la section précédente a été utilisé pour calculer la valeur de l'énergie requise par la batterie à chaque kilomètre parcouru par le véhicule. La seule différence est que le cycle européen normalisé NEDC est remplacé par une vitesse constante, c'est la vitesse de référence que le véhicule doit atteindre. De plus, d'autres paramètres ont été également ajoutés à l'intérieur du modèle, comme il est indiqué dans la *Figure 4.22*. Ces paramètres sont définis par les équations suivantes :

- *Energy*, c'est l'énergie totale consommée par le véhicule lors du cycle imposé ;

$$Energy = \frac{1}{3600}\int (powerSensor.Power)dt \tag{4.28}$$

- $Range_{Km}$, c'est la distance parcourue par le véhicule en kilomètre ;

$$Range_{Km} = \frac{1}{1000}\int (speedSensor.speed)dt \tag{4.29}$$

- $Energy_{Km}$, c'est l'énergie requise par le véhicule pour parcourir un kilomètre à une vitesse constante ;

$$Energy_{Km} = \frac{Energy}{Range_{Km}} \tag{4.30}$$

```
// Total Energy
Real Energy;
// Energy to travel one kilometer at a constant speed
Real Energy_Km "Wh/km";
// Range
Real Range_km(start=0) "in km";
equation
  der(Energy) = powerSensor.power/3600;
  der( Range_km*1000) = speedSensor.v;
  Energy_Km = if time > 1 then  Energy/Range_km else 0;
```

FIGURE 4.22 – Code Modelica pour l'estimation de l'énergie requise par la batterie.

La *Figure 4.23* représente une étude paramétrique *Parametricstudy* effectuée avec ModelCenter pour déterminer la fonction $Energy_{Km}$. Dans cette étude, les variables d'entrée sont le nombre total de cellules et la vitesse de référence à atteindre et le paramètre de sortie (réponse) est la fonction $Energy_{Km}$.

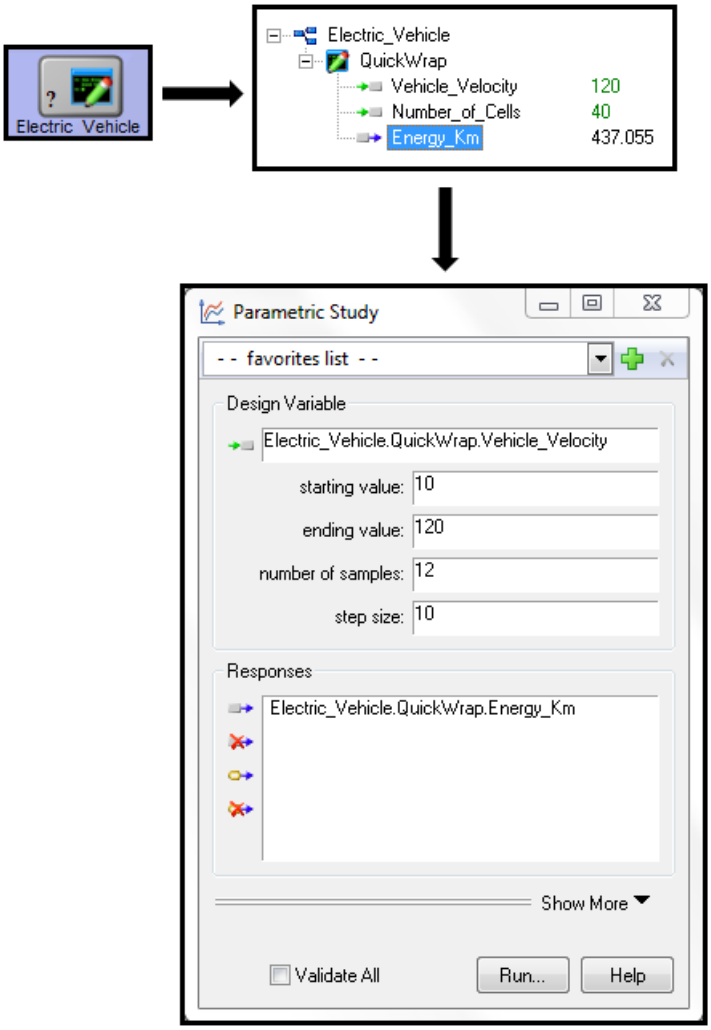


FIGURE 4.23 – Interface graphique de l'étude paramétrique avec du ModelCenter.

L'étude paramétrique fonctionne en boucle itérative. En fixant à chaque fois le nombre de cellules, $Energy_{Km}$ est calculée en changeant la vitesse de référence entre une valeur minimale *startingvalue* et une valeur maximale *endingvalue*, comme il est indiqué dans la *Figure 4.24*.

À partir des résultats indiqués dans la *Figure 4.24*, il est possible de tracer les courbes de la *Figure 4.25*, qui montrent la variation de $Energy_{Km}$ en fonction de la vitesse à atteindre pour un certain nombre cellules.

Ces graphiques peuvent être utilisés comme référence par le concepteur pour calculer l'énergie totale requise par un véhicule électrique en fixant certaines exigences telles que :

- la distance à parcourir,
- la vitesse du véhicule,

Vehicle Velocity	Number of Cells						
	200	250	300	350	400	450	500
10	204.1528	247.0976	290.1852	333.4155	376.7885	420.304	463.963
20	228.289	273.469	318.896	364.571	410.493	456.661	503.077
30	243.546	289.745	336.238	383.025	430.105	477.48	525.148
40	256.811	303.631	350.776	398.244	446.037	494.154	542.595
50	270.615	317.998	365.736	413.828	462.275	511.076	560.232
60	287.893	336.055	384.613	433.567	482.918	532.666	582.81
70	307.261	356.201	405.581	455.401	505.662	556.363	607.505
80	328.379	378.068	428.241	478.899	530.042	581.671	633.784
90	351.243	401.647	452.58	504.042	556.034	608.556	661.609
100	376.01	427.108	478.78	531.027	583.849	637.247	691.222
110	402.923	454.686	507.106	560.148	613.814	668.105	723.02
120	433.201	484.722	537.819	591.716	646.29	701.542	757.472

FIGURE 4.24 – Étude paramétrique effectuée à l'intérieur du ModelCenter pour estimer l'énergie.

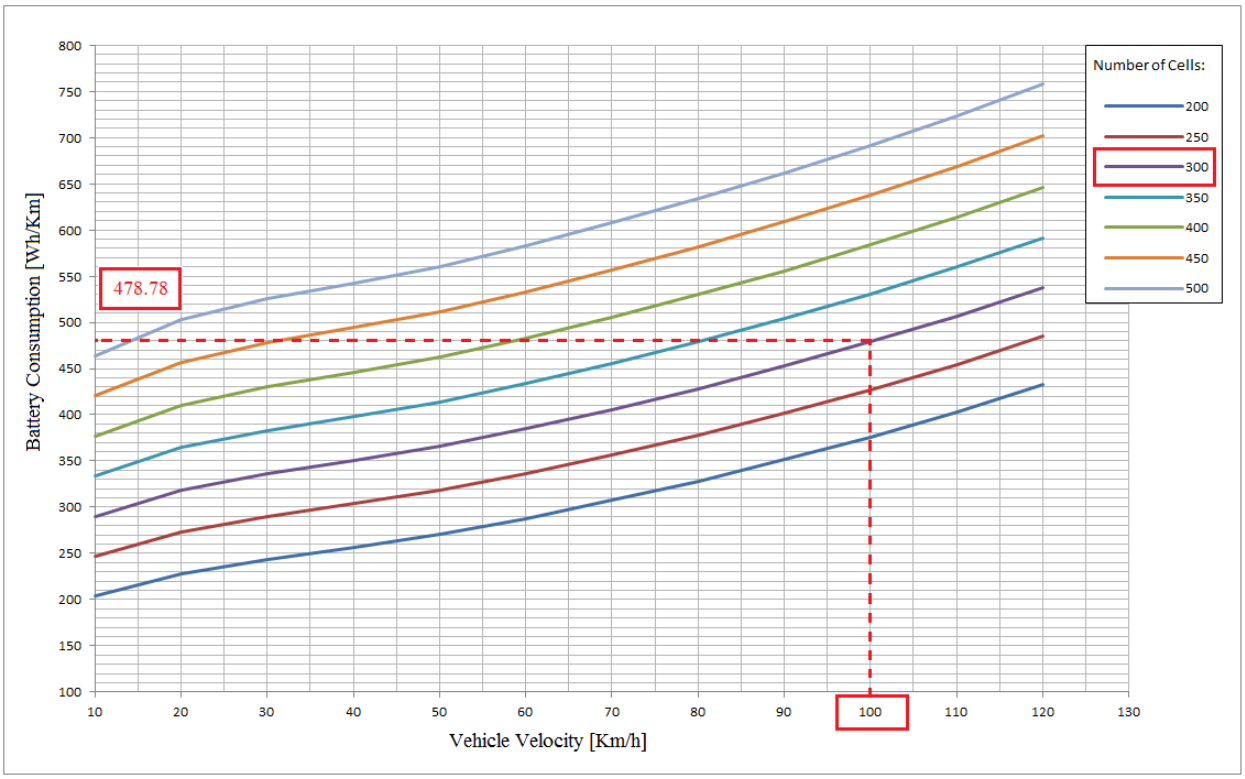


FIGURE 4.25 – Variation de $Energy_{Km}$ en fonction de la vitesse à atteindre pour un certain nombre cellules.

- le nombre de cellules.

EXEMPLE :

Exigences :

- la distance à parcourir = 100 km,
- la vitesse = 100 km/h,
- le nombre de cellules = 300.

Le concepteur doit sélectionner sur l'axe x la vitesse requise qui est égale à 100 km/h, après cela, en utilisant la courbe de 300 cellules, il doit lire sur l'axe y la valeur de $Energy_{Km}$ nécessaire. Si la distance à parcourir est égale à 100 km, compte tenu d'un cycle de décharge de la batterie, alors l'énergie totale $TotalEnergy$ est donnée par la relation suivante :

$$TotalEnergy = 100.k.Energy_{Km} \tag{4.31}$$

Où k est un facteur qui prend en considération que la batterie doit être rechargée lorsque son SoC est inférieur à une valeur minimale (généralement SoC_{min}= 20%).

$$k = \frac{1}{SoC_{int} - SoC_{min}} \tag{4.32}$$

4.6 Optimisation multi-objectif du véhicule électrique

4.6.1 Définition des critères d'optimisation

Les métriques essentielles pour mesurer les performances d'un véhicule sont liées à la vitesse maximale, l'accélération et le test de franchissement de rampe (en pente) ou "gradeability" :

- La vitesse maximale du véhicule est définie comme étant la vitesse maximale que le véhicule peut atteindre quand le moteur est en plein régime.
- La performance d'accélération est donnée par le temps d'accélération et la distance couverte, quand le véhicule part de la vitesse zéro jusqu'à une vitesse de référence donnée (par exemple de 0 à 100 km/h sur une route horizontale).
- La "gradeability" est définie par l'angle de la pente de route que le véhicule peut surmonter à une vitesse de référence constante. Elle peut être exprimée en degré ou en pourcentage.

Suite à l'analyse architecturale du véhicule électrique, le concepteur calcule les caractéristiques de la batterie, du moteur électrique et de la partie transmission mécanique tout en vérifiant les tests de performance de vitesse maximale, d'accélération et de gradeability. Pour cela, certains paramètres du véhicule doivent être déterminés au préalable ou donnés par le cahier des charges. Ces paramètres concernent la masse du véhicule, le coefficient de résistance au roulement, le coefficient de résistance aérodynamique, la surface frontale

du véhicule, le rendement de la transmission mécanique. Des spécifications de performance doivent être indiquées par le cahier des charges concernant la vitesse maximale, le temps d'accélération de 0 jusqu'à une certaine vitesse de véhicule, la gradeability demandée, etc.

4.6.2 Formulation d'un problème d'optimisation multi-objectif

Nous considérons l'optimisation d'un véhicule électrique avec quatre fonctions objectifs et trois contraintes de conception :

- Objectif f_1 : minimiser la masse du système de stockage d'énergie afin de choisir la batterie optimale.
- Objectif f_2 : maximiser l'état de charge de la batterie à la fin du cycle de conduite choisi.
- Objectif f_3 : minimiser la puissance électrique demandée par les systèmes de propulsion afin de choisir le moteur électrique optimal.
- Objectif f_4 : minimiser le rapport de transmission afin de réduire le volume occupé par la boîte de vitesses.
- Contrainte c_1 : Test d'accélération : la vitesse du véhicule après 10 secondes de démarrage (V_{10}) est égale à $60 \pm 5\ km/h$.
- Contrainte c_2 : Vitesse maximale : la vitesse maximale du véhicule (V_{max}) est égale à $120 \pm 5\ km/h$.
- Contrainte $c3$: Masse de la batterie : la masse de la batterie doit être inférieure à $200\ kg$.

Pour les variables de conception, nous limitons notre étude au nombre de cellules en série N_s, le nombre de cellules en parallèles N_p, la constante de la force électromotrice Emf du moteur électrique et le rapport de transmission R_g.

En utilisant une approche d'optimisation classique, en définissant un modèle d'analyse (modèle Modelica développé dans la partie précédente) pour les différents cas de test (test d'accélération, vitesse maximale, etc.) est une tâche complexe et coûteuse en temps de calcul. D'où l'idée d'utiliser l'approche multi-agents pour résoudre ces cas de problème.

4.6.3 Utilisation de l'approche multi-agent

La première étape du processus de conception consiste à trouver les configurations possibles pour le partitionnement du problème d'optimisation. Une métrique importante qui est utilisée, dans ce cas, est le degré de couplage entre les fonctions objectifs, les contraintes et les variables de conception.

D'après la formulation du problème d'optimisation, les deux contraintes V_{10} et V_{max} sont liées aux performances globales du système. Il est difficile d'affecter ces deux contraintes à des partitions spécifiques. De plus, la décomposition suivant les variables de conception est très efficace dans le cas où tout le problème est modélisé sous une forme mathématique. Mais dans notre cas, le problème du véhicule électrique est modélisé à la fois par des composants et par des modèles mathématiques. Et par conséquent l'idéal est d'étudier le degré de couplage

entre les fonctions objectifs qui sont associées à des composants spécifiques (batterie, moteur et réducteur) pour décomposer le problème en plusieurs partitions et d'affecter à chaque partition ses contraintes et ses variables de conceptions internes.

Plusieurs combinaisons peuvent être étudiées pour partitionner le problème d'optimisation du véhicule électrique ; certains d'entre elles sont réalisables. Les trois configurations possibles sont : une partition, deux partitions et trois partitions (*Figure 4.26*).

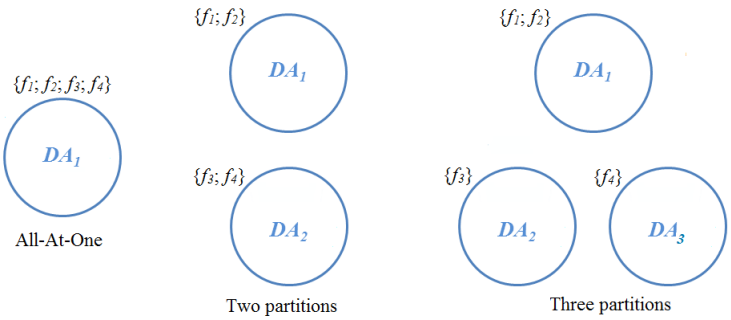


FIGURE 4.26 – Les configurations possibles de partitionnement pour le cas d'optimisation du véhicule électrique.

Une partition, est un problème d'optimisation multi-disciplinaire où toutes les fonctions objectifs sont associées à une seule partition qui correspond à un problème tout-en-un (AAO). Dans ce cas, un seul modèle d'analyse est nécessaire pour résoudre le problème global. Les méthodes AAO actuellement utilisées pour l'optimisation des systèmes multi-disciplinaires sont coûteuses en temps de calcul, difficiles à mettre en œuvre, et inflexibles avec la phase de conception préliminaire, dans laquelle les objectifs et les contraintes de conception changent fréquemment.

Dans la configuration avec deux partitions, les deux fonctions objectifs f_1 et f_2 sont fortement couplées, de sorte qu'elles doivent être assignées à la même partition. Par contre, le degré de couplage entre les deux autres fonctions objectifs f_3 et f_4 est faible. Par conséquent, la solution est de considérer deux modèles d'analyse. Le premier modèle, c'est le modèle de la source d'énergie électrique pour analyser f_1 et f_2 et le deuxième modèle, c'est le système de propulsion pour traiter f_3 et f_4. Ces deux modèles sont associés à deux partitions, qui peuvent être exécutées en parallèle sur le même ordinateur ou sur deux ordinateurs différents, ce qui réduit le temps de calcul.

Dans la configuration avec trois partitions, le degré de couplage entre f_3 et f_4 est faible de sorte que chaque fonction peut être associée à un modèle d'analyse. Dans ce cas, trois modèles d'analyse sont nécessaires pour traiter ce problème : modèle de la batterie pour traiter f_1 et f_2, modèle de la machine électrique pour calculer f_3 et le dernier modèle, c'est le modèle de la chaîne de transmission pour déterminer f_4.

La deuxième étape consiste à trouver la meilleure configuration à utiliser pour résoudre

ce problème. Les deux critères utilisés pour le choix de la meilleure configuration sont : le temps de calcul de la solution globale pour chaque configuration C_1 et nombre de partitions de la configuration à choisir C_2. Ces deux critères doivent être minimisés. D'après le *Tableau 4.3*, le choix est trivial puisque les deux critères de sélection sont en faveur pour la configuration avec deux partitions.

Critère	Configuration1	Configuration2	Configuration3
C_1 (pour une évaluation)	5.17 secondes	2.6 secondes	2.1 secondes
C_2	1	2	3

Tableau 4.3 – Choix de la meilleure configuration

Cette configuration est caractérisée par les variables de couplages suivantes : la tension U et le courant I. L'espace de recherche de ses variables est inconnu et par conséquent, il est nécessaire de passer par les étapes de 4 à 6 durant le processus de conception afin de déterminer le domaine de validité de U et I.

La détermination des domaines de validité des variables de couplages nécessite d'abord d'effectuer une étude de sensibilité des variables de conception. Les quatre variables de conceptions N_s, N_p, Emf et R_g ont une grande influence sur les performances demandées du système. En effet, la fonction objectif f_1 est exprimé en fonction de N_s et N_p ($M_{bat} = N_s * N_p * M_{cel}$), la fonction f_3 dépend des caractéristiques internes du moteur en particulier de la constante de la force électromotrice Emf et la fonction f_4 représente le rapport de réduction R_g.

L'étape suivante consiste à construire un méta-modèle du problème global. Plusieurs techniques ont été utilisées pour construire un méta-modèle. La méthode de surface de réponse polynomiale (RSM) est généralement considérée comme la première technique de modélisation de substitution. Elle est basée sur une formulation polynomiale pour approximer une fonction exacte. D'autres techniques tels que le krigeage « Kriging » et les réseaux de neurones « Radial Basis Function Neural Networks » RBFNNs sont également utilisés pour modéliser les relations complexes entre les entrées et les sorties du problème. Dans cet exemple, les techniques RSM et krigeage disponibles dans la bibliothèque de ModelCenter [154] sont utilisées pour construire le méta-modèle.

Le vecteur d'entrée du méta-modèle est donné par :

$$X = \begin{bmatrix} N_s \\ N_p \\ Emf \\ R_g \end{bmatrix}$$

Le vecteur de sortie (appelée aussi vecteur de réponse) est défini par :

$$Y = \begin{bmatrix} M_{bat} \\ SoC \\ P \\ Rg \\ V_{10} \\ V_{max} \\ U \\ I \end{bmatrix}$$

Le domaine de la conception est défini par :

$$\begin{cases} 1 \leq N_s \leq 100 \\ 1 \leq N_p \leq 10 \\ 0.1 \leq Emf \leq 2 \\ 1 \leq R_g \leq 12 \end{cases}$$

La *Figure 4.27* représente le choix de la technique de substitution pour chaque composant du vecteur de réponse Y. M_{bat}, SoC, P, I et U sont approximées par la méthode RSM alors que R_g, V_{10} et V_{max} sont approximées par un modèle de krigeage.

Response Surfaces

Name	Type	Status	R^2 Adj.
Model.QuickWrap.mbat	Polynomial	Done	99.7819%
Design Explorer Kriging	Design Explorer Kriging	Done	98.04%
Polynomial	Polynomial	Done	99.7819%
Model.QuickWrap.soc	Polynomial	Done	99.6999%
Design Explorer Kriging	Design Explorer Kriging	Done	99.09%
Polynomial	Polynomial	Done	99.6999%
Model.QuickWrap.pmax	Polynomial	Done	97.8797%
Design Explorer Kriging	Design Explorer Kriging	Done	97.79%
Polynomial	Polynomial	Done	97.8797%
Model.QuickWrap.rg	Design Explorer Kriging	Done	98.9829%
Design Explorer Kriging	Design Explorer Kriging	Done	98.9829%
Polynomial	Polynomial	Done	98.93%
Model.QuickWrap.v10	Design Explorer Kriging	Done	98.9997%
Design Explorer Kriging	Design Explorer Kriging	Done	98.9997%
Polynomial	Polynomial	Done	98.27%
Model.QuickWrap.vmax	Design Explorer Kriging	Done	99.3729%
Design Explorer Kriging	Design Explorer Kriging	Done	99.3729%
Polynomial	Polynomial	Done	96.08%
Model.QuickWrap.imax	Design Explorer Kriging	Done	99.7819%
Design Explorer Kriging	Design Explorer Kriging	Done	98.04%
Polynomial	Polynomial	Done	99.7819%
Model.QuickWrap.umax	Polynomial	Done	98.9606%
Design Explorer Kriging	Design Explorer Kriging	Done	98.9606%
Polynomial	Polynomial	Done	98.9%

FIGURE 4.27 – Construction du méta-modèle avec ModelCenter.

Une fois le méta-modèle est construit, le concepteur définit un scénario d'optimisation pour déterminer les domaines de validité des variables de couplages. Dans notre cas, l'optimisation de méta-modèle est réalisée en utilisant des algorithmes disponibles dans la bibliothèque de ModelCenter, en particulier l'algorithme Non-dominated Sorting Genetic Algorithm II (NSGA II). NSGA II [167] est une technique d'optimisation multi-objectifs qui utilise un algorithme génétique de tri non-dominé. Une conception est dite dominée s'il y a une autre conception qui lui est supérieure dans tous les objectifs.

Comme il est indiqué dans la *Figure 4.28*, trois composants qui sont insérés dans l'interface graphique du ModelCenter. Le premier composant *ElectricVehicle* est le modèle Modelica du véhicule électrique à partir duquel est construit le deuxième composant *SurrogateModel* (un méta-modèle du modèle d'analyse Modelica). Le dernier composant *OptimizationTool* représente l'optimisation basée sur le méta-modèle, en utilisant l'algorithme NSGAII de ModelCenter.

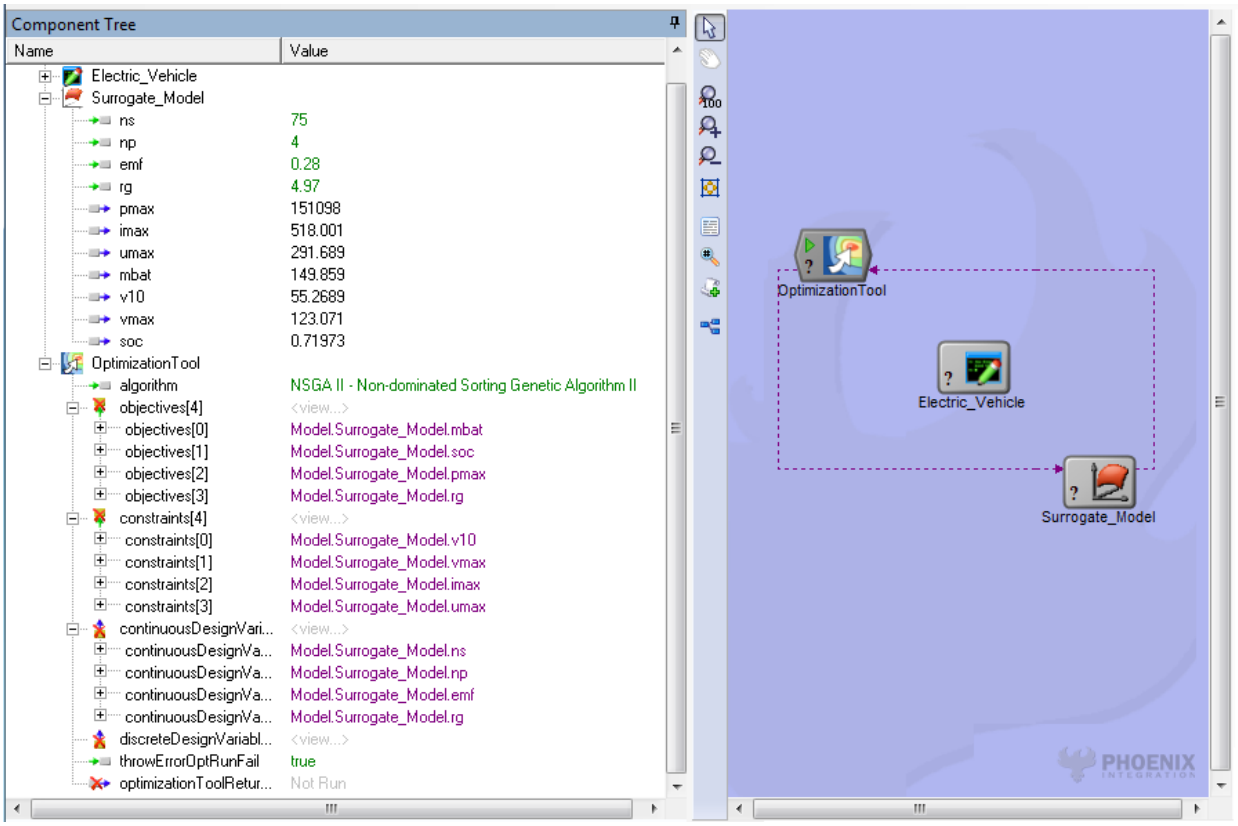


FIGURE 4.28 – Optimisation basée sur le méta-modèle avec ModelCenter.

Le problème d'optimisation est multi-objectif, le méta-modèle n'a pas une solution unique, mais un ensemble de solutions dont les limites min et max des vecteurs X et Y sont données dans le *Tableau 4.4*.

	Méta-modèle
N_s	74 - 83
N_p	4 - 5
$Emf(Nm/A)$	0.203 - 0.381
R_g	4.73- 7.91
$M_{bat}(Kg)$	149.7 - 198.2
SoC	0.722 - 0.754
$P(kW)$	54.6 – 65.7
$V_{10}(km/h)$	55.1– 64.4
$V_{max}(km/h)$	115.4 – 123.9
$I(A)$	141.6 - 155.3
$U(V)$	385.8- 422.7

Tableau 4.4 – Résultats de l'optimisation du modèle de substitution (méta-modèle)

L'étape suivante consiste à formuler les sous problèmes d'optimisation des deux partitions de la configuration choisie à partir des étapes précédentes. Chaque partition a un modèle d'analyse (modèle Modelica) pour évaluer ses contraintes et ses objectifs locaux. Pour chaque partition, nous allouons le même algorithme d'optimisation NSGA II avec ModelCenter.

Le premier modèle d'analyse est constitué par la batterie en série avec une résistance équivalente qui représente le reste du modèle du véhicule électrique. La valeur de cette résistance est déterminée à partir des résultats d'optimisation du méta-modèle.

$$R_{eq} = \frac{U_{max}}{I_{max}} = \frac{U_{min}}{I_{min}} = 2.72\Omega \tag{4.33}$$

Le deuxième modèle d'analyse, c'est le modèle du système de propulsion. Pour avoir une simulation exacte de ce système on doit lui connecter une source de tension constante U pour remplacer le modèle de la batterie.

La formulation de chaque sous-problème d'optimisation est donnée par :

$$Partition1 : \begin{cases} min(M_{bat}) \\ max(SoC) \\ Mbat \leq 200 \\ 141.6 \leq I \leq 155.3 \\ 385.8 \leq U \leq 422.7 \\ 1 \leq N_s \leq 100 \\ 1 \leq N_p \leq 10 \end{cases} \tag{4.34}$$

$$Partition2 : \begin{cases} min(P) \\ min(R_g) \\ 55 \leq V_{10} \leq 65 \\ 115 \leq V_{max} \leq 125 \\ 141.6 \leq I \leq 155.3 \\ 385.8 \leq U \leq 422.7 \\ 1 \leq R_g \leq 12 \\ 0.1 \leq Emf \leq 2 \end{cases} \tag{4.35}$$

Pour la configuration choisie avec deux partitions, nous allons considérer deux agents de conception DA_1 et DA_2. Chaque agent est affecté à une partition pour participer à la recherche des solutions optimales locales. Pour coordonner les résultats trouvés par ses derniers, nous allons considérer un deuxième type d'agent, appelé l'agent de coordination CA, dont le but est de participer à la recherche des solutions optimales du problème globales.

Les résultats d'optimisation de chaque partition et le processus de coordination entre le CA et les $DA(s)$ sont expliqués dans la partie suivante.

4.6.4 Résultats d'optimisation

L'automatisation est un avantage intégré des SMA pour résoudre les problèmes d'une manière distributive. Ce travail met en œuvre l'implémentation d'un système multi-agents pour démontrer le mécanisme de communication et de coordination entre les différents types d'agents afin de trouver la conception optimale du problème global. Le mécanisme de coordination proposé a été mis en œuvre en utilisant Eclipse [1]. Eclipse est l'environnement de développement intégré (IDE) couramment utilisé avec l'application JADE. Il est assez facile d'intégrer JADE dans Eclipse de sorte que lorsqu'un agent est exécuté, la plateforme JADE

1. http ://www.eclipse.org/downloads/

fonctionne et déploie l'agent dans l'environnement d'exécution.

Après la spécification complète des sous-problèmes d'optimisation, *CA* informe les deux agents de conception DA_1 et DA_2 à travers MTS que le processus de coordination commence en envoyant *ACLMessage.Request* et le contenu du message est la formulation du problème d'optimisation de chaque *DA*. Après la réception de cette demande, chaque *DA* informe son agent physique de la formulation proposée. L'agent physique effectue une optimisation locale de sa partition spécifique pour déterminer les solutions optimales par rapport à ses objectifs internes afin de respecter les exigences demandées.

Chaque partition est caractérisée par un optimiseur local et un modèle d'analyse. Les deux optimisations sont basées sur les paramètres et les variables du véhicule électrique indiquées dans le *Tableau 4.2*. Ces deux optimisations sont multi-objectifs de telle sorte que chaque partition n'a pas une seule solution mais un ensemble de solutions appelé front de Pareto comme il est indiqué dans les *Figures 4.29* et *4.30*. Chaque point du front de Pareto est caractérisé par un vecteur d'entrée (variables de conception à optimiser) et un vecteur de sortie (les objectifs à atteindre et les contraintes à respecter).

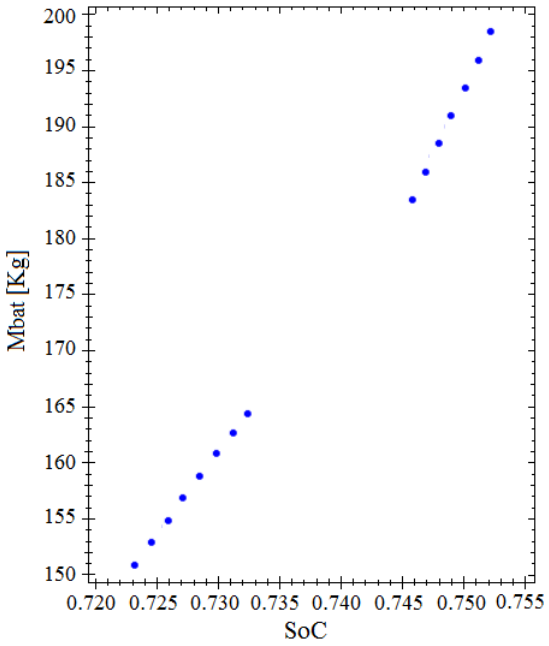


FIGURE 4.29 – Front de Pareto pour DA_1.

Les deux optimisations locales sont effectuées avec succès. Les meilleures solutions obtenues par chaque agent physique *DA* sont enregistrées dans un fichier Excel. Pour récupérer ces points, le rôle de chaque *DA* est de déterminer l'emplacement du son fichier Excel. L'étape suivante consiste à transformer le fichier Excel en Jtable (tableau java). Chaque Jtable est constitué par un certain nombre des cellules et chaque cellule est caractérisée par un entête et des données à récupérer à partir de son fichier Excel. Pour connaitre le nombre des points le

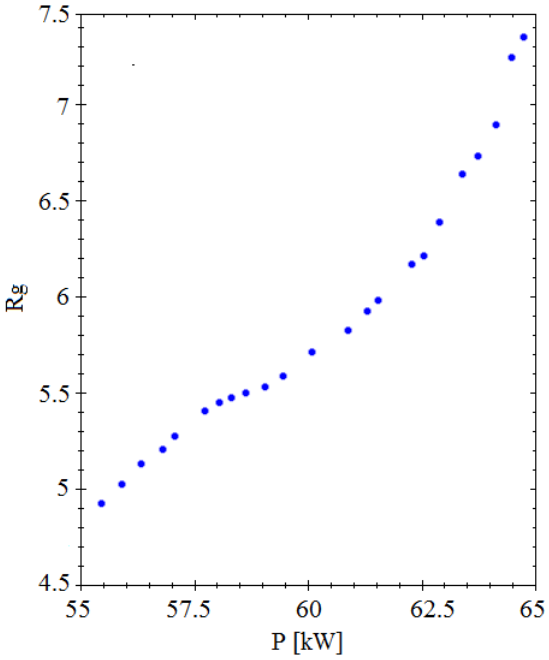


FIGURE 4.30 – Front de Pareto pour DA_2.

long de chaque front de Pareto, il suffit de déterminer le nombre de lignes de chaque tableau. Dans notre cas, le nombre des points de Pareto est 15 pour DA_1 et 23 pour DA_2. Pour savoir le domaine de validité des différentes variables (variables de conceptions, variables de couplages, fonctions objectifs, contraintes) il suffit de déterminer la valeur minimale et maximale de chaque cellule (chaque colonne) du Jtable.

Après la génération des fronts de Pareto, chaque *DA* communique avec *CA* en envoyant *ACLMessage.AGREE* pour l'informer que l'optimisation est effectuée avec succès. Le contenu du message est le domaine de validité de l'ensemble des solutions trouvées. Les meilleures solutions de conception obtenues par les $DA(s)$ sont synthétisées dans le *Tableau 4.5*.

L'optimisation réalisée par DA_2 montre que la puissance maximale requise par le moteur P est comprise entre 55,4 kW et 64,8 kW, une variation de ΔP est d'environ 14,5%. Le rapport de réduction de la boîte de vitesses Rg varie entre 4,92 et 7,36, une variation ΔR_g est d'environ 33,15%. Les deux critères varient donc fortement le long du front optimal, ce qui signifie qu'il existe un compromis entre les deux. Si on trouve que l'un des critères est montrant une variation significative par rapport à l'autre, par exemple, ΔP négligeable devant ΔR_g, cela signifie qu'il est possible d'améliorer le critère de la puissance maximale requise par le moteur P sans avoir un impact important sur le critère de rapport de réduction R_g. Le compromis existe dans ce cas de DA_2 et aussi dans le cas de DA_1. Le choix d'une solution sur les fronts est difficile. Et par conséquent, il est très difficile de déterminer les valeurs des variables de conception de DA_1 avec celles de DA_2 qui respectent les exigences demandées afin de trouver les solutions optimales du problème global. L'avantage de l'approche multi-agents est de définir des algorithmes au cours de processus de coordination permettant à *CA*

	Résultats après optimisation	
	N_s	75 - 82
	N_p	4 - 5
	$I(A)$	142.1 - 152.5
	$U(V)$	387.9- 416.1
DA_1	$M_{bat}(Kg)$	150.8 - 198.4
	SoC	0.723 - 0.752
	Temps de calcul (min)	14.56
	Nombre de solutions	15
	$Emf(Nm/A)$	0.214 - 0.348
	R_g	4.92- 7.36
	$I(A)$	142.6 - 151.6
	$U(V)$	392.1- 419.3
DA_2	$P(kW)$	55.4 –64.8
	$V_{10}(km/h)$	56.5– 63.8
	$V_{max}(km/h)$	116.2 – 123.5
	Temps de calcul (min)	21.36
	Nombre de solutions	23

Tableau 4.5 – Les résultats d'optimisation des $DA(s)$

de prendre des décisions dans ces cas de conflits.

Dans l'exemple du véhicule électrique, il est inutile d'appliquer l'algorithme 3.1 pour spécifier le lien à parcourir. En effet, le problème d'optimisation est formé par deux agents de conception et par conséquent il y a un seul lien à parcourir. Le premier algorithme utilisé par *CA* est l'algorithme 3.2 qui consiste à déterminer le minimum des maximums d'intervalles et le maximum des minimums d'intervalles de chaque variable de coordination afin de limiter l'espace de recherche des solutions optimales. Pour implémenter cet algorithme, tout d'abord, il suffit de récupérer les bornes (les limites inférieures et supérieures) de U et I envoyées par DA_1 et DA_2. Ensuite, il faut ajouter ces deux commandes mathématiques : *Math.max ()* et *Math.min ()* dans la classe de l'agent de coordination. Dans notre cas, cela donne ($392.1 \leq U \leq 416.1$, $142.6 \leq I \leq 151.6$). La *Figure 4.31* montre la mise en œuvre de cet algorithme dans l'environnement de développement JADE avec Eclipse. Les agents sont déployés dans le conteneur principal. Jtable 1 et 2 sont les tableaux de solutions de DA_1 et DA_2 respectivement.

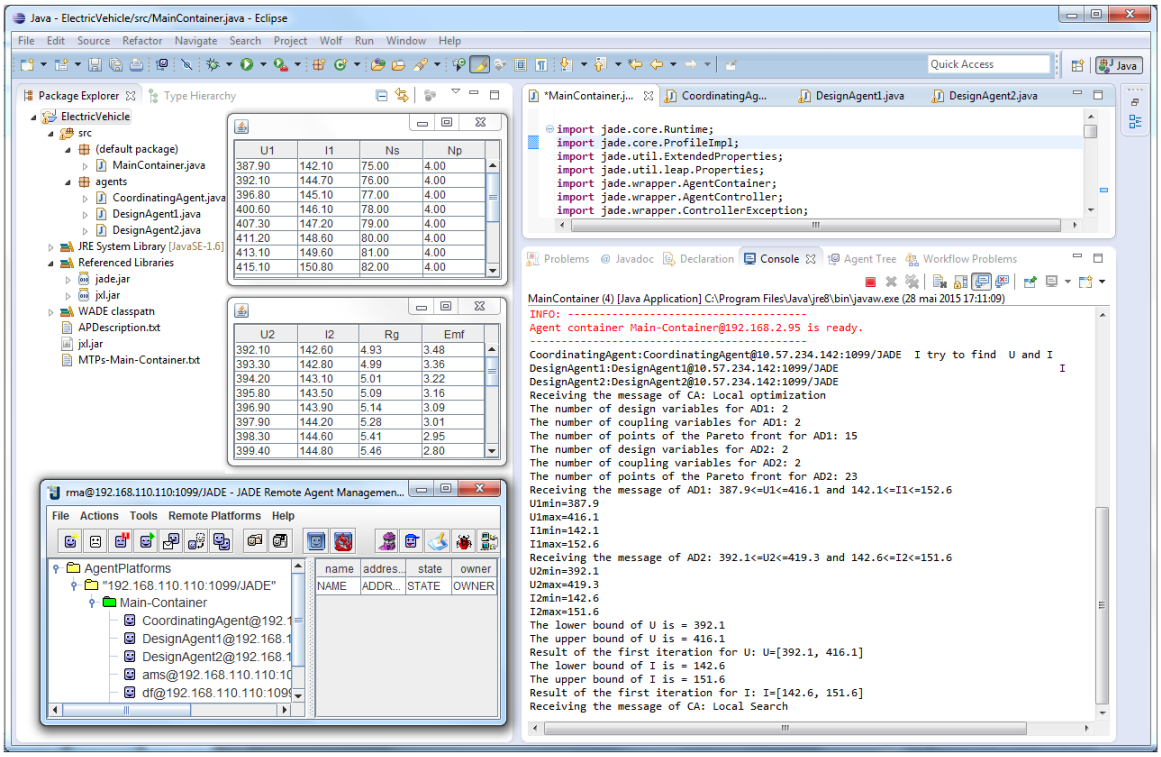


FIGURE 4.31 – La mise en œuvre de l'algorithme de limitation de l'espace de recherche dans l'environnement de développement JADE avec Eclipse.

Le deuxième algorithme de coordination utilisé par *CA* est l'alghorithme 3.3 qui consiste à coordonner les variables de couplages entre les différents agents de conception afin de trouver la cohérence entre les bonnes valeurs des variables de conception qui respectent les performances globales du système. Dans notre cas, l'algorithme 3.2 utilisé par *CA* pour déterminer les valeurs des variables de conception de DA_1 et DA_2 est donné par :

Algorithme 4.1 *Algorithme de détermination des solutions optimales entre* DA_1 *et* DA_2

---

for $i = 1$ *to* 15
  for $j = 1$ *to* 23
    if $\|U_1(i) - U_2(j)\|^2 \leq 1$ and $\|I_1(i) - I_2(j)\|^2 \leq 1$ **then**
      Return $(Ns(i),\ Np(i))$ of DA_1 and $(Emf(j),\ Rg(j))$ of DA_2
    end if
  end for
end for

Où
(U_1, I_1) et (U_2, I_2) sont les valeurs des variables de couplages de DA_1 et DA_2 respectivement.

Pour implémenter cet algorithme, *CA* demande à DA_1 et DA_2, en envoyant *ACLMessage.Request*, de déterminer l'ensemble des points (solutions des fronts de pareto) dont $392.1 \leq U \leq 416.1$ et $142.6 \leq I \leq 151.6$. Après la réception du message de *CA*, chaque *DA* effectue une recherche locale à l'intérieur de son tableau Jtable pour déterminer tous les points demandés par *CA*. Dans ce cas, le nombre de solutions trouvées est 13 pour DA_1 et 19 pour DA_2. Chaque *DA* communique avec *CA* en envoyant *ACLMessage.Inform* et le contenu du message est l'ensemble des solutions trouvées.

Après la réception des messages envoyés par DA_1 et DA_2, *CA* enregistre ces solutions dans un Jtable. Dans ce cas, le rôle de *CA* est de déterminer tous les points du Jtable dont $\|U_1(i) - U_2(j)\|^2 \leq 1$ et $\|I_1(i) - I_2(j)\|^2 \leq 1$. Pour chaque point trouvé, *CA* retourne les valeurs des variables de conception (N_s, N_p) de DA_1 et (R_g, Emf) de DA_2. La *Figure 4.32* représente l'implémentation du deuxième algorithme pour la détermination des solutions communes entre DA_1 et DA_2.

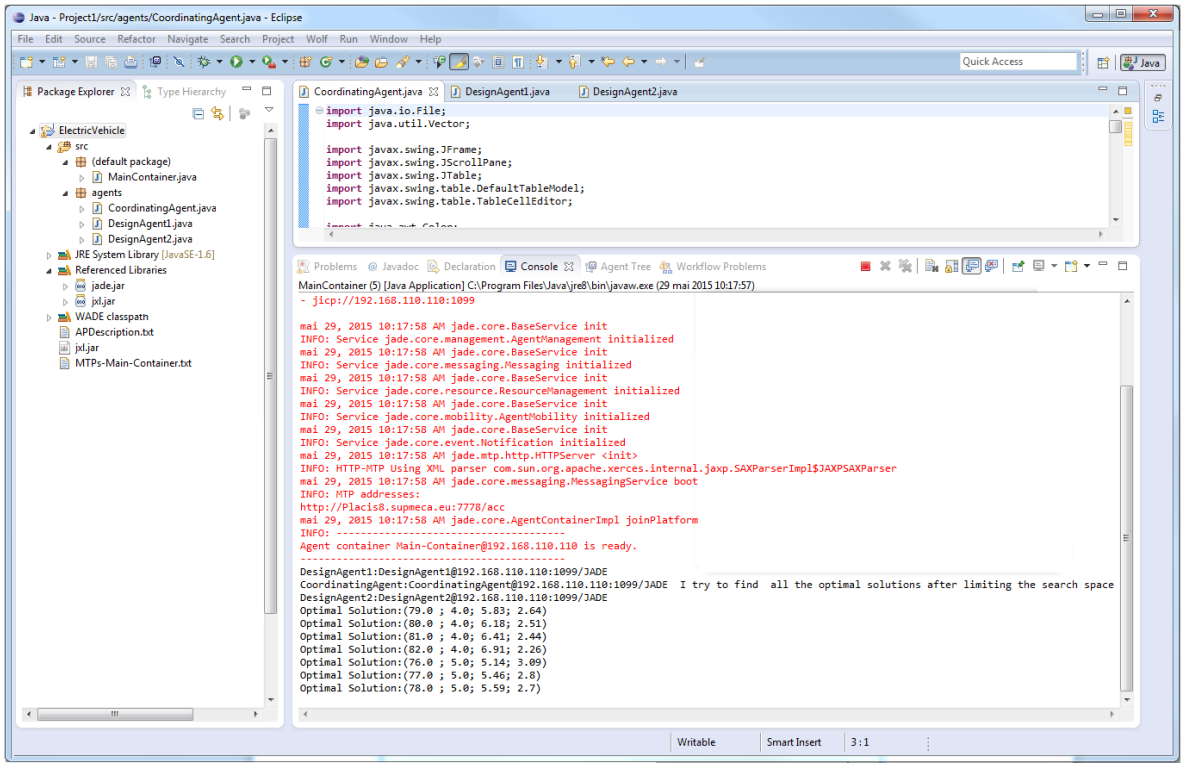


FIGURE 4.32 – Les solutions optimales répondant aux exigences demandées par DA_1 et DA_2.

L'implémentation de l'algorithme 4.1 nous a permis de déterminer 7 solutions optimales dont les valeurs sont indiquées dans le *Tableau 4.6*.

La *Figure 4.33* représente la variation de la vitesse du véhicule pour les 4 premières solutions optimales pour lesquelles nous avons $57.1 \leq V_{10} \leq 62.5$ and $118.3 \leq V_{max} \leq 122.4$.

Solutions	N_s	N_p	Emf	R_g
1	79	4	0.264	5.83
2	80	4	0.251	6.18
3	81	4	0.244	6.41
4	82	4	0.226	6.91
5	76	5	0.309	5.14
6	77	5	0.280	5.46
7	78	5	0.270	5.59

Tableau 4.6 – Solutions optimales de l'approche multi-agent

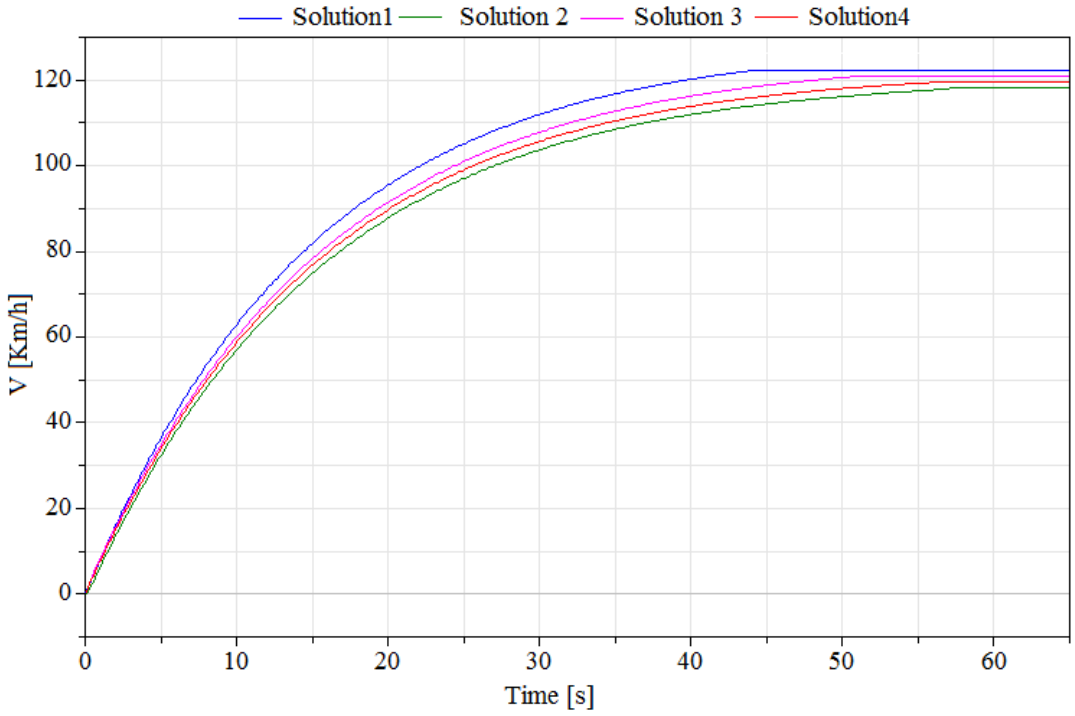


FIGURE 4.33 – La vitesse du véhicule pour les configurations optimales.

Pour valider notre approche, une étude comparative avec les méthodes classiques d'optimisation est nécessaire. Pour cela, nous avons utilisé la méthode AAO pour résoudre la 1 ère configuration de la partie précédente où toutes les fonctions objectifs sont associées à une seule partition. Cette configuration est caractérisée par un seul modèle d'analyse développé dans la *Figure 4.5*. Les résultats d'optimisation de cette configuration sont indiqués dans le *Tableau 4.7*.

	AAO
N_s	74 - 83
N_p	4 - 5
$Emf(Nm/A)$	0.214 - 0.413
R_g	4.52- 7.76
$M_{bat}(Kg)$	150.8 - 198.4
SoC	0.722 - 0.756
$P(kW)$	53.4 – 66.3
$V_{10}(km/h)$	55.1– 64.6
$V_{max}(km/h)$	115.2 – 123.6
Temps de calcul (min)	43.22
Nombre de solutions	10

Tableau 4.7 – Résultats de l'optimisation de la méthode AAO

Les deux critères sélectionnés ici pour comparer l'approche multi-agents avec la méthode AAO sont : le nombre total des solutions optimales et le temps de calcul global. L'approche muti-agents nous a permis de déterminer 7 solutions dans 23.36 min (c'est le maximum du temps de calcul des différentes partitions de la configuration choisie) alors qu'avec la méthode AAO nous avons détecté 10 solutions dans 43 min. En comparant ces résultats d'optimisation, nous pouvons en déduire que 70% du nombre total de solutions sont déterminées avec une réduction importante de 45% au niveau du temps de calcul global.

4.7 Conclusion

Dans ce chapitre, nous avons développé un modèle du véhicule électrique à partir duquel nous avons estimé l'état de charge de la batterie ainsi que la quantité d'énergie nécessaire pour parcourir une certaine distance. Nous avons utilisé ce modèle comme un cas d'application pour valider notre approche multi-agents. Nous avons décomposé le problème de conception du véhicule électrique en deux agents de conception et nous avons attribué à chaque agent ses objectifs et ses contraintes. Un agent de coordination est considéré pour coordonner les résultats trouvés par les agents de conception afin de parvenir à des décisions optimales. Nous avons montré que l'approche proposée permet aux agents de suivre un processus de conception afin de faciliter la coordination de la conception collaborative distribuée. Les résultats optimaux ont été trouvés et les résultats globaux de simulation ont été présentés.

En outre, nous avons décrit la mise en œuvre et la programmation de ces deux types d'agents en utilisant la plate-forme JADE avec Eclipse. Nous avons constaté que JADE est

tout à fait approprié pour le développement de systèmes multi-agents, car il supporte de nombreux concepts d'agents tels que la communication, le protocole et le comportement.

Conclusion générale et perspectives

Conclusion générale et perspectives

Une conception mécatronique typique implique des équipes multi-disciplinaires de travailler ensemble à la conception d'un système mécatronique. Les défis associés à la conception collaborative des équipes multi-disciplinaires nécessitent de diviser le travail en petits problèmes de conception plus facile à gérer et de coordonner les différents sous-problèmes de conception afin de trouver la conception optimale du problème global.

Notre principale contribution dans cette thèse est de proposer une approche multi-agents pour la conception optimale des systèmes mécatroniques distribués. Dans cette approche, on a utilisé deux types d'agents logiciels, un agent de coordination *CA* et des agents de conceptions $DA(s)$, qui communiquent avec des agents physiques (concepteurs, ingénieurs, etc) qui sont en relation directe avec les outils de conception. Le rôle de *CA* est de participer à la recherche des solutions optimales globales, alors que chaque *DA* est responsable à une partition locale spécifique pour participer à la recherche des solutions optimales locales.

L'approche proposée est basée sur un processus de coordination pour coordonner les résultats trouvés en cours des optimisations effectuées par les agents de conception. Cette approche a été appliquée au cas de la conception préliminaire d'un véhicule électrique pour illustrer comment l'utilisation du mécanisme de coordination permet aux concepteurs de faciliter la conception collaborative distribuée et de parvenir à une décision optimale du problème global.

Dans le premier chapitre, nous avons commencé par établir le cadre général de ce travail de thèse. Pour cela, nous avons présenté un bref aperçu des systèmes mécatroniques (historique, définition, applications et conception). Ensuite, nous avons abordé le processus de conception de ces systèmes en mettant l'accent sur les méthodes classiques de conception utilisées. Aussi, nous avons présenté les outils de modélisation et de simulation qui ont été l'origine du développement de ce type de système. Enfin, nous avons discuté quelques enjeux liés à la conception mécatronique.

Dans le deuxième chapitre, nous avons présenté une synthèse des approches existantes, telles que les approches MDO et les approches MOO, pour supporter les enjeux liés à la conception des systèmes mécatroniques, plus particulièrement la problématique d'optimisation. Nous avons discuté les avantages et les limites de chaque approche. Nous avons montré que l'utilisation de l'une de ces approches pour l'optimisation d'un problème multi-disciplinaire est coûteuse en temps de calcul puisqu'elle nécessite un grand nombre d'éva-

luations des fonctions objectifs pour atteindre une solution optimale. Afin de surmonter le problème de coût de calcul, nous avons parlé dans la dernière partie de ce chapitre de la technique de substitution qui consiste à remplacer un grand nombre d'évaluations par des approximations construites à partir des modèles simples appelés méta-modèles ou modèles de substitution.

Afin de surmonter les limites des approches classiques d'optimisation, nous avons développé dans le troisième chapitre une nouvelle approche basée sur les paradigmes multi-agents pour la conception optimale des systèmes mécatroniques.

Dans le quatrième chapitre, nous avons appliqué l'approche muti-agents sur le cas de conception préliminaire d'un véhicule électrique. Nous avons validé les résultats trouvés de l'approche proposée par comparaison avec les résultats obtenus en utilisant les approches classiques d'optimisation.

Avantages et limites de l'approche multi-agents

Par rapport aux techniques classiques MDO, l'approche multi-agents nous permet :

- d'intégrer les exigences du processus d'optimisation au niveau de conception,
- de simplifier la complexité d'une optimisation de la conception mécatronique,
- de réduire le temps de calcul,
- de définir des règles qui aident à prendre des décisions au cours du processus d'optimisation des systèmes mécatroniques.

Par contre, l'inconvénient majeur de cette approche, est que son implémentation est un peu difficile et nécessite de définir une politique de partage d'informations et de coordination.

Perspectives

Les résultats déterminés dans ce travail offrent des perspectives intéressantes de développement futur. Les recherches futures pourront se concentrer sur :

- La récupération des données via un format XML d'une configuration d'optimisation représentée par un bloc SysML de type IBD afin d'évaluer automatiquement le critère C_1.

- L'utilisation d'une base de données pour enregistrer et accéder aux résultats d'optimisation trouvés par les différents agents de conception.

- L'adaptation de l'algorithme du théorie des jeux (Gamme theory) ainsi que l'algorithme des colonies de fourmis (Ant colony) pour coordonner les résultats trouvés au cours des optimisations effectuées par les agents de conception.

Bibliographie

[1] M. Miladi Chaabane. *Modélisation géométrique et mécanique pour les systèmes mécatroniques*. PhD thesis, Châtenay-Malabry, Ecole centrale de Paris, 2014.

[2] C. Haskins and K. Forsberg. Systems engineering handbook : A guide for system life cycle processes and activities. In *International Council on Systems Engineering*. Incose, 2007.

[3] H. Trabelsi. *Contribution à la prise en compte d'exigences dynamiques en conception préliminaire de systèmes complexes*. PhD thesis, Châtenay-Malabry, Ecole centrale de Paris, 2014.

[4] J-P. Jamont. *DIAMOND : Une approche pour la conception de systèmes multi-agents embarqués*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2005.

[5] B.W. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5) :61–72, 1988.

[6] M. Tahan, A. Touil, J. Vareille, and P. Le Parc. La méthode de développement en x, un autre point de vue sur le cycle de vie. *20ème Congrès Français de Mécanique, 28 août/2 sept. 2011-25044 Besançon, France (FR)*, 2011.

[7] M. Hammadi. *Contribution à l'intégration de la modélisation et la simulation multi-physique pour conception des systèmes mécatroniques*. PhD thesis, Châtenay-Malabry, Ecole centrale de Paris, 2012.

[8] Modelica Association. Modelica : Modeling of complex physical systems. *URL http ://www. modelica. org*, 1997.

[9] K. Ejjabraoui. *Contribution à la conception de systèmes mécatroniques automobiles : méthodologie de prédimensionnement multi-niveau multiphysique de convertisseurs de puissance*. PhD thesis, SUPELEC, Gif sur Yvette, France, 2010.

[10] A. Berro. *Optimisation multi-objectif et strategie d evolution en environnement dynamique*. PhD thesis, 2001.

[11] J-B. Welcomme. *MASCODE : un système multi-agent adaptatif pour concevoir des produits complexes. Application à la conception préliminaire avion*. PhD thesis, Université Paul Sabatier, 2008.

[12] M. H. Bonte. *Optimisation strategies for metal forming processes*. University of Twente, 2007.

[13] K. Zidi. *Système interactif d'aide au déplacement multimodal (SIADM)*. PhD thesis, Ecole Centrale de Lille ; Université des Sciences et Technologie de Lille-Lille I, 2006.

[14] R.H. Bishop. *The Mechatronics Handbook*. CRC Press, 2014.

[15] O. Penas, R. Plateaux, J-Y. Choley, and A. Riviere. The different complexity levels in mechatronic design process. In *3rd International Conference on Software, Knowledge, Information Management and Applications SKIMA, Fès, Morocco*, 2009.

[16] T. More. Mechatronics. Technical report, Yaskawa Internal Trademark Application Memo 21.131. 01, 1969.

[17] N. Kyura. The development of a controller for mechatronics equipment. *Industrial Electronics, IEEE Transactions on*, 43(1) :30–37, 1996.

[18] VDI Guideline. 2206, design methodology for mechatronic systems, 2003.

[19] N. Kyura and H. Oho. Mechatronics-an industrial perspective. *Mechatronics, IEEE/ASME Transactions on*, 1(1) :10–15, 1996.

[20] M. Tomizuka. Mechatronics : from the 20th to 21st century. *Control engineering practice*, 10(8) :877–886, 2002.

[21] K. Craig. Mechatronic system design. In *Proceedings of the Motor, Drive & Automation Systems Conference*, 2009.

[22] S. Turki. *Ingénierie système guidée par les modèles : Application du standard IEEE 15288, de l'architecture MDA et du langage SysML à la conception des systèmes mécatroniques*. PhD thesis, Université du Sud Toulon Var, 2008.

[23] G. Rzevski. On conceptual design of intelligent mechatronic systems. *Mechatronics*, 13(10) :1029–1044, 2003.

[24] B. Canbaz. *Preventing and resolving design conflicts for a collaborative convergence in distributed set-based design*. PhD thesis, Ecole Centrale Paris, 2013.

[25] Y-E. Nahm and H. Ishikawa. Integrated product and process modeling for collaborative design environment. *Concurrent Engineering*, 12(1) :5–23, 2004.

[26] G. Pahl, W. Beitz, J. Feldhusen, and K-H. Grote. *Engineering design : a systematic approach*, volume 157. Springer Science & Business Media, 2007.

[27] C. Espanet. *Modélisation et conception optimale de moteurs sans balais à structure inversée-Application au moteur-roue*. PhD thesis, Université de Franche-Comté, 1999.

[28] J-P. Meinadier. *Ingénierie et intégration des systèmes*. Hermes, 1998.

[29] S. Arnold. Iso 15288 systems engineering :system life cycle processes. *International Standards Organisation*, 2002.

[30] W.W. Royce. Managing the development of large software systems. In *proceedings of IEEE WESCON*, volume 26. Los Angeles, 1970.

[31] P. Joore. The v-cycle for system innovation translating a broad societal need into concrete product service solutions : the multifunctional centre apeldoorn case. *Journal of Cleaner Production*, 16(11) :1153–1162, 2008.

[32] N.D. Birrell and M. A. Ould. *A practical handbook for software development*. Cambridge University Press, 1988.

[33] L.L. Ray. Security considerations for the spiral development model. *International Journal of Information Management*, 33(4) :684–686, 2013.

[34] S. Friedenthal, A. Moore, and R. Steiner. *A practical guide to SysML : the systems modeling language*. Morgan Kaufmann, 2014.

[35] F. Christophe, A. Bernard, and É. Coatanéa. Rfbs : A model for knowledge representation of conceptual design. *CIRP Annals-Manufacturing Technology*, 59(1) :155–158, 2010.

[36] E. Andrade, P. Maciel, B. Nogueira, C. Araújo, and G. Callou. A cots-based approach for estimating performance and energy consumption of embedded real-time systems. *Information Processing Letters*, 110(14) :525–534, 2010.

[37] A. Pop, D. Akhvlediani, and P. Fritzson. Towards unified system modeling with the modelicaml uml profile. In *EOOLT*, pages 13–24. Citeseer, 2007.

[38] H.M. Paynter. *Analysis and design of engineering systems*. MIT press, 1961.

[39] H. Elmqvist, S.E. Mattsson, and M. Otter. Modelica : The new object-oriented modeling language. In *12th European Simulation Multiconference, Manchester, UK*, 1998.

[40] Mathworks Simulink. Simulation and model-based design, 2004.

[41] C.B. Moler. *MATLAB–an interactive matrix laboratory*. Department of Mathematics and Statistics, University of New Mexico, 1980.

[42] H. Zhang, H. Wang, D. Chen, and G. Zacharewicz. A model-driven approach to multidisciplinary collaborative simulation for virtual product development. *Advanced Engineering Informatics*, 24(2) :167–179, 2010.

[43] T. Touya. *Méthodes d'optimisation pour l'espace et l'environnement*. PhD thesis, Université Paul Sabatier-Toulouse III, 2008.

[44] A. Guizani, M. Hammadi, J-Y. Choley, T. Soriano, M.S. Abbes, and M. Haddar. Multidisciplinary approach for optimizing mechatronic systems : Application to the optimal design of an electric vehicle. In *Advanced Intelligent Mechatronics (AIM), 2014 IEEE/ASME International Conference on*, pages 56–61. IEEE, 2014.

[45] O. Mouelhi. *Contribution à l optimisation multiobjectif en conception multidisciplinaire*. PhD thesis, INSA de Lyon, 2010.

[46] V. Pareto. *Cours d'économie politique*. Librairie Droz, 1964.

[47] Y. Collette and P. Siarry. *Optimisation multiobjectif*. Editions Eyrolles, 2002.

[48] J. Dréo, A. Pétrowski, É.D. Taillard, and P. Siarry. Métaheuristiques pour l'optimisation difficile. 2003.

[49] C. Solnon and K. Ghédira. Ant colony optimization for multi-objective optimization problems. *Internation Journal on computer science*, 2010.

[50] L. Cagnina, S.C. Esquivel, and C. Coello Coello. A particle swarm optimizer for multi-objective optimization. *Journal of Computer Science & Technology*, 5, 2005.

[51] K. Klamroth and K. Miettinen. Integrating approximation and interactive decision making in multicriteria optimization. *Operations Research*, 56(1) :222–234, 2008.

[52] R. Tappeta, J. Renaud, and J. Rodríguez. An interactive multiobjective optimization design strategy for decision based multidisciplinary design. *Engineering Optimization*, 34(5) :523–544, 2002.

[53] Z. Ren, F. Yang, N.M. Bouchlaghem, and C.J. Anumba. Multi-disciplinary collaborative building design : A comparative study between multi-agent systems and multi-disciplinary optimisation approaches. *Automation in Construction*, 20(5) :537–549, 2011.

[54] E.J. Cramer, J.E. Dennis, P.D. Frank, R.M. Lewis, and G.R. Shubin. Problem formulation for multidisciplinary optimization. *SIAM Journal on Optimization*, 4(4) :754–776, 1994.

[55] F. Flager and J. Haymaker. A comparison of multidisciplinary design, analysis and optimization processes in the building construction and aerospace industries. In *24th international conference on information technology in construction*, pages 625–630. Citeseer, 2007.

[56] R.J. Balling and J. Sobieszczanski-Sobieski. Optimization of coupled systems-a critical overview of approaches. *AIAA journal*, 34(1) :6–17, 1996.

[57] N.F. Brown and J.R. Olds. Evaluation of multidisciplinary optimization techniques applied to a reusable launch vehicle. *Journal of Spacecraft and Rockets*, 43(6) :1289–1300, 2006.

[58] P. Neittaanmäki, T. Rossi, S. Korotov, E. Oñate, J. Périaux, and D. Knörzer. Multidisciplinary optimisation of a supersonic transport aircraft wing planform. *ONERA-Applied Aerodynamics Department*, 2004.

[59] S. Kodiyalam. Evaluation of methods for multidisciplinary design optimization (mdo). phase 1. 1998.

[60] N. M. Alexandrov and R.M. Lewis. Comparative properties of collaborative optimization and other approaches to mdo. 1999.

[61] N. M. Alexandrov and R.M. Lewis. Analytical and computational aspects of collaborative optimization for multidisciplinary design. *AIAA journal*, 40(2) :301–309, 2002.

[62] N. M. Alexandrov and R.M. Lewis. Algorithmic perspectives on problem formulations in mdo. In *Proceedings of the 8th AIAA= USAF= NASA= ISSMO Symposium on MA & O, Long Beach, CA, AIAA*, 2000.

[63] N. Duranté, A. Dufour, V. Pain, G. Baudrillard, and M. Schoenauer. Multidisciplinary analysis and optimisation approach for the design of expendable launchers. In *10th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2004.

[64] M. Masmoudi and Y.S. Parte. Disciplinary interaction variable elimination (dive) approach for mdo. In *ECCOMAS CFD 2006 : Proceedings of the European Conference on Computational Fluid Dynamics, Egmond aan Zee, The Netherlands, September 5-8, 2006*. Delft University of Technology ; European Community on Computational Methods in Applied Sciences (ECCOMAS), 2006.

[65] R.F. Coelho and P. Breitkopf. Optimisation multidisciplinaire en mécanique : Réduction de modèles, robustesse, fiabilité, réalisations logicielles, 2009.

[66] J. Clément. *Optimisation multidisciplinaire : étude théorique et application à la conception des avions en phase d'avant projet*. PhD thesis, 2009.

[67] K.F. Hulme and C.L. Bloebaum. A simulation-based comparison of multidisciplinary design optimization solution strategies using cascade. *Structural and Multidisciplinary Optimization*, 19(1) :17–35, 2000.

[68] J.R. Martins and A.B. Lambe. Multidisciplinary design optimization : a survey of architectures. *AIAA journal*, 51(9) :2049–2075, 2013.

[69] G.R. Shubin. Application of alternative multidisciplinary optimization formulations to a model problem for static aeroelasticity. *Journal of Computational Physics*, 118(1) :73–85, 1995.

[70] R.D. Braun, I.M. Kroo, and A.A. Moore. Use of the collaborative optimization architecture for launch vehicle design. In *Proceedings of the 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1996.

[71] J. Agte. A tool for application of bi-level integrated system synthesis to multidisciplinary design optimization problems. In *German Air and Space Congress*, 2005.

[72] S. Yi, J-K. Shin, and G.J. Park. Comparison of mdo methods with mathematical examples. *Structural and Multidisciplinary Optimization*, 35(5) :391–402, 2008.

[73] R. D. Braun. Collaborative optimization : an architecture for large-scale distributed design. 1996.

[74] S. Parashar and C.L. Bloebaum. Multi-objective genetic algorithm concurrent subspace optimization (mogacsso) for multidisciplinary design. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, pages 2006–2047, 2006.

[75] C-H. Huang, J. Galuski, and C.L. Bloebaum. Multi-objective pareto concurrent subspace optimization for multidisciplinary design. *AIAA journal*, 45(8) :1894–1906, 2007.

[76] J. Sobieszczanski-Sobieski, J.S. Agte, and R.R. Sandusky. Bilevel integrated system synthesis. *AIAA journal*, 38(1) :164–172, 2000.

[77] J. Ahn and J. Kwon. An efficient strategy for reliability-based multidisciplinary design optimization using bliss. *Structural and Multidisciplinary Optimization*, 31(5) :363–372, 2006.

[78] C.D. Maranas and C.A. Floudas. A deterministic global optimization approach for molecular structure determination. *The Journal of chemical physics*, 100(2) :1247–1261, 1994.

[79] M. Locatelli and U. Raber. Packing equal circles in a square : a deterministic global optimization approach. *Discrete Applied Mathematics*, 122(1) :139–166, 2002.

[80] R. Horst and H. Tuy. *Global optimization : Deterministic approaches*. Springer Science & Business Media, 1996.

[81] A.J. Qureshi, J-Y. Dantan, J. Bruyere, and R. Bigot. Set based robust design of mechanical systems using the quantifier constraint satisfaction algorithm. *Engineering Applications of Artificial Intelligence*, 23(7) :1173–1186, 2010.

[82] R.E. Moore. *Interval analysis*, volume 4. Prentice-Hall Englewood Cliffs, 1966.

[83] U. Montanari. Networks of constraints : Fundamental properties and applications to picture processing. *Information sciences*, 7 :95–132, 1974.

[84] B. Faltings. Arc-consistency for continuous variables. *Artificial intelligence*, 65(2) :363–376, 1994.

[85] F. Benhamou and L. Granvilliers. Continuous and interval constraints. *Handbook of constraint programming*, 2 :571–603, 2006.

[86] B. Yannou and G. Harmel. Use of constraint programming for design'in advances in design, 2005.

[87] F. Benhamon, D. McAllester, and P. Van Hentenryck. Clp (intervals) revisited. *Rapport technique, Citeseer*, page 30, 1994.

[88] L. Granvilliers, E. Monfroy, and F. Benhamou. Symbolic-interval cooperation in constraint programming. In *Proceedings of the 2001 international symposium on Symbolic and algebraic computation*, pages 150–166. ACM, 2001.

[89] Y. Meyer and P.A. Yvars. Optimization of a passive structure for active vibration isolation : an interval-computation-and constraint-propagation-based approach. *Engineering Optimization*, 44(12) :1463–1489, 2012.

[90] P.A. Yvars, P. Lafon, and L. Zimmer. Optimization of mechanical system : Contribution of constraint satisfaction method. In *Computers & Industrial Engineering, 2009. CIE 2009. International Conference on*, pages 1379–1384. IEEE, 2009.

[91] S. Tunali and I. Batmaz. Dealing with the least squares regression assumptions in simulation metamodeling. *Computers & industrial engineering*, 38(2) :307–320, 2000.

[92] N. V. Queipo, R.T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P.K. Tucker. Surrogate-based analysis and optimization. *Progress in aerospace sciences*, 41(1) :1–28, 2005.

[93] R.H. Myers, D.C. Montgomery, and C.M. Anderson-Cook. *Response surface methodology : process and product optimization using designed experiments*, volume 705. John Wiley & Sons, 2009.

[94] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing*, 9(1) :3–12, 2005.

[95] I. Batmaz and S. Tunali. Small response surface designs for metamodel estimation. *European Journal of Operational Research*, 145(2) :455–470, 2003.

[96] D.G. Kbiob. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of Chemical, Metallurgical, and Mining Society of South Africa*, 1951.

[97] G. Matheron. *Traité de géostatistique appliquée*. Editions Technip, 1962.

[98] D.R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4) :345–383, 2001.

[99] D.R. Jones, M. Schonlau, and W.J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4) :455–492, 1998.

[100] J. Sack, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4) :409–435, 1989.

[101] M. Ahmed and N. Qin. Comparison of response surface and kriging surrogates in aerodynamic design optimization of hypersonic spiked blunt bodies. In *13th International Conference on Aerospace Sciences and Aviation Technology, May 26th–28th, Military Technical College, Kobry Elkobbah, Cairo, Egypt*, 2009.

[102] C. Badufle, C. Blondel, T. Druot, and M. Duffau. Automatic satisfaction of constraints set in aircraft sizing studies. *6th World Congresses of Structural and Multidisciplinary Optimization (WCSMO 05)*, 2005.

[103] B.M. Brochtrup and J.W. Herrmann. A classification framework for product design optimization. In *ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 13–20. American Society of Mechanical Engineers, 2006.

[104] N. M. Alexandrov and R.M. Lewis. Reconfigurability in mdo problem synthesis, part 1. In *Proceedings of the 10th AIAA/ISSMO multidisciplinary analysis and optimization conference, AIAA paper*, volume 4307, 2004.

[105] J. Ferber and J-F. Perrot. *Les systèmes multi-agents : vers une intelligence collective*. InterEditions, 1995.

[106] N. Jennings, K. Sycara, and M. Wooldridge. Autonomous agents and multi-agents systems. *A roadmap of agent research and development*, 1998.

[107] M. Wooldridge and N.R. Jennings. The cooperative problem-solving process. *Journal of Logic and Computation*, 9(4) :563–592, 1999.

[108] Y.D. Miiller. Decentralized artificial intelligence. *Decentralised AI*, pages 3–13, 1990.

[109] N.J. Nilsson. Distributed artificial intelligence. Technical report, DTIC Document, 1981.

[110] B. Benmammar. Intelligence artificielle et systèmes multi-agents. 2009.

[111] J-P. Müller. Modélisation organisationnelle en systèmes multi-agents. *7eme école d'été de l'Association pour la Recherche Cognitive, Bonas*, 2000.

[112] J. Ferber. Les systèmes multi-agents : un aperçu général. *Techniques et sciences informatiques*, 16(8), 1997.

[113] A.H Bond and L. Gasser. *Readings in distributed artificial intelligence*. Morgan Kaufmann, 2014.

[114] S. Franklin and A. Graesser. Is it an agent, or just a program ? : A taxonomy for autonomous agents. In *Intelligent agents III agent theories, architectures, and languages*, pages 21–35. Springer, 1997.

[115] C.A. Iglesias, M. Garijo, J.C. González, and J.R. Velasco. Analysis and design of multiagent systems using mas-commonkads. In *Intelligent Agents IV Agent Theories, Architectures, and Languages*, pages 313–327. Springer, 1998.

[116] J.Y. Halpern. Reasoning about knowledge : a survey, handbook of logic in artificial intelligence and logic programming (vol. 4) : epistemic and temporal reasoning, 1995.

[117] Y. Shoham. Agent-oriented programming. *Artificial intelligence*, 60(1) :51–92, 1993.

[118] S.R. Thomas. Placa, an agent oriented programming language. 1993.

[119] E. Malville and F. Bourdon. Task allocation : A group self-design approach. In *Multi Agent Systems, 1998. Proceedings. International Conference on*, pages 166–173. IEEE, 1998.

[120] H.S. Nwana and D.T. Ndumu. Agents of change in future communication systems. In *Software Agents for Future Communication Systems*, pages 58–85. Springer, 1999.

[121] J. Allen and C. Perrault. Analyzing intention in dialogues. 1978.

[122] E. Appelt Douglas. Planning english sentences, 1985.

[123] A. Chavez and P. Maes. Kasbah : An agent marketplace for buying and selling goods. In *Proceedings of the first international conference on the practical application of intelligent agents and multi-agent technology*, volume 434. London, UK, 1996.

[124] P.R. Cohen and C.R. Perrault. Elements of a plan-based theory of speech acts*. *Cognitive science*, 3(3) :177–212, 1979.

[125] P.R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial intelligence*, 42(2) :213–261, 1990.

[126] J.R. Galliers. *A theoretical framework for computer models of cooperative dialogue, acknowledging multi-agent conflict*. PhD thesis, Open University, 1988.

[127] F. Kaplan. A new approach to class formation in multi-agent simulations of language evolution. In *Multi Agent Systems, 1998. Proceedings. International Conference on*, pages 158–165. IEEE, 1998.

[128] D.J. Litman and J.F. Allen. A plan recognition model for subdialogues in conversations. *Cognitive science*, 11(2) :163–200, 1987.

[129] W. Wahlster and A. Kobsa. *User models in dialog systems*. Springer, 1989.

[130] N. Negroponte. *Being digital*. Vintage, 1996.

[131] G. Babin, Z. Maamar, and B. Chaib-draa. Metadatabase meets distributed ai. In *Cooperative Information Agents*, pages 138–147. Springer, 1997.

[132] G. Lakemeyer and H.J. Levesque. Query evaluation and progression in aol knowledge bases. In *IJCAI*, pages 124–131, 1999.

[133] Y. Lesperance. A formal theory of indexical knowledge and action. 1992.

[134] Y. Lesperance, H.J. Levesque, F. Lin, D. Marcu, R. Reiter, and R.B. Scherl. A logical approach to high-level robot programming–a progress report. Control of the Physical World by Intelligent Agents, Papers from the 1994 AAAI Fall Symposium, 1994.

[135] B. Chaib-draa. Industrial applications of distributed ai. *Communications of the ACM*, 38(11) :49–53, 1995.

[136] N. Jennings and M.J. Wooldridge. *Agent technology : foundations, applications, and markets*. Springer Science & Business Media, 1998.

[137] J. L. Austin. *How to do things with words*, volume 367. Oxford university press, 1975.

[138] J.R. Searle. *Speech acts : An essay in the philosophy of language*. Cambridge university press, 1969.

[139] M. Fernandez-Lopez and O. Corcho. *Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Publishing Company, Incorporated, 2010.

[140] F. Bellifemine, A. Poggi, and G. Rimassa. Jade–a fipa-compliant agent framework. In *Proceedings of PAAM*, volume 99, page 33. London, 1999.

[141] O. Gutknecht and J. Ferber. Madkit : A generic multi-agent platform. In *Proceedings of the fourth international conference on Autonomous agents*, pages 78–79. ACM, 2000.

[142] S. Luke, C. Cioffi-Revilla, L. Panait, and K. Sullivan. Mason : A new multi-agent simulation toolkit. In *Proceedings of the 2004 swarmfest workshop*, volume 8, page 44, 2004.

[143] D.J. Pate, J. Gray, and B.J. German. A graph theoretic approach to problem formulation for multidisciplinary design analysis and optimization. *Structural and Multidisciplinary Optimization*, 49(5) :743–760, 2014.

[144] E. Ghotbi. Bi-and multi level game theoretic approaches in mechanical design. 2013.

[145] S. Tosserams, L. Etman, and J.E. Rooda. Augmented lagrangian coordination for distributed optimal design in mdo. *International journal for numerical methods in engineering*, 73(13) :1885–1910, 2008.

[146] S. Tosserams, L. Etman, P.Y. Papalambros, and J.E. Rooda. An augmented lagrangian relaxation for analytical target cascading using the alternating direction method of multipliers. *Structural and multidisciplinary optimization*, 31(3) :176–189, 2006.

[147] A. Rousset, B. Herrmann, and C. Lang. Etude comparative des plateformes paralleles pour systemes multi-agents. In *ComPAS 2014 : conférence en parallélisme, architecture et systèmes*, 2014.

[148] R. Schmitz. La voiture electrique, comme alternative realiste a la voiture a essence. pages 1–20, Juin 2007.

[149] P. Courbe. Véhicules électriques ? changer de mobilité, pas de voiture. *Fédération Inter-Environnement Wallonie, Namur*, 2010.

[150] K. Jaber, A. Fakhfakh, and R. Neji. Modeling and simulation of high performance electrical vehicle powertrains in vhdl-ams. *ELECTRIC VEHICLES–MODELLING AND SIMULATIONS*, page 25, 2011.

[151] E. Schaltz. *Electrical vehicle design and modeling*. INTECH Open Access Publisher, 2011.

[152] A. Guizani, M. Hammadi, J-Y. Choley, T. Soriano, M.S. Abbes, and M. Haddar. Multidisciplinary optimization of mechatronic systems : Application to an electric vehicle. In *Mechatronic Systems : Theory and Applications*, pages 1–14. Springer, 2014.

[153] M. Hammadi, J-Y. Choley, O. Penas, and A. Riviere. Multidisciplinary approach for modelling and optimization of road electric vehicles in conceptual design level. In *Electrical Systems for Aircraft, Railway and Ship Propulsion (ESARS), 2012*, pages 1–6. IEEE, 2012.

[154] Phoenix integration. *URL http ://www.phoenix-int.com/software/phx-modelcenter.php*.

[155] D.W. Gao, C. Mi, and A. Emadi. Modeling and simulation of electric and hybrid vehicles. *Proceedings of the IEEE*, 95(4) :729–745, 2007.

[156] F.L. Mapelli, D. Tarsitano, and M. Mauri. Plug-in hybrid electric vehicle : Modeling, prototype realization, and inverter losses reduction analysis. *Industrial Electronics, IEEE Transactions on*, 57(2) :598–607, 2010.

[157] J. Regnier. Conception de systèmes hétérogènes en génie electrique par optimisation évolutionnaire multicritère. 2003.

[158] E. Schaltz. *Design of a fuel cell hybrid electric vehicle drive system*. Department of Energy Technology, Aalborg University, 2010.

[159] N. Janiaud. *Modélisation du système de puissance du véhicule électrique en régime transitoire en vue de l'optimisation de l'autonomie, des performances et des coûts associés.* PhD thesis, Supélec, 2011.

[160] R. Dolecek, J. Novak, and O. Cerny. Traction permanent magnet synchronous motor torque control with flux weakening. *Radioengineering*, 18(4) :601–605, 2009.

[161] S. Guenidi. *Modélisation, commande et gestion de l'énergie d'un véhicule électrique hybride*. PhD thesis, Ecole nationale supérieure polytechnique, 2011.

[162] R.S. Eddine. Commande de machine électrique en environnement matlab/simulink en temps réel, application à la machine asynchrone : commande vectorielle sans capteurs mécaniques svpwm, mode glissant, mras. *mémoire de magister en electrotechnique, université de Constantine*, 2009.

[163] [online] http ://www.trekmag.com/test-les-differents-types-batteries-rechargeables.

[164] [online] http ://voiture-electrique.durable.com/a-les-differentes-technologies-de-batteries-pour-voitures-electriques.

[165] [online] http ://www.a123systems.com/prismatic-cell-amp20.htm.

[166] T.J. Barlow, S. Latham, I. McCrae, and P. Boulter. *A reference book of driving cycles for use in the measurement of road vehicle emissions*. 2009.

[167] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2) :182–197, 2002.

Résumé & Mots clés

Résumé : La conception d'un système mécatronique est un problème d'optimisation multi-disciplinaire et multi-objectif. Les approches d'optimisation actuellement utilisées, pour l'optimisation des systèmes multidisciplinaires, sont coûteuses en temps de calcul, difficiles à mettre en œuvre et non-flexibles avec la phase de conception préliminaire, où les objectifs et les contraintes de conception changent fréquemment. D'où la nécessité de chercher une nouvelle technique plus simple à mettre en œuvre, moins coûteuse et qui permet d'adapter dynamiquement une solution suite à un changement des spécifications. C'est dans ce contexte que cette thèse se focalise sur le développement d'une approche multi-agents de conception qui, se basant sur les connaissances disciplinaires et par un comportement coopératif, permet de trouver collectivement une solution optimale qui satisfait les contraintes et les performances demandées.

L'approche proposée est basée sur un processus de conception pour faciliter la conception collaborative distribuée des systèmes mécatroniques. Cette approche est appliquée à la conception préliminaire d'un véhicule électrique pour illustrer comment l'utilisation du paradigme multi-agent aide les concepteurs à prendre des décisions efficaces et de parvenir à une décision optimale de l'ensemble du problème. Une étude comparative avec les méthodes classiques d'optimisation est faite afin de démontrer la validité et l'efficacité de l'approche proposée.

Mots clés : Approche multi-agents, système mécatronique, conception collaborative distribuée, véhicule électrique, modélisation, simulation, optimisation.

Abstract & Keywords

Abstract : The design of a mechatronic system is a multi-disciplinary and multi-objective optimization problem. Optimization approaches currently used for the optimization of multidisciplinary systems are expensive in computation time, difficult to implement, and inflexible with the preliminary design phase, in which the objectives and design constraints change frequently. It is therefore necessary to look for new techniques easier to implement and less expensive, that enable to adapt dynamically a solution due to a change in specifications. In this context, the thesis focuses on the development of a multi-agent design approach, based on disciplinary knowledge and cooperative behavior, and makes possible to collectively find an optimal solution that satisfies the required constraints and performance.

The proposed approach is based on a design process to facilitate collaborative distributed design of mechatronic systems. This approach is applied to the preliminary design of an electric vehicle to illustrate how the use of the multi-agent paradigm helps designers in making effective decisions and to achieve an optimal decision of the overall problem. A comparative study with traditional optimization methods is made to demonstrate the validity and effectiveness of the proposed approach.

Keywords : Multi-agent approach, mechatronic systems, distributed collaborative design, electric vehicle, modeling, simulation, optimization.