

UNIVERSITÉ DE MONTRÉAL

DÉVELOPPEMENT D'UN SYSTÈME D'OPTIMISATION
MULTIDISCIPLINAIRE FLEXIBLE ET ADAPTABLE À DIFFÉRENTES
MÉTHODES DE DÉCOMPOSITION

PASCAL PRADO
DÉPARTEMENT DE GÉNIE MÉCANIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE MÉCANIQUE)

JUIN 2005



Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 978-0-494-16840-0

Our file Notre référence

ISBN: 978-0-494-16840-0

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.


Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

DÉVELOPPEMENT D'UN SYSTÈME D'OPTIMISATION
MULTIDISCIPLINAIRE FLEXIBLE ET ADAPTABLE À DIFFÉRENTES
MÉTHODES DE DÉCOMPOSITION

présenté par: PRADO Pascal

en vue de l'obtention du diplôme de: Maîtrise ès science appliquées

a été dûment accepté par le jury d'examen constitué de:

M. REGGIO Marcello, Ph.D., président

M. TRÉPANIÉ Jean-Yves, Ph.D., membre et directeur de recherche

M. TRIBES Christophe, Doctorat, membre et codirecteur de recherche

M. GUIBAULT François, Ph.D., membre

REMERCIEMENTS

Ce mémoire de maîtrise conclut un projet gratifiant et enrichissant, rendu possible par l'implication et le soutien de tous ceux qui, professeurs, collègues ou amis, m'ont permis de mener ce projet à terme.

Merci tout d'abord à Mlle, ou devrais-je dire Docteur (!), Marianne Morand avec qui ces 2 années furent passionnantes et délicieuses!

Merci à Jean-Yves Trépanier, directeur de recherche, qui m'a confié ce projet puis qui m'a guidé efficacement tout au long de ces 2 années de travail.

Merci à Christophe Tribes qui fut un professeur éclairant et motivant avant d'être un conseil avisé, un relecteur diligent (et très exigeant) ainsi qu'un guide indispensable pour mes premières batailles dans l'univers (parfois hostile!) des **.bib*, **.sh* et autres **.cshrc*.

Merci à Mlle Yulia Panchenko, ingénieure chez Pratt & Whitney Canada, qui a toujours été disponible, impliquée et qui a consacré beaucoup de temps à ce projet. Merci de m'avoir fait bénéficier immédiatement de l'expérience du projet IDOT, puis de m'avoir aidé à apprivoiser iSIGHT. Merci, évidemment, d'avoir méticuleusement veillé, ligne par ligne, virgule par virgule, à ce que mon anglais écrit ne heurte aucun anglophone pointilleux!

Merci à Stephen Mah et François Brophy pour leur soutien technique, la description patiente des méthodes de design conceptuel ainsi que leur jugement en tant que futurs utilisateurs de PMDO.

Merci également à Harish Ganoory, Simon Labarre, Jean Maurais et Daniel Tardif,

tous étudiants de Polytechnique qui, par leur coopération à ce projet, ont permis de réaliser un prototype pertinent.

Merci enfin au CRIAQ, au FQRNT et à Pratt & Whitney Canada dont le soutien à cette recherche l'ont rendue possible.

RÉSUMÉ

Ce mémoire présente la conception et la réalisation de l'outil logiciel PMDO (Preliminary Multidisciplinary Design Optimisation). Basé sur une plate-forme d'intégration flexible, l'application de méthodes de décomposition et l'utilisation d'un logiciel d'optimisation, PMDO est destiné à accélérer le design conceptuel des turbomachines de Pratt & Whitney Canada. Les problèmes d'optimisations multidisciplinaires de turbomoteurs sont complexes car ils requièrent la coordination de nombreuses analyses couplées. Leur décomposition automatisée en problèmes équivalents (plus rapides à résoudre) constitue un requis important de la conception de PMDO.

Le prototype réalisé comporte trois composants indépendants et communicants par TCP/IP. Le premier, nommé Java Data Manager, comporte un serveur, un module d'interaction avec des feuilles de calcul Excel ainsi qu'une interface avec un fichier XML de partage de données. Un second composant PMDO_GUI constitue une interface dynamique entre PMDO et ses utilisateurs. Enfin, basé sur le logiciel commercial iSIGHT, le composant d'optimisation coordonne l'exécution de PMDO et dirige le processus d'optimisation jusqu'à un design optimal. En réponse aux attentes des ingénieurs, l'architecture du prototype offre beaucoup de flexibilité quant aux problèmes traités. Ainsi, PMDO est conçu pour comporter une importante étape de configuration automatique visant à adapter le processus d'optimisation à chaque cas d'utilisation.

Le prototype réalisé a été validé à l'aide de cas tests analytiques qui ont démontré le bon déroulement des optimisations pour les décompositions Fully Integrated Optimization (FIO), Distributed Analysis Optimization (DAO) et Distributed Optimizations (DO). La configuration automatisée de PMDO selon l'une de ces mé-

thodes est une contribution originale pouvant être transposée à d'autres problèmes d'optimisation de design multidisciplinaires (MDO).

ABSTRACT

Preliminary Multidisciplinary Design Optimization (PMDO) project was initiated to reduce aircraft engine concept development time. It addresses the development and implementation of the Multidisciplinary Design Optimization methodology in the concept stages of the complex gas turbine design process. These initial phases encompass a wide range of coupled engineering disciplines and make the largest influence on the final product configuration. The PMDO is a software tool intended to integrate existing analysis tools, decompose coupled multidisciplinary problems and, therefore, allow optimizers to enhance preliminary engine designs. It should allow engineers to quickly obtain sound preliminary designs and to explore a wider design space.

Among the requirements for PMDO System is the ability to handle any combination of engine sub-components. Moreover, end users are not familiar with decomposition and optimization issues. Therefore, the PMDO System must be user friendly, flexible and automate as much as possible the decomposition of any design optimization problem. Another key requirement is the necessity to integrate several different 0D/1D analysis tools (executables, macro, spreadsheets) into a platform independent and easily evolving framework.

Based on these requirements, PMDO has a flexible architecture based on three software components, possibly installed on different computers: The Java Data Manager (JDM), a commercial optimizer package and a PMDO Graphical User Interface. The core of the framework is the JDM that handles most of the tasks integration and problem decomposition. Through a dynamic preprocessing, it establishes the list of problem variables, determines their types (local, shared or coupled), duplicates the coupled variables and sets consistency constraints. All problem re-

lated information is stored using the XML standard. This data structure enables easy access to data and efficient archival of results. Finally, information transfer between the JDM and the other components relies on TCP/IP network communication. A commercial optimizer has been chosen as another key component of the PMDO System. Dynamic files (proprietary format) have been developed to handle optimization for both single and bi-level decomposed problems. A Java GUI is provided for initial overall problem description (starting point, included disciplines etc.).

A prototype has been developed based on this architecture. It has been evaluated successfully with test cases and three different decomposition methods (Distributed Optimizations (DO), Distributed Analysis Optimization (DAO) and Fully Integrated Optimization (FIO)).

TABLE DES MATIÈRES

REMERCIEMENTS	iv
RÉSUMÉ	vi
ABSTRACT	viii
TABLE DES MATIÈRES	x
LISTE DES TABLEAUX	xv
LISTE DES FIGURES	xvi
LISTE DES SIGLES ET ABRÉVIATIONS	xx
CHAPITRE 1 INTRODUCTION AU PROJET DE MAÎTRISE	1
1.1 Le contexte industriel	2
1.1.1 Les turbomoteurs modernes	2
1.1.2 Pratt & Whitney Canada : un motoriste majeur	2
1.1.3 La volonté de renouveau des méthodes de développement	3
1.2 Le Consortium de Recherche et d'Innovation en Aéronautique au Québec (CRIAQ)	5
1.3 Les objectifs du travail de maîtrise	7
1.4 L'organisation du document	7
CHAPITRE 2 DÉFINITION DU PROTOTYPE	9
2.1 La méthode de design actuelle	10
2.1.1 Établissement du cahier des charges	10
2.1.2 Choix du concept	11

2.1.3	Phase de validation du concept	13
2.1.4	Phase de design préliminaire	14
2.1.5	Phase de design détaillé	14
2.2	Principaux requis identifiés pour l'outil PMDO	15
2.2.1	Intégration des outils d'analyse éprouvés	15
2.2.2	Flexibilité dans la nature du système et du problème traité .	16
2.2.3	Choix d'une méthode de décomposition	18
2.2.4	Facilité à évoluer	19
2.2.5	Facilité d'utilisation, robustesse et traçabilité des résultats .	19
2.2.6	Requis informatiques spécifiques	21
CHAPITRE 3 INTRODUCTION AUX MÉTHODES D'OPTIMISATION		
	DE DESIGN MULTIDISCIPLINAIRE	22
3.1	Définition des concepts	23
3.1.1	L'optimisation de design multidisciplinaire (MDO)	23
3.1.2	Les analyses disciplinaires	24
3.1.3	L'analyse multidisciplinaire	26
3.2	Formulation et résolution des problèmes d'optimisation	28
3.2.1	L'optimisation mathématique	28
3.2.2	La formulation des problèmes	29
3.2.3	Les algorithmes d'optimisation	30
3.2.3.1	Généralités	30
3.2.3.2	Les optimiseurs à gradient	32
3.2.3.3	Les algorithmes métaheuristiques	33
3.2.3.4	Les systèmes experts	34
3.3	La décomposition de problèmes complexes	35
3.3.1	Enjeux et raison d'être	35
3.3.2	Les deux approches	36

3.4	Principales méthodes de décomposition applicables à PMDO	38
3.4.1	MDO exemple	39
3.4.2	Fully Integrated Optimization	41
3.4.3	Distributed Analysis Optimization	43
3.4.4	Distributed Optimizations	47
3.5	Les défis de la décomposition automatique	51
CHAPITRE 4 CONCEPTION ET FONCTIONNEMENT DE PMDO . .		54
4.1	Le développement d'un prototype de PMDO	55
4.1.1	Les spécificités du prototype	55
4.1.2	La réalisation du projet	55
4.2	L'architecture de PMDO	56
4.2.1	Principe général	56
4.2.1.1	Une structure à trois composants	56
4.2.1.2	Une exécution en trois phases	57
4.2.2	Le composant Java Data Manager	58
4.2.2.1	Programmation	58
4.2.2.2	Le multiserveur	59
4.2.2.3	Le fichier unique de partage de données	59
4.2.2.4	L'exécution des analyses disciplinaires	62
4.2.3	Le composant d'interface graphique PMDO_GUI	65
4.2.3.1	Organisation et fonctionnement de l'interface graphique	65
4.2.3.2	Description du système et du problème	66
4.2.3.3	Présentation et sauvegarde des résultats	69
4.2.4	Le composant d'optimisation	71
4.2.4.1	Le rôle de iSIGHT dans PMDO	71
4.2.4.2	Structure du composant d'optimisation	72

4.2.4.3	Programmation de iSIGHT	73
4.3	Séquence d'exécution détaillée	77
4.3.1	Phase de pré-traitement	79
4.3.1.1	Remplissage de l'interface graphique PMDO_GUI	79
4.3.1.2	Initialisation du Java Data Manager	79
4.3.1.3	Initialisation du composant d'optimisation	81
4.3.1.4	Derniers ajustements, modifications optionnelles et lancement de l'exécution	83
4.3.2	Phase d'exécution	84
4.3.2.1	Séquence détaillée	84
4.3.2.2	Mécanismes de partage de données entre les tâches	87
4.3.3	Phase de post-traitement	89
 CHAPITRE 5 MÉCANISMES DE CONFIGURATION AUTOMATISÉE		
	EN FONCTION DE LA DÉCOMPOSITION	91
5.1	Les défis de la configuration automatique	91
5.2	Choix de conception et principes de programmation	94
5.3	Les trois étapes de la configuration automatisée	95
5.3.1	Constitution automatique du MDA	97
5.3.2	Structuration du composant d'optimisation	98
5.3.3	Définition des objectifs et contraintes	99
5.4	Configuration automatisée selon FIO	101
5.4.1	Éléments du MDO configuré pour FIO	101
5.4.2	Détails de l'algorithme de résolution du couplage	103
5.4.3	Séquence d'exécution	107
5.5	Configuration automatisée selon DAO	110
5.5.1	Éléments du MDO configuré pour DAO	110
5.5.2	Séquence d'exécution	110

5.6	Configuration automatisée selon DO	112
5.6.1	Éléments du MDO configuré pour DO	112
5.6.2	Séquence d'exécution	114
5.7	Ajout d'une méthode de décomposition	115
CHAPITRE 6 VALIDATION DU PROTOTYPE PMDO À L'AIDE DE CAS TESTS		116
6.1	Nature et conditions de réalisation des tests	117
6.2	Conditions de test	117
6.3	Présentation du cas test analytique utilisé	120
6.4	Procédure d'ajout de nouvelles disciplines	121
6.5	Résultats	122
6.5.1	Tests de validation de la flexibilité	122
6.5.2	Tests de validation de l'optimisation	123
6.5.2.1	Résolution avec la décomposition FIO	124
6.5.2.2	Résolution avec la décomposition DAO	129
6.5.2.3	Résolution avec la décomposition DO	131
6.5.3	Analyse	134
6.5.3.1	Choix d'une méthode de décomposition	134
6.5.3.2	Fractionnement du temps de calcul	136
6.5.3.3	Conclusions	139
CONCLUSION		140

LISTE DES TABLEAUX

TABLEAU 6.1	Résultats de convergence FIO en fonction du paramètre ϵ , pour le cas test TP300	126
TABLEAU 6.2	Solutions du cas test TP300 pour différentes méthodes de décomposition	135
TABLEAU 6.3	Récapitulatif des résultats pour le cas test TP300	136
TABLEAU 6.4	Résumé du nombre des évaluations disciplinaires et des mes- sages échangés	138

LISTE DES FIGURES

FIGURE 1.1	Durée virtuellement nécessaire à l'analyse CFD, en régime transitoire, d'une turbine à deux étages (d'après Moustapha (2003))	4
FIGURE 1.2	Réduction du temps de design des étages de turbines (d'après Moustapha (2003))	4
FIGURE 2.1	Progression opposée des processus de design et de fabrication (adapté de Bailey (1999))	11
FIGURE 2.2	Processus de design actuellement utilisé	12
FIGURE 3.1	Les blocs grisés correspondent aux outils d'analyse intégrés dans PMDO (d'après Lytle (1999))	25
FIGURE 3.2	Représentation schématique d'une analyse disciplinaire . . .	26
FIGURE 3.3	Exemple de MDA constitué de deux disciplines couplées . .	27
FIGURE 3.4	Arbre décisionnel pour l'assignation d'un algorithme d'optimisation en fonction du problème (d'après NEOS (2003)) .	32
FIGURE 3.5	Les éléments du problème d'optimisation	40
FIGURE 3.6	Décomposition par la méthode FIO	42
FIGURE 3.7	Décomposition par la méthode FIO : évaluation des contraintes et de la fonction coût	42

FIGURE 3.8	Exemple de diagramme de dépendance pour un MDA à quatre disciplines couplées	44
FIGURE 3.9	Système d'analyses sans itération et séquentiel	44
FIGURE 3.10	Système d'analyses sans itération et parallèle	44
FIGURE 3.11	Décomposition DAO appliquée à un MDA de deux disciplines	47
FIGURE 3.12	Décomposition hiérarchique par une méthode multi-niveaux	48
FIGURE 4.1	Architecture générale du prototype	56
FIGURE 4.2	Structure interne d'un fichier <i>PMDO_Datafile.xml</i>	63
FIGURE 4.3	Exemple d'interface PMDO_GUI lors de la phase de pré-traitement	68
FIGURE 4.4	Exemple d'interface PMDO_GUI lors de la phase de post-traitement	70
FIGURE 4.5	Exemple de l'interface iSIGHT-PMDO pour un MDO à trois disciplines	73
FIGURE 4.6	Diagramme de composantes associé au Coordinateur	76
FIGURE 4.7	Diagramme de composantes associé à la discipline i	77
FIGURE 4.8	Diagramme de séquence simplifié	78
FIGURE 4.9	Flux de données au cours de la phase d'exécution (décomposition DO)	87

FIGURE 4.10	Mécanismes de gestion automatisée du couplage interdisciplinaire par l'emploi de variables globales dupliquées et de contraintes de consistance	89
FIGURE 5.1	MDO originel	92
FIGURE 5.2	Étapes du processus de configuration automatisée	96
FIGURE 5.3	Interfaces iSIGHT pour l'édition de la fonction coût	100
FIGURE 5.4	MDO originel reformulé selon la décomposition FIO	102
FIGURE 5.5	Exemple de MDA de quatre disciplines ne comportant qu'une seule boucle de retour	104
FIGURE 5.6	Exemple de MDA de quatre disciplines comportant deux boucles de retour imbriquées	105
FIGURE 5.7	Séquence d'exécution détaillée pour une itération du <i>Coordinator</i> lors de la résolution d'un MDO de deux disciplines par la méthode FIO	109
FIGURE 5.8	MDO originel reformulé selon la méthode DAO	111
FIGURE 5.9	Séquence d'exécution détaillée pour une itération du <i>Coordinator</i> lors de la résolution d'un MDO de deux disciplines par la méthode DAO	111
FIGURE 5.10	MDO originel reformulé selon la méthode DO	113

FIGURE 5.11	Séquence d'exécution détaillée pour une itération du <i>Coordinator</i> lors de la résolution d'un MDO de deux disciplines par la méthode DO	114
FIGURE 6.1	Décomposition FIO : nombre moyen d'itérations requises pour la résolution du couplage interdisciplinaire	126
FIGURE 6.2	Décomposition FIO : temps de calcul requis (secondes) avant convergence	127
FIGURE 6.3	Décomposition FIO : valeur de la fonction coût à l'optimum, en fonction de ϵ	127
FIGURE 6.4	Progression de la convergence FIO	128
FIGURE 6.5	Améliorations de la convergence FIO	128
FIGURE 6.6	Progression de la convergence DAO	130
FIGURE 6.7	Améliorations de la convergence DAO	130
FIGURE 6.8	Progression de la convergence DO	133
FIGURE 6.9	Améliorations de la convergence DO	133
FIGURE 6.10	Temps d'exécution de PMDO en fonction du nombre de messages TCP/IP échangés	138

LISTE DES SIGLES ET ABRÉVIATIONS

Acronymes utilisés dans le texte

AE	Advanced Engineering
CAO	Conception Assistée par Ordinateur
CFD	Computational Fluid Dynamics
CRIAQ	Consortium de Recherche et d'Innovation en Aéronautique au Québec
DAO	Distributed Analysis Optimization
DO	Distributed Optimizations
DOC	Direct Operating Cost
DOE	Design Of Experiments
DOM4J	Document Object Model for Java
FIO	Fully Integrated Optimization
GUI	Graphical User Interface
HTML	Hyper Text Meta Language
IDOT	Integration Design and Optimization Tool
JDM	Java Data Manager
JSC	Java Spreadsheet Component
MDA	Multi Disciplinary Analysis
MDO	Multidisciplinary Design Optimization
MDOL	Multi Disciplinary Optimization Language
PMDO	Preliminary Multidisciplinary Design Optimization
PR	Pressure Ratio
P&WC	Pratt & Whitney Canada

R&D	Recherche & Développement
SFC	Specific Fuel Consumption
SOAPP	State Of the Art Performance Program
TCL	Tool Command Language
TCP/IP	Transmission Control Protocol/Internet Protocol
XML	eXtensible Markup Language
0D/1D/3D	0 dimension / 1 dimension / 3 dimensions

Caractères usuels

A_i	Fonction mathématique associée à l'analyse disciplinaire i
a_i	Vecteur des variables de sortie associées à l'analyse disciplinaire i
C	Contrainte du problème d'optimisation
$Cout(x)$	Fonction objectif ou coût dépendant des variables du vecteur x
c_{ij}	Vecteur des variables couplées entre les disciplines j (en entrée) et i (en sortie)
l_i	Vecteur des variables locales à la discipline i
p_i	Vecteur des variables paramètres pour la discipline i
$\mathbb{R}$	Ensemble des nombres réels
s	Vecteur des variables partagées (<i>shared</i>)
t	Vecteur des variables auxiliaires associées aux variables couplées
v	Vecteur des variables potentielles pour une optimisation donnée
v^{low}	Vecteur des bornes inférieures de l'espace de définition de v
v^{up}	Vecteur des bornes supérieures de l'espace de définition de v
x_i	Vecteur des variables d'entrée nécessaires à l'analyse disciplinaire i
x^{opt}	Vecteur solution optimale
*	L'exposant * identifie les vecteurs de variables correspondant à une solution du MDA, c'est-à-dire un design réalisable
+	L'exposant + identifie les vecteurs de variables correspondant à un design faisable
ϵ	Tolérance de résolution du couplage interdisciplinaire
Γ	Ensemble de définition d'une fonction mathématique considérée

CHAPITRE 1

INTRODUCTION AU PROJET DE MAÎTRISE

Les avions et leurs sous-systèmes sont certainement parmi les réalisations les plus complexes et abouties de l'industrie moderne. Leur conception mobilise des centaines d'ingénieurs experts effectuant diverses analyses spécialisées. La coordination de leur travail ainsi que l'évaluation de très nombreuses versions intermédiaires sont nécessaires avant d'arriver à un design acceptable satisfaisant les contraintes structurelles, aérodynamiques ou tout simplement économiques.

Dans un contexte de concurrence féroce, il n'est pourtant plus suffisant de faire voler un avion résistant et sûr, il faut aussi qu'il soit meilleur que tous les autres sur le marché! Ainsi, les concepteurs doivent maintenant évaluer le plus grand nombre possible de designs (chacun nécessitant toujours de très nombreuses analyses) afin de se rapprocher d'un optimum. L'optimisation du design des systèmes complexes fait donc l'objet de nombreuses recherches visant à développer de nouvelles façons de concevoir ainsi que des outils logiciels qui tirent parti des ressources (méthodes, ordinateurs, etc.) aujourd'hui disponibles.

Dans ce contexte et s'inscrivant dans le cadre du projet MOSAIC du Consortium de Recherche et d'Innovation en Aéronautique au Québec (CRIAQ), ce travail de maîtrise est voué à l'optimisation du design préliminaire des turbomoteurs. Il fut conduit en réponse au besoin exprimé par le motoriste Pratt & Whitney Canada. Ce chapitre d'introduction est une présentation du contexte dans lequel il fut exécuté, de sa raison d'être, ainsi qu'un énoncé des objectifs de recherche qui s'y rattachent. La dernière section de cette introduction décrit l'organisation du mémoire.

1.1 Le contexte industriel

1.1.1 Les turbomoteurs modernes

Inventé en 1930 par Frank Whittle, le moteur à réaction a marqué le début d'une nouvelle ère pour le transport aérien en supplantant le moteur à pistons. Étroitement intégré à un aéronef dès sa conception, le turbomoteur assure sa propulsion et, dans le cas des hélicoptères, sa sustentation. À ce titre, il en constitue un composant majeur dont les performances déterminent l'enveloppe de vol du véhicule et son coût d'exploitation.

Héritiers des prototypes des années 40, les turbomoteurs modernes se doivent d'être tout à la fois compacts, peu gourmands en carburant ainsi que peu dispendieux à l'achat et à l'entretien. Devant satisfaire simultanément autant de requis peu conciliables, ils représentent de véritables prouesses technologiques, obligeant leurs concepteurs à d'incessantes innovations dans les domaines de l'aérodynamique, du calcul structurel, de la science des matériaux, des automatismes de contrôle et des méthodes de fabrication.

1.1.2 Pratt & Whitney Canada : un motoriste majeur

Pratt & Whitney Canada Cie (P&WC) est un leader mondial parmi les motoristes produisant des turbines à gaz de petite et moyenne puissance (turbosoufflantes, turbopropulseurs, turbomoteurs et générateurs auxiliaires de bord). Bien que souvent cité cette année parce que fournissant le générateur auxiliaire de bord du nouveau gros porteur civil Airbus A380, PW&C développe essentiellement des moteurs destinés aux avions d'affaire, hélicoptères et avions régionaux. Au 31 décembre 2001, soit en presque 40 ans de production, 53 031 moteurs avaient été livrés à 823 com-

pagnies aériennes réparties dans 183 pays. P&WC est une filiale de United Technologies Corporation, société de haute technologie de Hartford (Connecticut). Au Canada, P&WC est le premier investisseur en recherche et développement (R&D) dans l'industrie aérospatiale et le deuxième dans l'ensemble du secteur privé.¹

La compagnie de près de 9000 employés (au 31 décembre 2001) possède des installations et des centres de services dans le monde entier. Les deux plus importants sont à Longueuil, en banlieue sud de Montréal, et à Toronto. Ainsi les activités de R&D sont partagées entre plusieurs départements répartis sur 2 sites ayant chacun des domaines d'expertise particuliers. La conception des compresseurs, soufflantes et chambres de combustion par exemple est réalisée par les 300 ingénieurs du site de Toronto. À Longueuil, par contre, un centre de R&D de 600 employés conçoit les étages de turbines, les arbres de transmission et les sous-ensembles annexes au moteur (boîtes de vitesses, circuits d'huile et d'essence, etc.). La conception de chaque nouveau modèle de moteur requiert une communication très étroite entre les départements et donc entre les différents sites.

1.1.3 La volonté de renouveau des méthodes de développement

Au cours des 2 dernières décennies, l'industrie des petites turbines à gaz a réalisé un grand bond technologique. Cette période se caractérise par un rapide développement des codes d'analyse 3D coïncidant avec le doublement annuel de la capacité des ordinateurs (voir figure 1.1). Ceci, combiné à l'intégration des équipes au sein de groupes de projet moteur, ainsi qu'à la diminution conséquente des périodes de test, a permis une réduction du temps de développement d'un moteur, passé de 5 à 3 ans (la figure 1.2 illustre la réduction du temps de design d'une turbine) (d'après

¹D'après www.pwc.ca.

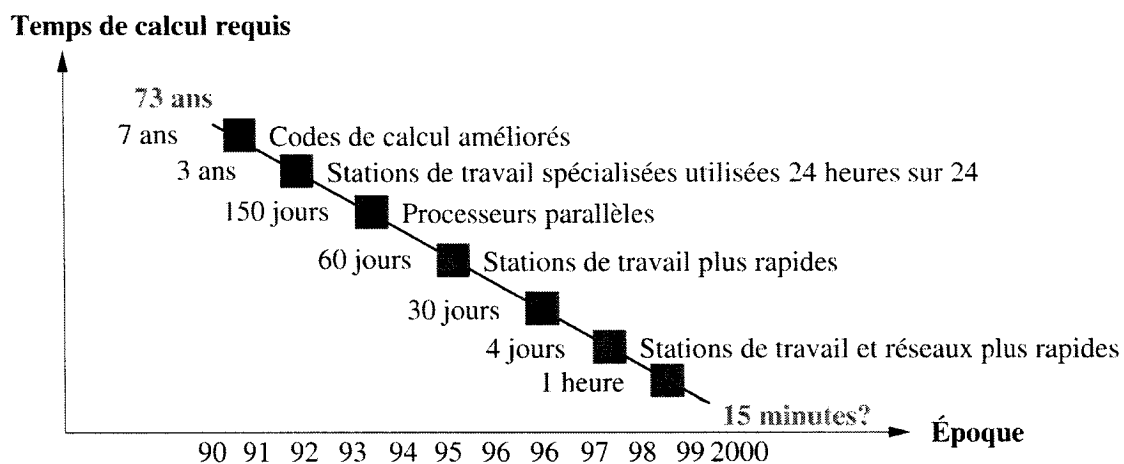


FIGURE 1.1: Durée virtuellement nécessaire à l'analyse CFD, en régime transitoire, d'une turbine à deux étages (d'après Moustapha (2003))

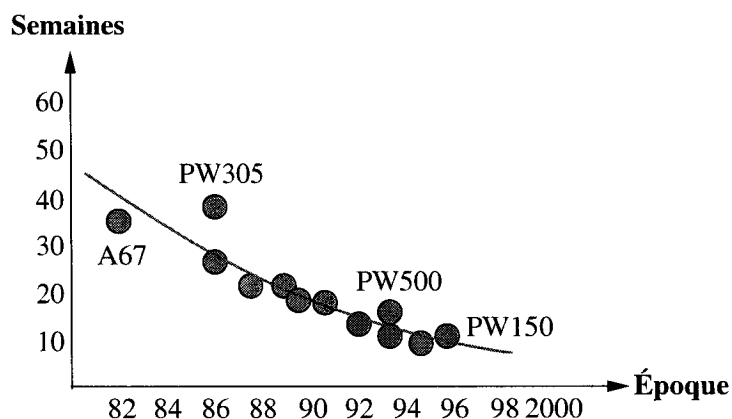


FIGURE 1.2: Réduction du temps de design des étages de turbines (d'après Moustapha (2003))

Moustapha (2003)).

Les réductions des coûts et du poids sont devenues les moteurs des nouveaux projets. De plus, dans les années à venir, les questions d'ordre environnemental telles que le bruit ou l'émission de polluants vont gagner en importance avec l'arrivée de nouveaux règlements très sévères. Ainsi, l'industrie va continuer à évoluer, avec

l'objectif annoncé, pour les nouveaux moteurs, de réduire à 2 ans les délais de conception/développement et de supprimer les activités post-certification.

Pour ce faire, il a été conclu, parmi les équipes de direction de P&WC, qu' « une façon de permettre d'atteindre le moteur optimal, pour des requis donnés, est de développer et d'appliquer des méthodes de MDO dans le processus de conception »² (d'après Moustapha (2003)). Présentée en détails dans le chapitre suivant, l'étape de design conceptuel se prête bien à la MDO, car elle s'accommode d'outils d'analyse simples et rapides. De plus, il est particulièrement indiqué d'effectuer cette optimisation dès la définition du concept car les bénéfices sur le design final sont alors maximisés (Ryan *et al.* (1996) affirment par exemple que « les plus grands efforts déployés dans la phase de conception détaillée ne peuvent compenser totalement le choix d'un mauvais concept »). Ces constatations ont motivé la création, en interne, du projet de recherche intitulé Preliminary MultiDisciplinary Design Optimization (PMDO). Le projet PMDO a pour but la mise en service d'un outil informatique, destiné aux ingénieurs de recherche et développement, permettant d'accélérer le processus de design conceptuel des moteurs, de diminuer le risque d'erreur humaine et aussi, dans une certaine mesure, de faciliter le travail de collaboration entre les départements d'ingénierie.

1.2 Le Consortium de Recherche et d'Innovation en Aéronautique au Québec (CRIAQ)

Le projet PMDO nécessite une part importante de recherche pré-concurrentielle avant la mise en service d'un outil éprouvé. Une partie de sa réalisation est conduite en partenariat avec l'École Polytechnique de Montréal, dans le cadre du Consortium

²Le concept de MDO (Multidisciplinary Design Optimization), soit l'optimisation multidisciplinaire appliquée au design, sera introduit en détails au chapitre 3.

de Recherche et d'Innovation en Aéronautique au Québec (CRIAQ).

Le CRIAQ est un consortium à but non lucratif qui a été formé grâce au soutien financier de Valorisation-Recherche Québec (VRQ), du Fonds Québécois de Recherche sur la Nature et les Technologies (FQRNT), des industries et des universités dans le but de promouvoir et de réaliser des projets de recherche industrielle au stade pré-concurrentiel principalement dans les universités. Le projet PMDO est une mise en application concrète du premier des 5 objectifs du CRIAQ, soit « réaliser des projets de recherche générique ciblés et axés sur l'industrie ».³

Au sein du CRIAQ, le projet MOSAIC de « systèMe d'Optimisation basé sur des Standards et des Analyses permettant une Intégration Configurable » consiste au développement d'environnements de conception et de design multidisciplinaire optimal (MDO). MOSAIC est structuré en un ensemble de modules qui devront inter-agir au travers de protocoles d'échanges définis au cours du projet. Les programmes informatiques développés incluent :

- des outils permettant l'utilisation des protocoles d'échanges,
- des modules d'analyse et d'optimisation,
- une maquette numérique structurée et des extracteurs de données contextuelles.

L'ensemble constituera un environnement modulaire permettant la composition flexible de systèmes MDO pour résoudre des problèmes d'optimisation multidisciplinaire en conception aéronautique. Ce projet de maîtrise recherche est la dixième des douze tâches que comporte le projet MOSAIC.

³D'après www.criaq.aero.

1.3 Les objectifs du travail de maîtrise

Dans la perspective du projet PMDO, l'objectif principal de cette maîtrise est la conception, puis la validation d'un prototype démonstrateur de l'outil logiciel. À partir de ce travail, un logiciel opérationnel pourra ensuite être installé chez Pratt & Whitney Canada, puis mis en service au profit des ingénieurs de R&D. Dans le présent document, le prototype ainsi que le futur logiciel de conception qu'il deviendra seront désignés du nom de PMDO. Les principaux objectifs de ce travail de maîtrise sont les suivants :

1. Établissement des requis par l'étude des méthodes de conception actuelles ainsi que des attentes des futurs utilisateurs de PMDO.
2. Recherches bibliographiques et étude des approches explorées (en interne chez Pratt & Whitney Canada) par le passé.
3. Définition d'un nouveau concept pour la réalisation de PMDO.
4. Conception d'un prototype et programmation des sous-ensembles.
5. Validation du concept à l'aide d'un cas test.

1.4 L'organisation du document

En sus de ce premier chapitre d'introduction au projet, ce document en comporte cinq de développement et un de conclusion.

Le second chapitre précise, dans ses grandes lignes, la définition du prototype telle qu'établie au cours de la première phase de recherche.

Par la suite, le troisième chapitre est une introduction à l'optimisation multidisciplinaire ainsi qu'aux méthodes de décomposition des problèmes complexes. Après

une présentation des concepts et des notations utilisées, l'accent est mis sur la difficulté que représente la configuration automatique de PMDO pour l'une ou l'autre des méthodes de décomposition.

Dans un quatrième chapitre, sont ensuite décrits le prototype réalisé, ses constituants ainsi que sa mise en œuvre pour la résolution d'un problème. Ce chapitre reprend, en particulier, les détails de l'architecture et de la séquence d'exécution correspondant à l'outil PMDO qui a été conçu au cours de ce projet.

Le cinquième chapitre met l'accent sur les mécanismes de configuration automatisée du prototype en fonction de la méthode de décomposition utilisée.

Reposant sur l'utilisation de cas tests et de mesures de performance, le sixième chapitre constitue une validation du bon fonctionnement du prototype.

Enfin, une courte septième partie de bilan reprend les conclusions et perspectives de développements ultérieurs.

CHAPITRE 2

DÉFINITION DU PROTOTYPE

La définition du prototype de l'outil PMDO a constitué la première phase de ce projet de maîtrise.

Près de 3 mois passés dans les bureaux de Pratt & Whitney Canada et de nombreux entretiens avec les ingénieurs de recherche et développement des centres de Longueuil et de Toronto ont permis de comprendre les méthodes de conception actuellement en place et de prendre connaissance des principaux outils d'analyse utilisés. De plus, ces discussions avec les futurs utilisateurs de PMDO permirent de définir leurs besoins et attentes en terme de fonctionnalités, de performance et d'interface.

Cette période fut également mise à profit pour effectuer des recherches bibliographiques et prendre connaissance des résultats d'essais conduits par le passé. Portant sur l'application de méthodes de MDO à l'optimisation des étages de turbines, le travail de maîtrise de M. Jasmin Turcotte¹ a ainsi apporté de nombreuses conclusions précieuses au développement de PMDO (notamment pour le choix de méthodes de décomposition et le partage de données entre plusieurs analyses). De même, les résultats du projet IDOT² et l'expérience partagée par Mme. Yulia Panchenko, ingénieure du département de turbines de Pratt & Whitney Canada, apportèrent de nombreux enseignements relatifs aux outils d'analyse et d'optimisation

¹Le lecteur pourra se référer au mémoire de maîtrise de M. Turcotte (2004) dont les références figurent dans la bibliographie.

²Le projet IDOT et certains de ses résultats firent l'objet d'une publication. Voir Panchenko *et al.* (2002).

utilisés par la suite.

Ces recherches ont fait l'objet d'un premier rapport faisant office d'inventaire des requis identifiés pour l'outil PMDO ainsi que des conclusions des recherches antérieures. Ce chapitre 2 reprend l'essentiel des requis et s'appuie sur une description des méthodes de conception actuellement utilisées.³

2.1 La méthode de design actuelle

La conception d'un moteur est un processus long, complexe, multidisciplinaire (faisant intervenir un grand nombre d'ingénieurs et de départements), itératif et initié au niveau de l'aéronef. Ainsi, la phase de conception de la turbine à gaz commence au niveau du système global (l'avion ou l'hélicoptère) pour descendre ensuite à celui des sous-composants et des pièces mécaniques. Tel qu'illustré sur la figure 2.1, le processus de fabrication, quant à lui, évolue en sens inverse, i.e. depuis chacun des éléments individuels jusqu'à la turbine entièrement assemblée (d'après Panchenko *et al.* (2002)).

La phase de conception s'articule en 5 étapes détaillées ci-après et se succédant tel qu'indiqué sur la figure 2.2.

2.1.1 Établissement du cahier des charges

La première étape du processus est habituellement l'identification par le département de marketing d'un marché potentiel pour un moteur inexistant. Le service commercial du motoriste décrit alors les principales caractéristiques du moteur at-

³Voir le rapport originel intitulé « PMDO: Project Summary »(Prado *et al.* (2003)) pour un inventaire plus exhaustif.

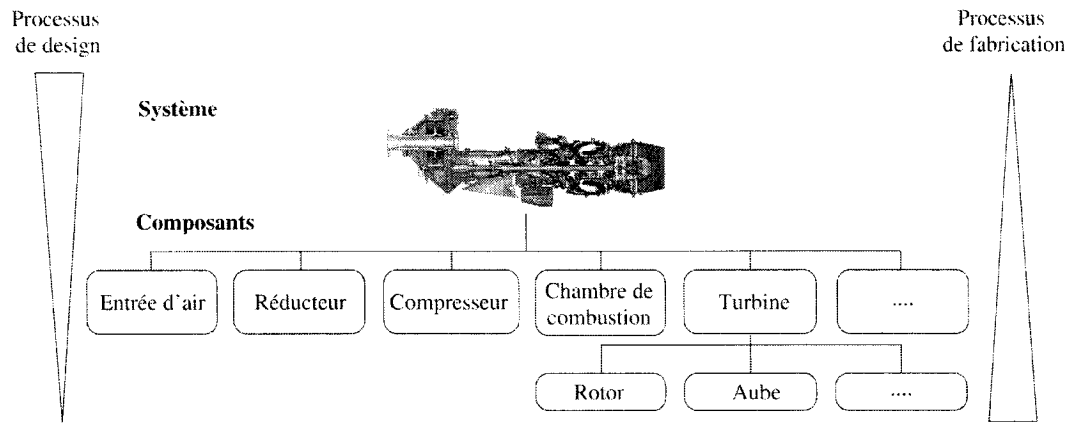


FIGURE 2.1: Progression opposée des processus de design et de fabrication (adapté de Bailey (1999))

tendu sous la forme d'un cahier des charges : type de moteur (turbosoufflante ou turbopropulseur), puissance fournie, consommation spécifique (Specific Fuel Consumption ou SFC), poids et taille, prix de vente, coût d'entretien, durée de vie opérationnelle et enfin profils de missions types.

Le cahier des charges est parfois fourni directement par un avionneur qui le soumet alors généralement à plusieurs fabricants concurrents.

2.1.2 Choix du concept

La phase suivante est dite de design conceptuel; elle vise à établir un concept de moteur pour en estimer, avec une précision raisonnable, la faisabilité, le poids ainsi que les coûts de développement et de production. Cette information est utilisée, le cas échéant, pour formuler une proposition au client avionneur (d'après Moustapha (2003)). C'est à cette étape que sont choisis l'architecture générale du moteur ainsi que son cycle thermodynamique de fonctionnement : rapport de pression dans le compresseur, température de sortie de la chambre de combustion, débits d'air et

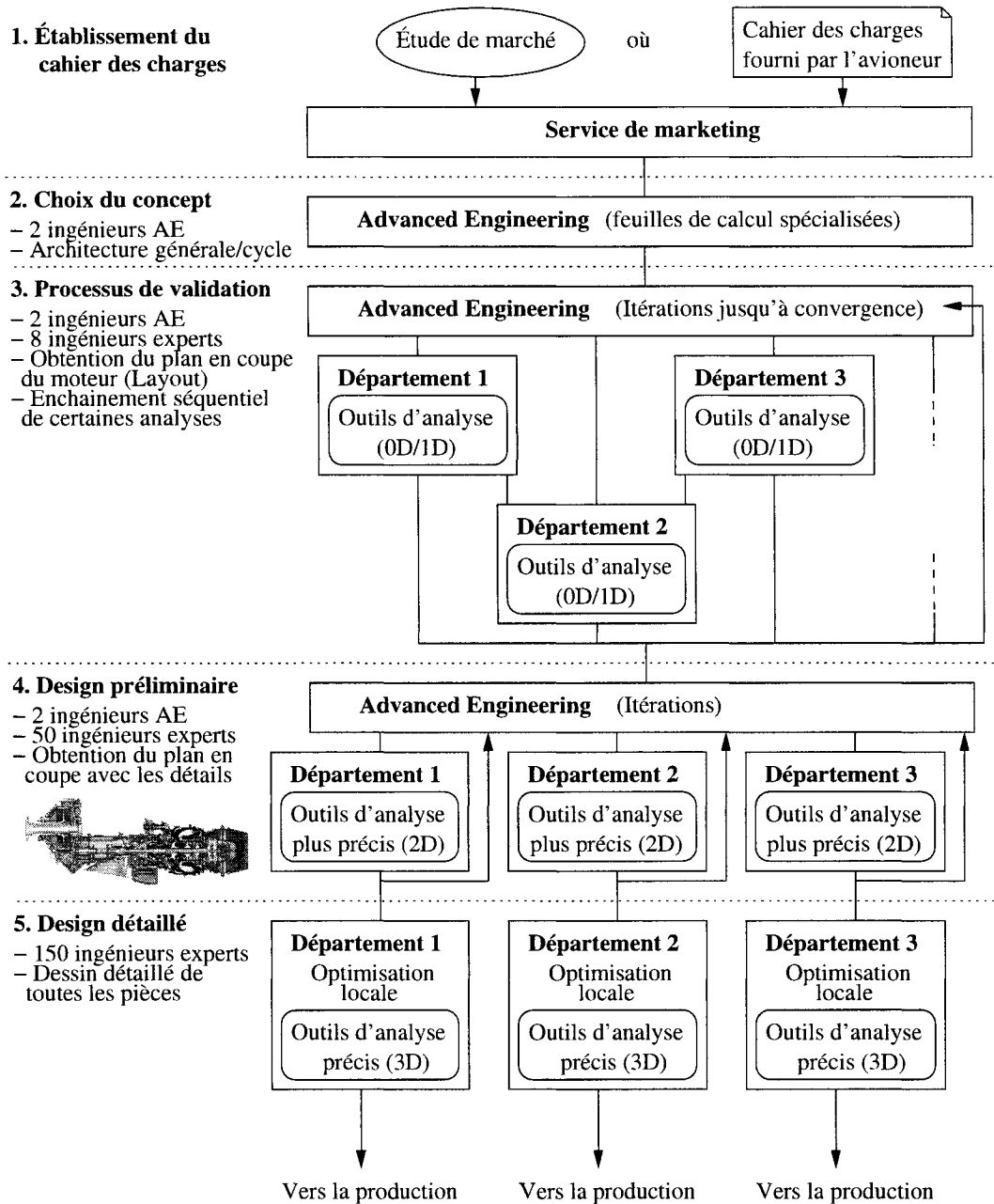


FIGURE 2.2: Processus de design actuellement utilisé

de carburant, nombre d'arbres de rotation, nombre d'étages pour les compresseurs et turbines, etc. Le choix du concept est effectué par des ingénieurs d'expérience (généralement 2 ou 3) travaillant au département Advanced Engineering (AE) et ayant une connaissance transverse de tous les composants des moteurs. Pour ce faire, ils utilisent des codes de calcul 0D très simples : le programme State Of the Art Performance Program (SOAPP) est utilisé pour les calculs thermodynamiques tandis que des feuilles de calcul spécialisées servent à des analyses aérodynamiques et structurelles simplifiées.

2.1.3 Phase de validation du concept

La phase de validation du concept aboutit à la vue en coupe du moteur. Dès lors, la géométrie générale du produit fini est fixée. Cette étape aussi est de la responsabilité du département d'Advanced Engineering. Il s'agit d'un processus itératif au cours duquel différentes versions du design sont testées successivement (itérations dites de haut niveau) en soumettant des demandes aux départements de spécialistes. Ainsi, à la demande du département AE, le design préliminaire de la veine fluide du compresseur est effectué par un ingénieur expert de la division d'analyse aérodynamique des compresseurs. Celui-ci a la latitude d'optimiser la géométrie préliminaire du sous-ensemble dont il a la charge en variant certaines variables de design qui lui sont propres (les variables locales) tout en respectant les contraintes imposées par Advanced Engineering (en cas d'incompatibilités entre disciplines, les variables partagées sont changées pour l'itération de haut niveau suivante). Pour ces optimisations locales, sont utilisés uniquement des codes d'analyses simples (0D ou 1D) s'exécutant très rapidement (durée d'exécution typiquement de l'ordre de la seconde). Chaque itération de haut niveau est donc une succession d'optimisations locales. Certaines de ces optimisations locales sont effectuées en parallèle tandis que

d'autres se suivent séquentiellement au fur et à mesure que les données sont disponibles). À l'heure actuelle, la durée moyenne de cette étape de validation du concept est élevée car les itérations de haut niveau doivent se succéder jusqu'à convergence vers un moteur réalisable (i.e. satisfaisant simultanément les contraintes souvent contradictoires de toutes les disciplines) et ayant les performances requises. En conséquence, « il ne faut pas s'étonner du fait que les solutions actuelles ne soient pas optimales car il n'y a pas suffisamment de temps pour étudier avec attention les interactions entre les variables qui, généralement, sont liées à plusieurs disciplines » (d'après Panchenko *et al.* (2002)). Cette phase mobilise en moyenne 10 ingénieurs (2 provenant du département AE et 8 ingénieurs experts disciplinaires).

2.1.4 Phase de design préliminaire

Cette étape de design préliminaire est très semblable à la précédente, mais mobilise plus de ressources (environ 2 ingénieurs AE et 50 experts disciplinaires utilisant des outils d'analyse plus précis, généralement en 2D). Elle aboutit à la définition d'un plan en coupe détaillé, mais toujours en 2 dimensions.

2.1.5 Phase de design détaillé

Enfin, au cours d'une dernière phase de design détaillé, le design final de chaque pièce du moteur, ainsi que les procédures de fabrication et d'assemblage, sont conçus par des experts (au nombre de 150 environ) et au sein de départements spécialisés. Les codes de calcul utilisés alors sont très complexes, requièrent une géométrie précisément décrite (3D) et demandent des temps de calcul très longs (parfois de l'ordre de la journée). Des analyses en régime transitoire sont aussi effectuées pour valider la conception finale du moteur. À la fin de cette phase de design détaillé, le

turbomoteur est prêt à passer aux étapes de tests, puis de mise en production.

2.2 Principaux requis identifiés pour l'outil PMDO

Le but du projet PMDO est avant tout d'accélérer l'étape de design conceptuel en automatisant les itérations menant d'un design initial pressenti (et souvent infaisable) au concept convergé optimal. Ainsi, le projet PMDO ne vise pas à augmenter la qualité des analyses (car les outils d'analyse utilisés dans PMDO sont ceux qui servent déjà manuellement), mais plutôt à en accélérer l'utilisation et en diminuer le coût (en heures d'ingénieur), permettant ainsi d'investir plus d'effort dans l'exploration de l'espace de design. Ce faisant, l'ambition est aussi de permettre à un algorithme d'optimisation d'aider à la recherche d'un design conceptuel optimal pour les requis imposés par le cahier des charges.

De nombreuses rencontres avec les ingénieurs de R&D (AE ainsi que départements de spécialistes) qui seront les futurs utilisateurs de PMDO ont permis d'identifier les requis indispensables à l'obtention d'un outil répondant aux attentes citées ci-dessus. Aux requis manifestés, se sont ajoutés ensuite ceux qui se sont imposés d'eux-mêmes au fur et à mesure que progressait le travail de définition de PMDO. Ces requis décrivent un outil qui devra entraîner une réorganisation du processus de design actuel.

2.2.1 Intégration des outils d'analyse éprouvés

L'application de techniques de MDO pour la conception préliminaire requiert avant tout l'établissement d'une plate-forme d'intégration qui permette une communication automatique entre les programmes de calcul, et donc un partage efficace des

données. Les codes de design/analyse à intégrer sont ceux actuellement utilisés lors du design conceptuel. À l’heure actuelle, ces outils ne peuvent communiquer sans intervention humaine (facteur ralentissant l’ensemble du processus de design ainsi que potentiellement source d’erreurs). Il s’agit d’outils 0D ou 1D de nature très variée (feuilles de calcul Excel et Applix, exécutables avec fichiers textes d’entrées et de sortie, routines Matlab) mais ayant en commun leur temps d’exécution relativement bref (de l’ordre de la seconde). Intégrer des codes connus des ingénieurs, calibrés, déjà éprouvés et répondant à tous les impératifs de qualité de Pratt & Whitney Canada, au point que leurs résultats individuels soient acceptés sans validations ultérieures et puissent faire l’objet d’un engagement contractuel auprès d’un client, est un requis important pour que la qualité des futurs designs soit garantie.

2.2.2 Flexibilité dans la nature du système et du problème traité

Le logiciel PMDO doit offrir une très grande flexibilité. En effet, le nombre des outils devant être utilisés, et donc inclus au système PMDO pour une utilisation donnée, dépend de la tâche de conception à effectuer. Ainsi, certains outils sont nécessaires pour la conception d’une turbosoufflante (c’est le cas par exemple des codes d’analyse structurelle d’une pale de soufflante), mais pas d’un turbopropulseur. De même, la liste des variables accessibles lors de l’optimisation d’un moteur dépend de sa configuration : nombre de rotors, nombre d’arbres, etc. De plus, une contrainte de flexibilité supplémentaire vient du fait que le système PMDO doit permettre de concevoir un moteur tout en conservant la géométrie de certains sous-ensembles inchangée. Il s’agit alors, par exemple, d’optimiser un turbopropulseur sans en modifier le sous-ensemble compresseur.

Pour un système donné (avec un design initial pour chacun de ses sous-composants),

PMDO doit offrir 3 *Working_mode*. Chacun correspond à un but spécifique :

1. *Working_Mode_A* : Modification automatique des designs de composants jusqu'à l'obtention d'un nombre défini n de designs globaux faisables, c'est-à-dire sans incompatibilité entre les sous-ensembles ni violation des contraintes disciplinaires.
2. *Working_Mode_B* : Modification automatique des designs de composants, dans les limites d'un espace de design fixé, jusqu'à l'obtention du design global optimum (compte tenu des objectifs spécifiés et des contraintes imposées).
3. *Working_Mode_C* : Exécution automatique d'un plan d'expérience, appelé Design Of Experiment (DOE), afin d'explorer l'espace de design à des points pré-définis par le plan. L'ensemble des designs ainsi testés est alors utilisé pour déterminer l'influence de chacune des variables de design sur les grandeurs de sortie ou bien pour construire des surfaces de réponses.

La pertinence de ces 3 *Working_mode* distincts vient du fait que, du point de vue des ingénieurs, le design optimal ne correspond habituellement pas exactement à l'optimum mathématique atteint par un algorithme d'optimisation. Dans la pratique, le design conceptuel optimal est retenu en fonction d'un nombre de critères pondérés trop important pour que ceux-ci soient tous repris dans une fonction coût multi-objectifs (par exemple le moteur de consommation spécifique SFC le plus bas, de poids et de coût d'exploitation DOC minimum, de tolérance maximale aux imprécisions de fabrication, etc.). Les futurs utilisateurs de PMDO souhaitent donc avoir les outils pour juger et améliorer ponctuellement le design fruit d'un processus d'optimisation. L'expertise et l'expérience interviendraient alors toujours dans le processus de conception. Ainsi, le *Working_mode_C* doit permettre d'exécuter des DOE, de réunir un grand nombre de concepts convergés couvrant un ensemble de design très vaste, puis de construire des graphiques de tendances (poids minimum par rapport au nombre de rotors ou au coût, etc.). Le premier *Working_mode_A*,

quant à lui, permet de réaliser plus rapidement des études de faisabilité pour déterminer, par exemple, s'il est possible de réutiliser un composant existant pour concevoir un nouveau moteur.

2.2.3 Choix d'une méthode de décomposition

Tel que détaillé précisément dans le chapitre 3, la conduite de l'optimisation d'un design reposant sur plusieurs analyses couplées (c'est-à-dire qui partagent certaines variables) est une tâche complexe. Dans le but d'accélérer, ou parfois de permettre, la résolution du problème, des méthodes mathématiques de décomposition ont été définies pour permettre de fractionner le processus de résolution en un nombre aussi limité que possible d'étapes élémentaires organisées selon une séquence précise. Le problème décomposé ainsi obtenu doit alors conduire à la même solution que celui d'origine, mais son traitement est facilité par certaines propriétés. Le processus de design préliminaire actuel, reposant entièrement sur une intégration manuelle des disciplines, ne permet cependant pas d'appliquer rigoureusement ces méthodes de décomposition. L'intégration automatique des disciplines, au sein de PMDO, devra offrir cette nouvelle possibilité.

De plus, les recherches bibliographiques portant sur l'optimisation de systèmes complexes ont toujours fait ressortir l'importance du choix d'une méthode adaptée à chaque problème. Un choix malheureux, tel que celui de l'optimisation séquentielle de toutes les disciplines, augmente de beaucoup la durée des calculs et peut même empêcher de converger vers une solution (d'après Alexandrov et Lewis (2004)). C'est donc un requis indispensable et très contraignant que d'offrir à l'utilisateur final le choix entre plusieurs méthodes de décomposition. La structure intrinsèque du logiciel PMDO doit offrir une grande flexibilité car la plate-forme d'intégration doit pouvoir se configurer automatiquement selon l'une ou l'autre des méthodes

choisies.

2.2.4 Facilité à évoluer

PMDO est conçu pour devenir un outil de design devant évoluer en permanence. L'un des requis important quant à sa flexibilité est la possibilité, pour un administrateur du système, d'intégrer de nouvelles disciplines et de nouveaux outils au fur et à mesure qu'ils deviendront disponibles. De plus, certaines évolutions pourraient porter sur l'ajout de méthodes de décomposition ou de fonctionnalités nouvelles dès lors qu'elles deviennent pertinentes : il peut s'agir, par exemple, de la construction des modèles utilisant des surfaces de réponses pour remplacer certaines analyses complexes. Dans ce cas, plusieurs analyses sont effectuées automatiquement en des points répartis sur l'espace de design puis les résultats de ces évaluations sont utilisés pour calibrer directement un modèle mathématique qui, si assez fidèle, peut ensuite remplacer à moindre coût l'analyse originelle.

2.2.5 Facilité d'utilisation, robustesse et traçabilité des résultats

PMDO doit devenir un outil de design permettant d'accélérer le travail des ingénieurs de R&D. Il doit donc être robuste, facile d'utilisation et ses résultats doivent être retraçables et reproductibles.

Dans la plupart des cas d'utilisation de l'outil PMDO, un algorithme d'optimisation, au moins, est nécessaire. Il est reconnu (Vanderplaats (1999)) que le choix d'un algorithme influence fortement l'obtention rapide de résultats valables. S'adressant à des utilisateurs qui ne sont pas, à priori, des spécialistes des optimiseurs, PMDO doit donc comporter certains automatismes pour associer un algorithme à chaque

cas d'utilisation. Ce requis important, entre autres, a conduit P&WC à choisir le logiciel commercial iSIGHT, de Engineous Software, pour traiter toutes les tâches d'optimisation : iSIGHT est donc le moteur des optimisations effectuées à travers PMDO. Ce logiciel inclut un conseiller appelé Advisor qui peut construire un plan d'optimisation (par exemple en choisissant et utilisant de façon successive 2 algorithmes choisis parmi les 16 disponibles dans la version 8 du produit) en fonction du problème traité (nombre et nature des variables, objectifs, éventuelle présence de contraintes, etc.). Dès la conception de l'outil PMDO, il a aussi été choisi d'utiliser les interfaces de iSIGHT (nous les appelons interfaces de « bas niveau ») pour permettre à l'utilisateur d'éditer l'information relative à chaque variable (bornes de l'ensemble de design, éventuel blocage en tant que paramètre, imposition de contraintes ou d'objectif, etc.), éditer des fonctions coût ou de contraintes ou pour construire des graphiques permettant de suivre, en temps réel, la progression de l'optimisation.

En amont de l'interface « bas niveau » de iSIGHT, il est nécessaire de construire une interface permettant à l'utilisateur de décrire le système considéré ainsi que le problème traité. Cette interface conviviale, appelé PMDO_GUI, est utilisée pour recueillir les informations suivantes :

- Tâche à effectuer (Choix de l'un des 3 *Working_mode*),
- conditions de design et objectifs en terme de performance (référence à un fichier issu de l'utilisation du programme SOAPP),
- nature du système considéré (établissement de la liste des disciplines à intégrer ainsi que référence à un fichier d'entrée pour chacune d'entre elles),
- méthode de décomposition choisie.

Afin de garantir la traçabilité et l'archivage des résultats, toute l'information nécessaire à leur obtention (c'est-à-dire essentiellement celle provenant de l'interface PMDO_GUI) doit y être associée pour constituer un fichier unique.

2.2.6 Requis informatiques spécifiques

S'ajoutant aux requis fonctionnels ci-dessus, le cahier des charges de l'outil PMDO comporte aussi certaines contraintes relatives à son développement.

Déjà utilisé dans les départements de recherche et développement (pour des optimisations de sous-ensembles moteurs), le logiciel iSIGHT s'est imposé pour toutes les tâches d'optimisation. Ce choix a de fortes répercussions sur la conception et l'architecture de l'outil PMDO. Par contre, à cause de certaines insuffisances (et par la volonté de ne pas dépendre outre mesure d'un logiciel commercial pour la réalisation de PMDO), les outils de lecture et d'écriture automatique de fichiers textes, ainsi que ceux d'intégration, proposés par Engineous Software seront laissés de côté. Par le passé (années 2001 et 2002), un projet de MDO relatif au design préliminaire d'une turbine (alors baptisé IDOT, soit Integration Design and Optimization Tool) a permis de reconnaître les forces et les faiblesses de la solution reposant entièrement sur iSIGHT. Ainsi, les outils d'intégration de iSIGHT n'offrent pas suffisamment de liberté pour garantir la flexibilité requise. De plus, les mécanismes d'analyse syntaxique de fichiers textes (requis pour l'exécution automatique de programmes utilisant un fichier texte en entrée) se sont avérés lents et complexes à programmer. PMDO devra donc effectuer ces tâches en amont de iSIGHT.

Enfin, les ingénieurs de R&D entamant une longue période de transition de UNIX vers Windows, PMDO doit donc être facilement portable d'un système d'exploitation vers un autre. Puisque le logiciel iSIGHT est potentiellement disponible sur toutes les plate-formes requises, les autres composants de PMDO doivent être développés à l'aide de langages et de bibliothèques offrant les mêmes qualités de portabilité.

CHAPITRE 3

INTRODUCTION AUX MÉTHODES D'OPTIMISATION DE DESIGN MULTIDISCIPLINAIRE

Tel que souligné dans la littérature (voir Kroo (1997)), il existe un grand décalage entre les cas tests souvent utilisés pour mettre à l'épreuve des méthodes d'optimisation et les vrais problèmes rencontrés dans l'industrie. Ceux-ci impliquent non seulement un grand nombre d'analyses mais aussi un nombre très élevé de contraintes et de variables de design, ce qui complexifie grandement le processus de design. L'ampleur des besoins a donc entraîné de nombreuses recherches conduisant au développement de nouvelles méthodes d'optimisation adaptées au design multidisciplinaire en ingénierie.

Le chapitre constitue une introduction aux problèmes représentés par le couplage de plusieurs analyses et par la formulation de problèmes d'optimisation. Il présente également les méthodes de décomposition mises en place pour relever le défi posé par l'optimisation du design de systèmes complexes tels que les turbomoteurs. Enfin, l'assise de ces bases théoriques permettra de souligner l'un des caractères novateurs que devra avoir un outil PMDO apte à répondre aux besoins de ses futurs utilisateurs : la configuration automatisée à plusieurs méthodes de décomposition.

3.1 Définition des concepts

3.1.1 L'optimisation de design multidisciplinaire (MDO)

MDO est l'acronyme anglais de « Multidisciplinary Design Optimization » qui se traduit littéralement par « optimisation de design multidisciplinaire ». Étant donné l'emploi généralisé du terme MDO, même chez les francophones, nous l'utiliserons dans ce document pour décrire un concept que certains ingénieurs (d'après Panchenko *et al.* (2002)) interprètent « comme une méthodologie formelle qui facilite l'exploration et l'exploitation d'interactions interdisciplinaires pour aboutir à un meilleur design global ».

Les premiers développements en MDO remontent aux années 70. Ces recherches ont sans cesse progressé depuis afin de démontrer à l'industrie leur grand potentiel pour augmenter la qualité des designs tout en réduisant significativement la durée des cycles de conception. Ce sont ces perspectives d'amélioration du processus de design conceptuel des turbines à gaz qui ont poussé Pratt & Whitney Canada à s'intéresser aux recherches en MDO.

Du point de vue de l'ingénierie, le recours aux techniques de MDO vise à trouver une solution qui améliore ou optimise un design basé sur plusieurs analyses reliées entre elles, tout en satisfaisant un ensemble de contraintes de design et physiques. L'obtention de cette solution est un processus comportant au moins 2 étapes qu'il est important de distinguer : il s'agit tout d'abord de poser le problème dans un langage mathématique, puis de définir une procédure pour en trouver la solution (d'après Alexandrov et Lewis (1999)). La première étape est celle de la formulation du problème tandis que la seconde sous-entend la mise en application d'un algorithme d'optimisation, éventuellement précédé, de celle d'une méthode de décom-

position. La mise en application des techniques de MDO, dont celles de résolution de couplage, permet généralement de fractionner l'analyse multidisciplinaire complexe en plusieurs analyses disciplinaires pouvant être traitées indépendamment par des spécialistes compétents ou à l'aide d'outils adaptés. Cette possibilité est particulièrement précieuse dans les grandes entreprises pour lesquelles l'expertise est partagée en un grand nombre de départements et d'experts.

3.1.2 Les analyses disciplinaires

Le processus de conception préliminaire des turbines à gaz repose sur l'exécution répétitive de plusieurs analyses spécialisées, dites « analyses disciplinaires », que le projet PMDO vise à intégrer en une plate-forme unique.

Le schéma 3.1 permet de représenter schématiquement l'ensemble des analyses intervenant dans le processus de design des moteurs, depuis l'étape conceptuelle jusqu'à celle de design détaillé. Les outils de calcul correspondant se distinguent par les composants pour lesquels ils sont définis, la discipline concernée ainsi que le degré de fidélité de l'analyse effectuée.¹

1

1. Un composant est un sous-ensemble physique de la turbine à gaz (e.g. le réducteur de vitesse, un rotor ou bien le compresseur dans son intégralité).
2. Une discipline est un type d'analyse effectuée sur un ou plusieurs composants du moteur (e.g. analyse structurelle d'un rotor de turbine ou bien analyse aérodynamique de la veine fluide pour l'ensemble du compresseur). À une discipline sont associés un outil de calcul (formule mathématique, feuille de calcul Excel, exécutable, macro, etc.), des variables d'entrée et de sortie.
3. La fidélité d'une analyse disciplinaire est une mesure de la qualité de ses résultats. La fidélité augmente habituellement avec la complexité (coût et temps d'exécution) de l'analyse ainsi qu'avec le nombre des variables utilisées en entrée ou en sortie. Ainsi une simple corrélation en une dimension (1D) et une analyse par éléments finis, en 3 dimensions et régime transitoire (4D) peuvent servir à effectuer la même analyse avec 2 niveaux de fidélité différents.

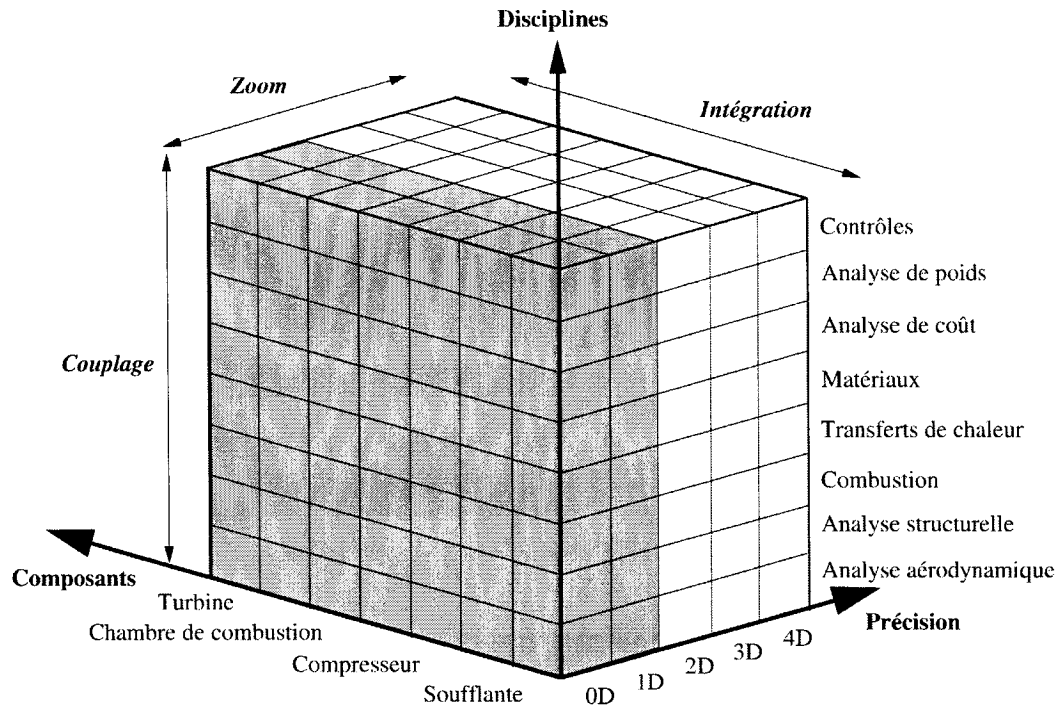


FIGURE 3.1: Les blocs grisés correspondent aux outils d'analyse intégrés dans PMDO (d'après Lytle (1999))

Chacune des « briques » du schéma 3.1 correspond à une analyse unique effectuée sur un composant donné et pour un degré de fidélité attendu. PMDO doit intégrer toutes celles (grisées) de fidélité modérée.

Certaines règles ont été mises en place pour décrire, mathématiquement et schématiquement, les analyses disciplinaires et les vecteurs de variables d'entrée et de sortie. L'analyse de la discipline i est représentée (voir la figure 3.2) par une transformation A_i effectuée sur un vecteur de paramètres d'entrée x_i pour aboutir à des grandeurs de sortie a_i . Ainsi, x_i et a_i sont respectivement les vecteurs des variables d'entrée et de sortie (aussi appelées variables de décision) de l'analyse i associée à une fonction A_i de x_i .

Au cours d'un processus de design, les paramètres pouvant être modifiés sont qua-



FIGURE 3.2: Représentation schématique d'une analyse disciplinaire

liés de variables de design. Dans l'éventualité où plusieurs analyses disciplinaires interviennent, on sépare alors ces variables en 3 catégories distinctes. Étant donné une analyse i :

- Les variables d'entrée locales, regroupées dans le vecteur l_i , ne sont utilisées (et donc modifiables) que par la seule discipline i .
- Les variables d'entrée partagées (ou « shared » en anglais) sont utilisées en entrée par au moins 2 disciplines. Elles sont désignées par le vecteur s .
- Les variables de couplage c_i que nous allons expliciter au §3.1.3 qui suit.

Par ailleurs, l'analyse i requiert aussi un vecteur de paramètres p_i , dont la valeur est fixée pour la discipline i et que nous omettrons dans ce qui suit pour ne pas surcharger la notation, car il ne varie pas au cours de l'optimisation.

Avec les notations vectorielles ci-dessus :

$$x_i = \{s, l_i, c_i\} \quad (3.1)$$

$$a_i = A_i(s, l_i, c_i) \quad (3.2)$$

3.1.3 L'analyse multidisciplinaire

L'ensemble des analyses disciplinaires requises pour la conception préliminaire constitue le MDA (de l'anglais MultiDisciplinary Analysis), c'est-à-dire le système d'analyses disciplinaires reliées entre elles par des relations de couplage.

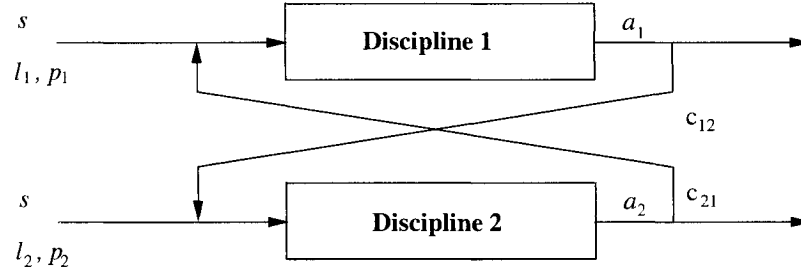


FIGURE 3.3: Exemple de MDA constitué de deux disciplines couplées

La notion de couplage interdisciplinaire reflète la nécessité, pour une solution du MDA, de satisfaire simultanément à toutes les analyses. Le couplage représente le lien existant entre la sortie d'une discipline et l'entrée d'une autre : les variables couplées c_i proviennent des sorties a_j d'autres analyses disciplinaires. Plus précisément, les variables tout à la fois sortie d'une analyse disciplinaire A_i et entrée d'au moins une autre A_j sont qualifiées de couplées et notées c_{ij} (avec $c_{ij} \subset a_i$). La figure 3.3 illustre l'exemple d'un MDA de 2 disciplines couplées pour lequel $c_1 = c_{21}$ et $c_2 = c_{12}$.

Sous forme analytique, l'analyse multidisciplinaire couplée se traduit par un système d'équations. L'exposant * marquant un vecteur d'entrée $x_i = \{s, c_i, l_i\}$ signifie qu'il est solution du système et donc du MDA considéré (dont les sorties sont alors notés $a_i^* \subset a^*$). Cette solution du MDA, lorsqu'elle existe, définit une relation implicite entre x , le vecteur de design, et a_i . Dans notre exemple avec 2 disciplines, on obtient alors 2 fonctions que l'on note $a_1^*(s, l_1, l_2)$ et $a_2^*(s, l_1, l_2)$. Du point de vue de l'analyse multidisciplinaire, un design est dit réalisable s'il correspond à une solution pour laquelle le couplage est résolu. Il est alors noté x^* . Nous discuterons plus loin (voir la section 3.4) des méthodes de résolution du MDA dans le contexte de l'optimisation.

3.2 Formulation et résolution des problèmes d'optimisation

3.2.1 L'optimisation mathématique

Un problème d'optimisation sans contraintes peut s'énoncer de la façon suivante :

Soit une fonction $F : \Gamma \longrightarrow \mathbb{R}$ d'un ensemble $\Gamma \subset \mathbb{R}^m$, trouver un élément $x^{opt} \in \Gamma$ tel que $F(x^{opt}) \geq F(x)$ pour tout $x \in \Gamma$ (maximisation) ou tel que $F(x^{opt}) \leq F(x)$ pour tous les $x \in \Gamma$ (minimisation).

Une solution possible qui maximise (ou minimise, si c'est le but) la fonction objectif est appelée un optimum.

En général, dans un problème d'ingénierie, il y aura plusieurs maximums ou minimums locaux. Un minimum local x^{local} est défini comme un point tel que pour un $\delta \geq 0$ donné et tous les x tels que $\|x - x^{local}\| \leq \delta$, la formule $F(x) \geq F(x^{local})$ reste vraie; c'est-à-dire que sur une boule autour de x^{local} , toutes les valeurs des fonctions sont plus grandes que la valeur en ce point. Les maximums locaux sont définis semblablement. En général, il est possible de trouver les optimums locaux, cependant des connaissances additionnelles sur le problème (par exemple l'éventuelle convexité de la fonction) sont nécessaires pour vérifier que la solution trouvée est un optimum global.

Il n'existe pas, à l'heure actuelle, de méthode connue assurant, quel que soit le type de fonction, que l'on trouvera un extremum global.²

²d'après une définition de l'optimisation mathématique sur <http://fr.wikipedia.org> .

3.2.2 La formulation des problèmes

Formuler un problème d'optimisation au moyen d'un énoncé mathématique succinct équivalent nécessite l'établissement des éléments suivants :

1. Une fonction d'analyse : Le processus d'analyse multidisciplinaire d'un design donné est assimilable à l'évaluation d'une fonction $f_{analyses}$. Dans le cas de PMDO, le MDA est représenté par la fonction $f_{MDA} : \Gamma \longrightarrow \mathbb{R}$ d'un ensemble $\Gamma \subset \mathbb{R}^m$ vers celui des nombres réels. On note $a = f_{MDA}(x)$.
2. Des variables potentielles : Parmi les variables d'entrée $x = (x_1, \dots, x_n)$ de la fonction f_{MDA} , il faut distinguer les variables potentielles v de celles qui sont fixées p . Contrairement à ces dernières, la valeur des variables potentielles peut-être modifiée au cours de l'optimisation afin d'améliorer un design.
3. Un ensemble de design : L'ensemble Γ des variables de design x est défini par l'imposition de contraintes $C(a, x)$ ainsi que par des bornes (supérieures et inférieures) limitant le domaine de variation des variables potentielles v .
4. Une fonction coût : La fonction coût $Cout(a, x)$ est exprimée en fonction de $a = f_{MDA}(x)$ et, occasionnellement, du vecteur x . Sa valeur permet de comparer numériquement deux designs x . L'objectif d'un processus d'optimisation est de minimiser, ou bien de maximiser, la valeur de $Cout(a, x)$.

Sous forme condensée, l'écriture mathématique du problème d'optimisation est :

$$\text{minimise } Cout(a, x) \tag{3.3}$$

$$x = \{v, p\}, \quad a = f_{MDA}(x)$$

$$v \in \Gamma = \{v \in \mathbb{R}^m \mid v_i^{low} \leq v_i \leq v_i^{up}, \quad C(a, x) \geq 0, \quad i \in [1 : N]\}$$

Une solution x est dite faisable si le système équivalent à l'ensemble des analyses est résolu et que toutes les contraintes du problème d'optimisation sont vérifiées. Pour le distinguer d'un design uniquement réalisable x^* (c'est-à-dire satisfaisant le couplage entre disciplines, mais pas les contraintes du problème d'optimisation), le design faisable est noté x^+ . La solution du problème d'optimisation multidisciplinaire (MDO) est le design faisable pour lequel la valeur de $Cost$ est la meilleure compte tenu de l'objectif. Il est qualifié d'optimum et est noté x^{opt} . Il n'existe habituellement qu'un seul optimum global, mais plusieurs optimums locaux peuvent coexister.

3.2.3 Les algorithmes d'optimisation

3.2.3.1 Généralités

Une fois le problème correctement formulé, il reste à choisir un algorithme d'optimisation adapté à sa résolution. Partant d'un design initial (parfois plusieurs pour certains algorithmes génétiques multi-îles par exemple) ne satisfaisant pas nécessairement les contraintes imposées, un tel algorithme répète l'évaluation de la fonction coût, ainsi que des contraintes, jusqu'à l'éventuelle convergence vers un optimum.

Il existe un grand nombre d'algorithmes d'optimisation différents car leur efficacité varie significativement en fonction du problème formulé. Celle-ci n'est pas aisée à quantifier, mais 4 critères au moins peuvent être retenus :

1. La capacité à trouver l'optimum (lorsqu'il existe), et ce quel que soit le point de départ de l'optimisation.
2. Le taux de convergence. Il quantifie la rapidité d'obtention de l'optimum, soit le nombre d'évaluations de f_{MDA} et de la fonction coût $Cost$ qui sont nécessaires à la convergence.

3. L'aptitude à explorer l'ensemble de design. Dans le cas où des optima locaux existent, certains algorithmes parviennent à ne pas rester bloqués au voisinage d'un extremum local, mais plutôt à le dépasser afin de poursuivre la recherche.
4. La capacité des algorithmes à gérer les contraintes.

Aucune méthode connue en 2004 (à part bien entendu l'énumération exhaustive ou l'analyse algébrique) ne permet de trouver avec certitude l'extremum global d'une fonction non convexe en un nombre fini d'itérations. Les extrema déterminables sont toujours locaux à un domaine, et demandent souvent, même dans ce cas, que la fonction ait quelques caractéristiques particulières, par exemple qu'elle soit continue ou continûment dérivable.

L'utilisation d'un arbre décisionnel (voir la figure 3.4) pour caractériser le problème est une bonne méthode permettant d'appuyer le choix d'un algorithme d'optimisation (d'après NEOS (2003)). Un tel arbre permet de choisir les familles de candidats probables en fonction de critères tels que le caractère discret ou non des variables ou bien l'éventuelle présence de contraintes. Ainsi, les problèmes d'optimisation liés à PMDO relèvent de la programmation non linéaire avec contraintes, la plupart du temps pour des variables continues. Cependant, certaines grandeurs (telles que le nombre de pales sur un rotor) intervenant dans la conception des moteurs sont des valeurs discrètes : leur inclusion dans la liste des variables accessibles à l'optimiseur peut empêcher l'utilisation de certains algorithmes.

Trois classes, présentées ci-dessous, peuvent être identifiées parmi les nombreux algorithmes d'optimisation susceptibles d'être utilisés pour PMDO.

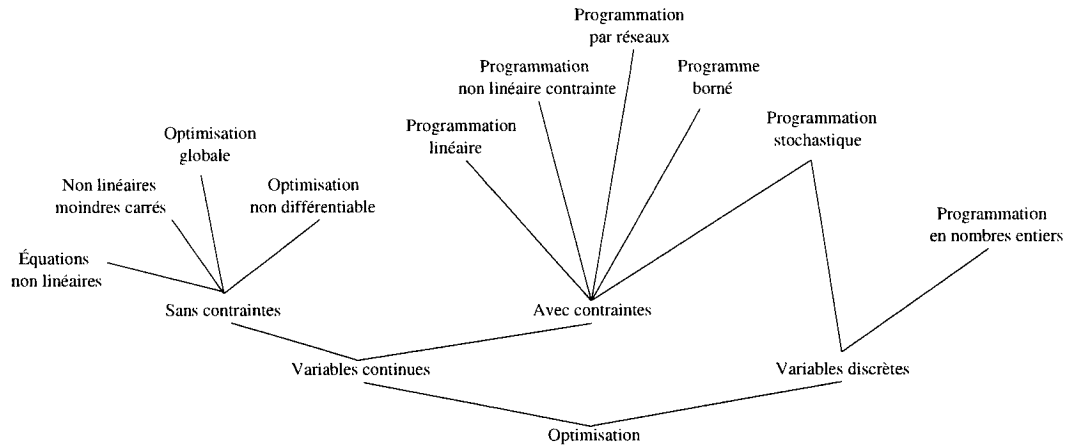


FIGURE 3.4: Arbre décisionnel pour l'assignation d'un algorithme d'optimisation en fonction du problème (d'après NEOS (2003))

3.2.3.2 Les optimiseurs à gradient

Les optimiseurs à gradients utilisent l'information relative au vecteur ∇Cost des dérivées de la fonction coût Cost par rapport à chacune des variables pour déterminer, pour l'itération en cours, la direction de recherche suivant l'objectif du problème (minimiser ou maximiser). L'application de ces méthodes requiert que la fonction coût soit continûment dérivable. Il en existe de nombreuses variantes car elles sont très efficaces pour beaucoup de problèmes d'optimisation. En matière d'exemples d'algorithmes à gradient, on pourra retenir Sequential Quadratic Programming (SQP), Steepest Descent et Sequential Linear Programming (SLP).

Les algorithmes à gradient sont particulièrement indiqués dans le cas de fonction coût dont les n dérivées peuvent s'exprimer à l'aide de fonctions analytiques simples : dans ce cas, la détermination de la dérivée ne demande que n simples évaluations de fonctions. Dans l'éventualité fréquente où la fonction coût nécessite l'utilisation d'une « boîte noire » dont le contenu ne peut être remplacé par une unique expression analytique, les valeurs de dérivée doivent être estimées : la

méthode des différences finies est généralement choisie pour cela. L'approximation des dérivées est alors un calcul dispendieux (l'évaluation des n dérivées partielles requiert tout autant d'appels à la « boîte noire » et donc de résolution du MDA) et la précision est tributaire du choix du pas de différentiation. Plus récemment, diverses approches ont été proposées pour éliminer l'utilisation des différences finies. On peut citer la méthode de l'équation des sensibilités, les algorithmes de différenciation automatique et la méthode des nombres complexes. Bien qu'efficaces, ces méthodes requièrent toutes que le module d'analyse initial soit modifié selon un algorithme spécifique pour fournir les gradients en plus des solutions. Ceci limite cependant leur utilisation dans le contexte industriel.

3.2.3.3 Les algorithmes métaheuristiques

La seconde classe est celle des algorithmes métaheuristiques (également dits de recherche directe) qui tentent d'obtenir une valeur approchée de l'optimum global dans le cas de problèmes d'optimisation difficiles et pour lesquels on ne connaît pas de méthode classique plus efficace. Ces méthodes sont généralement des algorithmes stochastiques, qui progressent vers un optimum par échantillonnage d'une fonction coût. Certaines métaheuristiques sont théoriquement convergentes (i.e capable de trouver l'optimum global si le temps de calcul tend vers l'infini) sous certaines conditions. Cependant, elles sont rarement vérifiées en pratique et ces méthodes ne donnent alors aucune garantie sur l'optimalité du résultat.³

Le fonctionnement des algorithmes métaheuristiques s'articule autour de trois notions : l'exploration, l'exploitation et l'apprentissage. L'exploration désigne le processus visant à récolter de l'information sur le problème optimisé. L'exploitation

³d'après une définition des algorithmes métaheuristiques sur <http://fr.wikipedia.org> .

est l'étape d'utilisation de l'information déjà recueillie pour définir et parcourir les zones intéressantes de l'espace de recherche. L'apprentissage permet à l'algorithme de ne tenir compte que des zones où l'optimum global est susceptible de se trouver, évitant ainsi les optimums locaux. Les métaheuristiques progressent de façon itérative, en alternant des phases d'exploration, d'exploitation et d'apprentissage. L'état de départ est souvent choisi aléatoirement, l'algorithme se déroulant ensuite jusqu'à ce qu'un critère d'arrêt soit atteint.

Les métaheuristiques sont souvent inspirées par des systèmes naturels, qu'ils soient pris en physique (cas du recuit simulé), en biologie de l'évolution (cas des algorithmes génétiques) ou encore en éthologie (cas des algorithmes de colonies de fourmis ou de l'optimisation par essaims particulières).

3.2.3.4 Les systèmes experts

Les systèmes experts ne sont pas, à proprement parler, des algorithmes d'optimisation. Ils peuvent toutefois se révéler très efficaces, dans le cas de l'outil PMDO, pour améliorer localement une solution existante.

Le système expert est constitué par l'alliance d'un ordinateur et d'un logiciel conçu pour simuler le savoir-faire d'un spécialiste dans un domaine précis. L'organisation d'un système expert est caractérisée par une base de connaissances représentant l'expertise proprement dite (cette base ne peut être constituée qu'avec l'aide d'un spécialiste du domaine traité), par les algorithmes gérant cette connaissance (constituant le moteur d'inférences, ou interpréteur, ils sont généraux, c'est-à-dire indépendants du domaine), par une base de faits (ensemble de faits décrivant le problème précis que l'on propose au système de traiter) et, enfin, par deux interfaces indispensables à la bonne communication homme-machine (l'une facilite

le dialogue avec l'utilisateur au cours d'une session; l'autre permet à l'expert du domaine de consulter ou d'enrichir la base de connaissances du système). La base de connaissances est la clé d'un système expert. Celui-ci ne produira des résultats intéressants que dans la mesure où cette base contient, pour un domaine donné, des connaissances aussi précises, aussi justes, aussi complètes et détaillées que celles d'un expert dans le même domaine (d'après Cordier et Rousset (2005)).

Le logiciel iSIGHT, moteur des optimisations conduites par PMDO, contient un outil permettant de construire un système expert applicable à des problèmes d'ingénierie. Il offre également (dans sa version 8) quinze autres algorithmes appartenant aux 2 premières catégories.

3.3 La décomposition de problèmes complexes

3.3.1 Enjeux et raison d'être

Tel qu'énoncé dans le second chapitre, le processus de design conceptuel d'une turbine à gaz requiert l'exécution d'un grand nombre d'analyses couplées. Celles-ci se font à l'aide d'une douzaine de codes de calcul indépendants. Dans le but d'obtenir un bon design général pour un système complexe, l'optimisation successive et indépendante de chaque discipline est une méthode qui fut utilisée, parce que facilement mise en place, mais qui ne saurait conduire à un système optimum. En effet, les designs disciplinaires produits sont fréquemment en désaccord les uns par rapport aux autres vis-à-vis de leurs contraintes et objectifs respectifs. Le couplage entre les disciplines oblige alors à recourir à des ajustements post-optimisation pour modifier les optimums disciplinaires et constituer un design faisable non optimal. L'obtention d'un optimum requiert l'intégration des analyses. Ainsi, une plate-forme de design

automatisé permet de constituer un MDA complexe, comportant un grand nombre de variables de design (pouvant être de l'ordre de 500 pour un moteur complet) et plusieurs couplages interdisciplinaires. Pourtant, la traduction d'un problème d'optimisation basé sur ce MDA (appelé problème ingénieur) en sa formulation mathématique directe laisse présager des difficultés de calcul. Ce problème est très lourd et déraisonnablement long à résoudre à l'aide d'un seul optimiseur.

Cette réalité explique les recherches visant à transformer de tels problèmes en leur associant d'autres formulations mathématiquement équivalentes mais plus facilement mises en œuvre. Les méthodes de décomposition sont donc des transformations appliquées au MDA et à la formulation directe du problème ingénieur afin de fractionner le processus de résolution en plusieurs étapes simples conduisant au même résultat.

Certaines méthodes de décomposition transforment un problème d'optimisation en un équivalent nécessitant la résolution en parallèle de plusieurs sous-problèmes confiés à des optimiseurs indépendants.

3.3.2 Les deux approches

Parmi les nombreuses méthodes de décomposition existantes, il existe deux approches motivées par des besoins différents : la perspective structurelle, qui respecte l'organisation en disciplines indépendantes, ainsi que la perspective algorithmique qui elle prend en compte les caractéristiques des méthodes d'optimisation (d'après Alexandrov et Lewis (2000)).

La perspective structurelle est la plus intuitive. Elle conduit à des décompositions visant à respecter l'autonomie de chaque discipline du MDA et, dans le cas des méthodes multi-niveaux, à coordonner les solutions de plusieurs sous-problèmes

de façon à obtenir l'optimum recherché. Ces méthodes multi-niveaux comportent alors plusieurs optimiseurs spécialisés, ou locaux, et un optimiseur système pour les coordonner.

Le fractionnement du problème entre plusieurs optimiseurs permet de diviser les variables de design en plusieurs petits groupes et ainsi d'alléger la tâche de l'optimiseur système/maître. Cela permet aussi de distribuer les optimisations disciplinaires en fonction des départements et donc des compétences et des ressources. L'exécution simultanée des optimisations locales permet aussi un gain de temps par rapport aux enchaînements séquentiels. De plus, cela laisse la possibilité d'utiliser un optimiseur adapté à chaque analyse et de profiter de l'expertise des spécialistes de la discipline considérée pour finalement converger plus rapidement.

Les méthodes de décomposition basées sur la perspective structurelle tendent à reproduire naturellement le processus de design dans les grandes entreprises qui passe généralement par (au moins) 2 niveaux d'optimisation : l'optimisation globale (portant sur quelques variables systèmes (ou globales) remarquables et les optimisations locales (susceptibles de modifier les variables locales à une discipline uniquement). Ce mode de fonctionnement s'apparente à celui actuellement utilisé dans le service de R&D de Pratt & Whitney Canada : le département AE coordonne des tâches d'optimisations disciplinaires attribuées à des départements spécialisés afin de converger vers le meilleur design conceptuel.

Toutefois, cette perspective structurelle naturelle peut conduire à des formulations non-linéaires multi-niveaux qui sont réputées difficiles à résoudre (d'après Migdalas *et al.* (1998)).

Au contraire, la perspective algorithmique est basée sur la connaissance des forces et faiblesses des algorithmes d'optimisation. Elle conduit à formuler le problème

de MDO de façon à ce que l'optimisation correspondante puisse être effectuée de façon fiable et efficace (d'après Alexandrov et Lewis (2000)) par les optimiseurs à disposition des ingénieurs.

Pour le vecteur x^{opt} , le problème décomposé se doit d'être mathématiquement équivalent à celui d'origine. Cette équivalence n'est cependant pas toujours assurée sur l'ensemble de design Γ dans son intégralité. De plus, certaines méthodes de décomposition aboutissent à des problèmes que les algorithmes d'optimisation disponibles ne peuvent résoudre. Ceci explique qu'il soit si important de garder la possibilité de changer de méthode de décomposition selon le besoin.

3.4 Principales méthodes de décomposition applicables à PMDO

La définition de nouvelles méthodes de décomposition est un sujet de recherche très fertile. Les avancées dans ce domaine accompagnent celles des techniques de conception assistée par ordinateurs et la définition de nouveaux algorithmes propres à traiter des problèmes de plus en plus complexes, impliquant un nombre toujours croissant de variables de design. Les recherches portent tant sur la définition de nouvelles méthodes que sur la mise en place de moyens pour les décrire ou comparer leurs performances et adéquation à des problèmes typiques. Pour exemple, il suffit de rappeler que, parmi les projets actuellement financées par le CRIAQ (voir §1.2), l'une des tâches de MOSAIC est explicitement un projet de recherche sur les méthodes de décomposition.

L'objet de cette section du document n'est aucunement de faire un inventaire exhaustif des différentes méthodes disponibles aujourd'hui, mais plutôt d'en présenter les grandes familles en prenant pour exemple les trois mises en place dans le prototype de PMDO. Les deux premières méthodes de décomposition présentées

ci-après appartiennent à la classe dite « à 1 niveau » pour laquelle le contrôle de l'optimisation est géré par un unique optimiseur. Pour la méthode Fully Integrated Optimization (FIO), les tâches de résolution du couplage interdisciplinaire et d'optimisation sont dissociées tandis que, dans le cas de Distributed Analysis Optimization (DAO), l'emploi de contraintes de compatibilité ajoutées à l'optimiseur permet de lui déléguer la résolution du couplage. La méthode Distributed Optimizations (DO) constitue quant à elle un exemple de décomposition « multi-niveaux » nécessitant l'utilisation d'un optimiseur maître pour coordonner plusieurs optimiseurs locaux.

3.4.1 MDO exemple

Les descriptions qui suivent s'appuient sur l'exemple de l'optimisation aérostructurale d'une aile d'avion simplifiée (d'après Tribes *et al.* (2004)). Le MDA associé (présenté à la figure 3.3) comporte 2 disciplines : une analyse aérodynamique indexée 1 et une analyse structurelle indexée 2. Les analyses 1 et 2 (i et j) sont couplées car chacune utilise, parmi ses paramètres d'entrée x_i , des variables produites par la seconde. Ces variables couplées issues de la discipline i et utilisées en entrée par la discipline j sont regroupées dans le vecteur c_{ij} . Dans notre exemple, le vecteur c regroupe $c_{12} \subset a_1$ et $c_{21} \subset a_2$, soit l'ensemble des variables couplées du MDA. Les variables non couplées sont dites partagées (notées s) si elles servent en entrée pour les 2 disciplines, locales l_i dans le cas contraire.

L'optimisation conduite vise à déterminer les vecteurs de variables de design pour lesquelles la fonction coût C_{out} est maximisée pour un design satisfaisant toutes les contraintes (de couplage, disciplinaires et interdisciplinaires). C_{out} est une mesure du rayon d'action de l'avion dont les ailes sont décrites par une valeur de s , l_1 et l_2 . Les contraintes de couplage sont celles représentées par la présence de va-

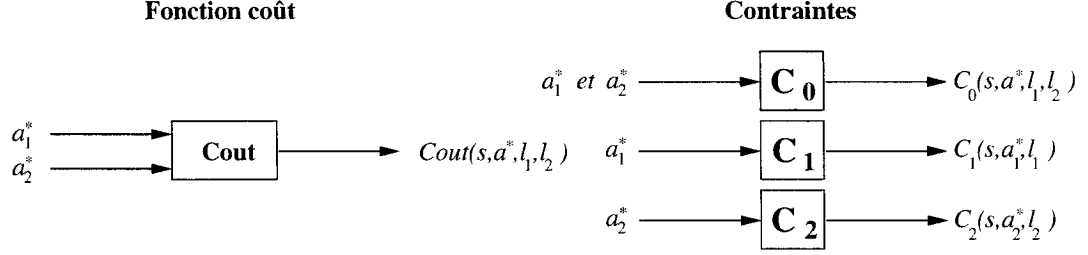


FIGURE 3.5: Les éléments du problème d'optimisation

riables couplées et traduisent l'interaction des disciplines entre elles. Les vecteurs de variables d'entrée x et de sortie a_1 et a_2 sont notés $x^* = (s, c, l_1, l_2)$, a_1^* et a_2^* lorsqu'ils correspondent à un design réalisable, c'est-à-dire pour lequel toutes les contraintes de couplage sont satisfaites. Les contraintes globales C_0 sont calculées en fonction des résultats a_1 et a_2 des analyses 1 et 2 simultanément. La satisfaction des contraintes disciplinaires C_i , par contre, dépend des seuls résultats de l'analyse i (e.g. le stress maximal calculé dans l'aile, par la discipline 2, doit rester inférieure à une valeur limite). La figure 3.5 illustre les fonction coût et contraintes participant à la formulation de ce problème d'optimisation.

$$\underset{s, l_1, l_2}{\text{maximise}} \text{Cout}(s, a^*, l_1, l_2) \quad (3.4)$$

$$\text{tel que : } C_0(s, a^*, l_1, l_2) \geq 0,$$

$$C_1(s, a^*, l_1) \geq 0,$$

$$C_2(s, a^*, l_2) \geq 0$$

Il est important de rappeler que seules certaines combinaisons des variables d'entrée c , s , l_1 et l_2 , identifiées par un exposant $^+$, conduisent à des design faisables, c'est-à-dire réalisables et satisfaisant simultanément toutes les contraintes C_0 , C_1 et C_2

du problème d'optimisation. Les autres aboutissent à des designs infaisables pour lesquels une contrainte au moins est violée. Pour le design optimal x^{opt} , être faisable est une condition nécessaire mais non suffisante.

3.4.2 Fully Integrated Optimization

S'agissant d'optimiser un design en fonction d'une fonction coût basée sur l'évaluation d'un MDA, la méthode la plus intuitive consiste à séparer les tâches d'intégration des disciplines et d'optimisation. Du point de vue de l'optimiseur, l'ensemble des analyses intégrées peut alors être perçu comme une unique « boîte noire » (c.f. la figure 3.6 pour illustration) dont la structure interne est cachée. Cette méthode est dite Fully Integrated Optimization (FIO), c'est-à-dire « Optimisation Entièrement Intégrée ». Dans ce cas, une méthode de résolution du couplage interdisciplinaires (méthode de Gauss-Seidel (d'après Perez *et al.* (2004)) ou du point fixe) doit être mise en place indépendamment de l'algorithme d'optimisation. Pour chaque vecteur des variables potentielles $v = (s, l_1, l_2)$, son rôle est de déterminer la valeur de c^* (vecteur des variables couplées) pour laquelle les couplages interdisciplinaire sont résolus (design réalisable). En cas de succès, les calculs des fonctions C_{out} ainsi que des contraintes C_i peuvent être effectués (c.f. les figures 3.5 et 3.7). Au contraire, aux designs irréalisables correspondent les vecteurs $v = (s, l_1, l_2)$ pour lesquels le couplage ne peut être résolu. Comme déjà spécifié, un design peut-être réalisable mais non faisable si, contrairement aux contraintes de couplage, les contraintes disciplinaires ne peuvent être vérifiées.

Un des avantages de la méthode FIO est qu'un design réalisable est obtenu à chaque itération de l'optimiseur (sous réserve que la valeur de c résolvant le couplage existe et puisse être trouvée). Cependant, étant donné le coût de calcul de chaque analyse, le nombre d'itérations effectuées pour la détermination de c (par des méthodes

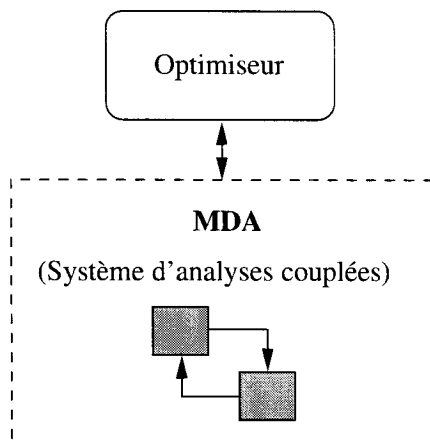


FIGURE 3.6: Décomposition par la méthode FIO

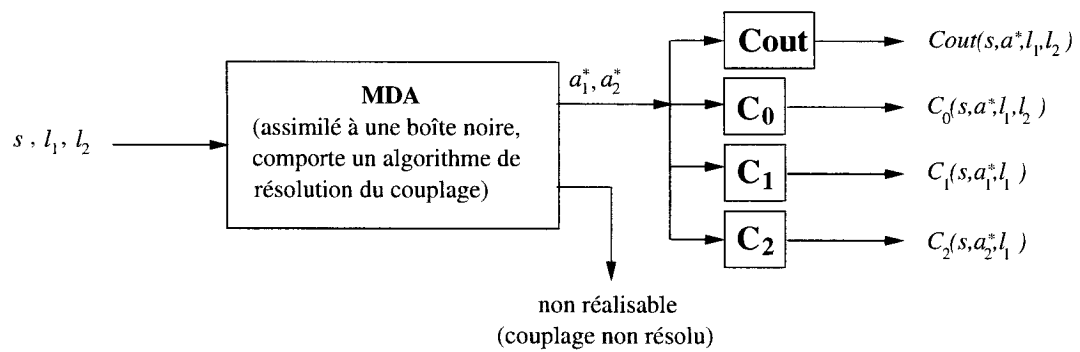


FIGURE 3.7: Décomposition par la méthode FIO : évaluation des contraintes et de la fonction coût

itératives de type point fixe) se limite, dans la pratique, au minimum requis pour converger avec un niveau de précision ϵ acceptable. Dans ce cas, les résultats d'analyse du MDA comportent une marge d'erreur non négligeable qui s'amplifie lors du calcul (par différences finies) des dérivées de C_{out} , C_0 , C_1 et C_2 . Ce manque de précision peut compromettre grandement l'efficacité des algorithmes d'optimisation à gradients (Tribes *et al.* (2004)). De plus, il existe des cas dans lesquels le MDA ne peut être résolu par des algorithmes simples tels que celui du point fixe ou pour lesquels il n'existe pas de solution. Il peut s'agir d'une impossibilité mathématique, dans le cas des systèmes pour lesquels les méthodes itératives divergent, ou bien de limites d'ordre pratique : un nombre très élevé de variables couplées rendra la convergence trop lente. Dans ce cas, la méthode FIO ne peut s'utiliser. On peut cependant contourner ce problème, s'il ne survient que de temps en temps, en retournant $C_{out} = \infty$ (cette parade comporte cependant des inconvénients si des optimiseurs à gradient sont utilisés).

3.4.3 Distributed Analysis Optimization

Une nouvelle architecture, impliquant un couplage serré entre l'optimisation et les analyses, est requise pour augmenter l'efficacité de la décomposition, tout en simplifiant la structure des problèmes de design multidisciplinaires complexes (comportant un grand nombre de variables de design) nécessitant des analyses dispendieuses. Certaines méthodes de décomposition du système reposent sur l'utilisation de contraintes de compatibilité pour simplifier la structure d'un MDA et permettre l'exécution en parallèle des analyses (d'après Cramer *et al.* (1993)).

L'illustration de cette approche est facilitée par l'utilisation de diagrammes de dépendance représentant le flux d'information entre les différentes disciplines d'un MDA. Chaque discipline est alors symbolisée par une boîte grisée susceptible de

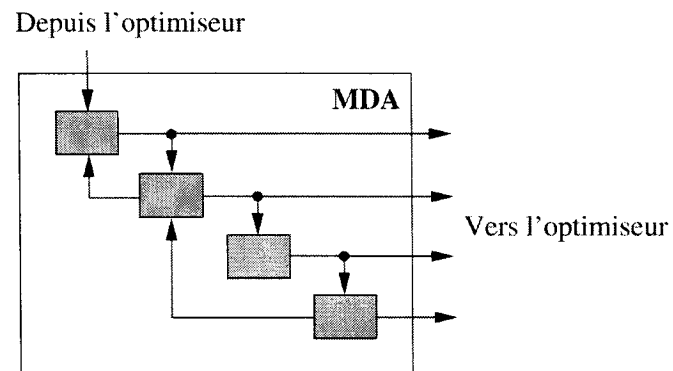


FIGURE 3.8: Exemple de diagramme de dépendance pour un MDA à quatre disciplines couplées

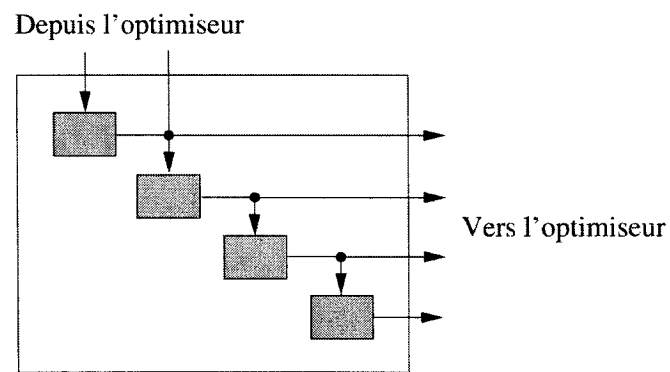


FIGURE 3.9: Système d'analyses sans itération et séquentiel

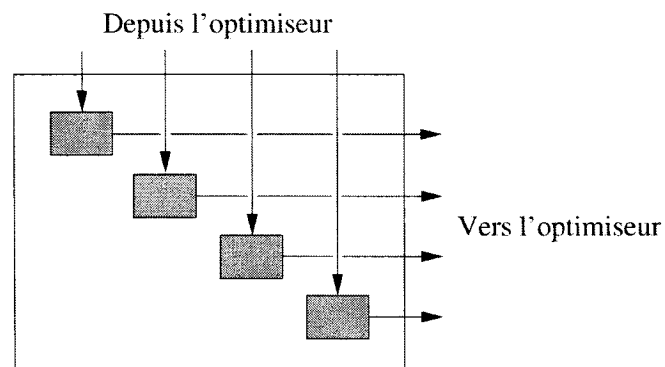


FIGURE 3.10: Système d'analyses sans itération et parallèle

comporter des données en entrée (flèches arrivant sur les faces supérieures et inférieures) ainsi qu'en sortie (flèches partant des faces gauche et droite). La figure 3.8 présente par exemple un MDA de 4 analyses couplées comportant des transferts de données dans les 2 directions avant et arrière. Les boucles de retour sur le diagramme de dépendance traduisent le couplage interdisciplinaire.

Certains outils tels que DeMaid (d'après Park et Choi (2003), Rogers (1992) et Rogers et Bloebaum (1994)) peuvent être utilisés pour réorganiser et agencer les disciplines au sein du MDA afin de minimiser le nombre et la longueur des boucles de retour. Toutefois, ils ne les suppriment habituellement pas complètement et il reste nécessaire de recourir à des calculs mathématiques additionnels pour la résolution. Recourir à l'optimiseur pour forcer la résolution des contraintes de couplage permet de ramener ces systèmes complexes à des diagrammes directs et acycliques. S'ajoutant à la liste des variables de design accessibles à l'optimiseur, des variables auxiliaires t sont alors requises pour toutes celles couplées c . Une contrainte de compatibilité $C_{compatibilite}$ (dite aussi de consistance interdisciplinaire), s'ajoute alors à la formulation directe du problème d'optimisation. Elle représente en quelque sorte le critère de convergence géré par l'optimiseur comme une contrainte d'égalité:

$$C_{compatibilite} = c - t = 0 \quad (3.5)$$

Cette approche permet de supprimer les boucles de retour et donc les itérations qui constituaient un handicap à l'emploi de la décomposition FIO. Le nouvel agencement des analyses au sein du MDA est illustré sur la figure 3.9.

L'emploi de cette même technique pour supprimer les boucles avant permet de

transformer le système d'analyses en celui, simplifié, de la figure 3.10. Ainsi, cette forme de décomposition conduit à un système d'analyse non seulement sans itération mais aussi parallèle (les analyses peuvent être effectuées en parallèle car elles n'échangent plus directement de données). À l'optimiseur revient la tâche de satisfaire les contraintes de consistance (tout au moins à hauteur d'une tolérance de faisabilité ϵ définie à l'avance) et donc le couplage interdisciplinaire.

La méthode de décomposition dite Distributed Analysis Optimization (DAO) (initialement baptisée MultiDisciplinary Feasible ou MDF par Cramer *et al.* (1993)) est basée sur ce principe (c.f. figure 3.11). Appliquée à l'exemple de l'optimisation d'un MDA de deux disciplines couplées, la formulation du problème d'optimisation devient alors :

$$\underset{s, a, l_1, l_2}{\text{maximise}} \text{ } Cout(s, a, l_1, l_2) \quad (3.6)$$

$$\text{tel que : } C_0(s, a, l_1, l_2) \geq 0,$$

$$C_1(s, a_1, l_1) \geq 0, \quad C_2(s, a_2, l_2) \geq 0,$$

$$t_1 - c_{21} = 0 \text{ et } t_2 - c_{12} = 0, \quad (3.7)$$

$$\text{avec : } c_{12} \subset a_1, \quad c_{21} \subset a_2,$$

$$a_1 = a_1(s, l_1, t_1) \text{ et } a_2 = a_2(s, l_2, t_2)$$

Par rapport à la méthode FIO, DAO requiert que l'optimiseur gère un nombre plus élevé de variables de design. Par contre, DAO permet d'effectuer toutes les analyses indépendamment et simultanément sur plusieurs ordinateurs distincts. L'équivalence mathématique entre le problème initial et sa formulation DAO est établie. De plus, il est possible d'utiliser pour l'optimisation des algorithmes qui maintiennent les contraintes interdisciplinaires quasiment vérifiées tout en maximisant la fonction coût. Ceci permet plus de flexibilité dans la recherche de l'optimum et donc

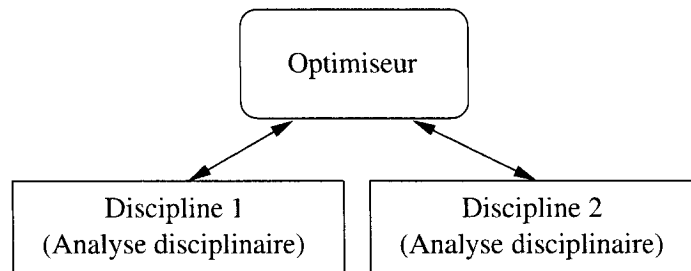


FIGURE 3.11: Décomposition DAO appliquée à un MDA de deux disciplines

d'obtenir des designs faisables, ou presque, avant que l'optimisation n'ait convergé (la direction de recherche pour améliorer la fonction coût vérifie alors $t - c = 0$).

Toutefois, DAO est basé sur une très forte centralisation de toutes les tâches au niveau de l'optimiseur. Le rôle de chaque discipline est alors uniquement d'effectuer une analyse et de transférer les résultats à un optimiseur unique. Ce mode de fonctionnement ne permet pas d'exploiter les procédures de design et d'optimisation (liberté de choix des variables de design, contraintes et fonctions de coût spécifiques) susceptibles d'avoir été développées, au cours des années, par les spécialistes de chaque discipline.

3.4.4 Distributed Optimizations

L'ambition des méthodes de décomposition multi-niveaux est de formuler, puis de résoudre des problèmes de MDO d'une façon qui préserve l'autonomie de chaque discipline en éliminant de l'optimiseur système (optimiseur unique utilisé pour les méthodes à 1 niveau) toutes les variables locales qui ne sont utilisées que par une seule discipline (d'après Alexandrov et Lewis (2000)). À cette fin, le problème originel se voit décomposé en plusieurs sous-problèmes. La coordination entre ceux-ci est assurée par un optimiseur maître dont l'une des fonctions est de résoudre le

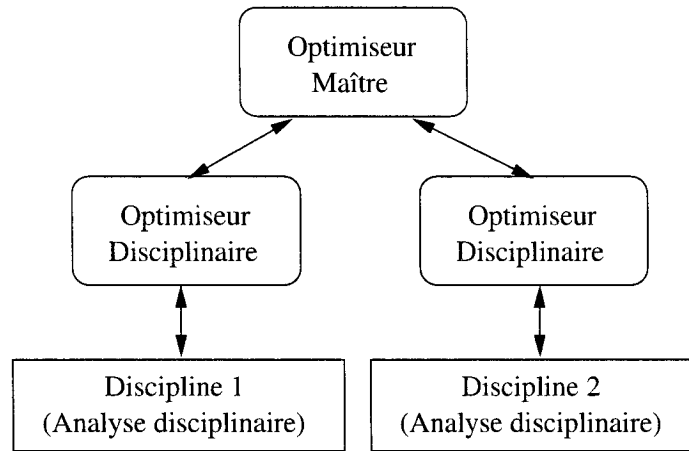


FIGURE 3.12: Décomposition hiérarchique par une méthode multi-niveaux

couplage entre les disciplines. Ce mode de fonctionnement est donc basé sur une décomposition hiérarchique du problème multidisciplinaire; les optimisations disciplinaires (dites aussi de bas-niveau ou également locales) ne sont reliées entre elles que par un optimiseur maître (voir la figure 3.12). Cette organisation permet d'utiliser des formulations éprouvées qui respectent les limites naturelles des disciplines et l'expertise des spécialistes. Enfin, elle rend possible l'optimisation simultanée des sous-problèmes.

Plusieurs méthodes de décomposition multi-niveaux ont successivement été proposées et appliquées avec bénéfices à des problèmes de MDO. Les Collaborative Optimization (CO)(Braun (1996)), Inexact Penalty Decomposition (IPD) ou autres Exact Penalty Decomposition (EPD)(De Miguel (2001)) en sont de bonnes illustrations. Certaines d'entre elles (CO par exemple) présentent toutefois des spécificités qui, pour certains types de problèmes, rendent la convergence difficile (voir Alexandrov et Lewis (2000)). La récente méthode Distributed Optimizations (DO), proposée par Tribes *et al.* (2004) et présentée en détails ci-après, constitue une alternative intéressante.

Pour les développements ci-dessous, nous supposons que la fonction coût du problème ingénieur originel ne dépend pas explicitement des variables locales l_1 et l_2 (dans le cas contraire, la décomposition DO ne peut s'appliquer directement). Partant de la décomposition DAO d'un problème de deux analyses couplées (voir §3.4.3), la formulation équivalente suivante a été obtenue en séparant la fonction coût initiale en 2 termes, associés chacun à une discipline. Comme pour la formulation DAO, des contraintes de consistance $t_i - a_i = 0$ doivent être appliquées aux variables couplées. Elles permettent à l'optimiseur maître d'assurer la résolution du couplage interdisciplinaire. Avec $a_1 = A_1(s, l_1, t_1)$ et $a_2 = A_2(s, l_2, t_2)$:

$$\begin{aligned}
 & \underset{s, t_1, t_2, l_1, l_2}{\text{maximise}} \quad \frac{1}{2} (\text{Cout}(s, a_1, t_2) + \text{Cout}(s, t_1, a_2)) & (3.8) \\
 & \text{tel que : } C_{01}(s, a_1, t_2) \geq 0, \quad C_{02}(s, t_1, a_2) \geq 0, \\
 & \quad C_1(s, a_1, l_1) \geq 0, \quad C_2(s, a_2, l_2) \geq 0, \\
 & \quad t_1 - a_1 = 0, \quad t_2 - a_2 = 0
 \end{aligned}$$

Lorsque les variables globales s , t_1 et t_2 sont fixées, le problème d'optimisation ci-dessus peut être séparé en deux sous-problèmes indépendants :

$$\begin{aligned}
 & \underset{l_1}{\text{maximise}} \quad \frac{1}{2} (\text{Cout}(s, a_1, t_2)) & (3.9) \\
 & \text{tel que : } C_{01}(s, a_1, t_2) \geq 0, \\
 & \quad C_1(s, a_1, l_1) \geq 0, \\
 & \quad t_1 - a_1 = 0
 \end{aligned}$$

$$\begin{aligned}
& \underset{l_2}{\text{maximise}} \quad \frac{1}{2} (\text{Cout}(s, t_1, a_2)) & (3.10) \\
& \text{tel que : } C_{02}(s, t_1, a_2) \geq 0, \\
& \quad C_2(s, a_2, l_2) \geq 0, \\
& \quad t_2 - a_2 = 0
\end{aligned}$$

Ici encore, les contraintes de couplage $t_i - a_i = 0$ servent à résoudre le couplage entre les analyses disciplinaires. Cependant, elles sont très restrictives lorsqu'elles s'appliquent aux sous-problèmes. Il est préférable de les reporter au niveau de l'optimiseur maître. On obtient alors le sous-problème I suivant :

$$\begin{aligned}
& \underset{l_1}{\text{maximise}} \quad \frac{1}{2} (\text{Cout}(s, a_1, t_2)) & (3.11) \\
& \text{tel que : } C_{01}(s, a_1, t_2) \geq 0, \\
& \quad C_1(s, a_1, l_1) \geq 0
\end{aligned}$$

et le sous-problème II :

$$\begin{aligned}
& \underset{l_2}{\text{maximise}} \quad \frac{1}{2} (\text{Cout}(s, t_1, a_2)) & (3.12) \\
& \text{tel que : } C_{02}(s, t_1, a_2) \geq 0, \\
& \quad C_2(s, a_2, l_2) \geq 0
\end{aligned}$$

Pour un s donné, si un design faisable existe, le sous-problème I fournit les valeurs optimales suivantes : $\text{Cout}_1^+(s, t_1, t_2) = \text{Cout}_1(s, a_1^+, t_2)$, $C_{01}^+ = C_{01}(s, a_1^+, t_2)$ et $C_1^+ = C_1(s, a_1^+, l_1^+)$. De même, pour le sous-problème II : Cout_2^+ , C_{02}^+ et C_2^+ . L'équivalence avec la formulation DAO initiale est vérifiée si $t_1 = a_1^+$ et $t_2 = a_2^+$. La responsabilité de trouver t_1 et t_2 vérifiant ces conditions est laissée à l'optimiseur

maître :

$$\begin{aligned}
& \underset{s, t_1, t_2}{\text{maximise}} \quad \frac{1}{2} (\text{Cout}_1^+(s, t_1, t_2) + \text{Cout}_2^+(s, t_1, t_2)) & (3.13) \\
& \text{tel que : } C_{01}^+(s, t_1, t_2) \geq 0, \quad C_{02}^+(s, t_1, t_2) \geq 0, \\
& \quad C_1^+(s, t_1, t_2) \geq 0, \\
& \quad C_2^+(s, t_1, t_2) \geq 0, \\
& \quad t_1 - a_1^+ = 0, \\
& \quad t_2 - a_2^+ = 0
\end{aligned}$$

Étant donné qu'elles dépendent également de variables globales fixées par l'optimiseur maître, les contraintes C_{01} , C_{02} , C_1 et C_2 peuvent ne pas être satisfaites au niveau des sous-problèmes. Dans ce cas, le sous-problème correspondant ne peut aboutir à une solution faisable et c'est alors la moins infaisable qu'il renvoie à l'optimiseur maître. C'est à lui que revient la charge de trouver s , t_1 et t_2 pour que les sous-problèmes conduisent à des optimums (donc des designs faisables).

3.5 Les défis de la décomposition automatique

Tel qu'illustré par les exemples précédents, il existe plusieurs façons de décomposer un problème de MDO. S'il est mathématiquement équivalent au problème ingénieur originel, le MDO décomposé repose bien souvent sur un système MDA grandement modifié (c.f. l'exemple des figures 3.8, 3.9 et 3.10).

Le choix de la décomposition à adopter n'est pas anodin. En effet, de nombreuses études analytiques ainsi que des expérimentations numériques (à l'aide de cas tests) indiquent que la formulation des problèmes de MDO a une influence directe sur l'efficacité de l'optimisation subséquente. La méthode de décomposition utilisée a

donc de profondes conséquences sur le choix d’algorithmes de résolution adaptés et donc sur l’efficacité du processus d’optimisation (d’après Alexandrov et Lewis (2000)).

Ce choix n’est pas trivial non plus. Dans le cas de problèmes de MDO réels, il est souvent difficile de déterminer à priori si une formulation donnée va conduire à des résultats satisfaisants (d’après Alexandrov et Lewis (2004)). Dans la pratique, le choix dépend du problème formulé, et, plus spécifiquement, de nombreux critères tels que le couplage entre les analyses ou les moyens à disposition pour l’optimisation (les méthodes permettant les optimisations disciplinaires simultanées se justifient surtout s’il est possible d’utiliser plusieurs ordinateurs mis en réseau).

En conséquence, et étant donné la variété des tâches auxquelles le PMDO est destiné (cf. Chapitre 2), il est nécessaire de le concevoir de façon à ce qu’il offre un choix parmi plusieurs méthodes de décomposition. Sa structure doit comporter la flexibilité qui lui permettra de s’adapter automatiquement tant à une méthode multi-niveaux qu’à une décomposition de type FIO. Ce besoin de configuration automatique de PMDO est un requis majeur qui en oriente fondamentalement la conception. La flexibilité liée à la configuration automatique pour l’une ou l’autre des méthodes de décomposition doit porter sur plusieurs éléments :

1. Le nombre d’optimiseurs : Selon la méthode de décomposition et le *Working_Mode* choisis, chaque discipline incluse dans le MDA peut être associée à une optimisation disciplinaire ou bien à une simple analyse.
2. La nature de chaque optimiseur : Le choix de l’optimiseur utilisé dépend de l’analyse correspondante.
3. Un moteur de résolution de couplage : Pour permettre d’utiliser la méthode FIO, PMDO doit comporter (au besoin) un moteur de résolution de couplage

(tel que l'algorithme de Gauss-Seidel) chargé de déterminer, par itérations successives, la valeur des variables couplées du vecteur c .

De plus, pour permettre l'utilisation des méthodes de type DAO ou DO,

5. La création de variables de couplage : Pour les méthodes devant découpler les disciplines, il est nécessaire de dupliquer une partie des variables de sortie a_i (c'est-à-dire les c_i) pour créer les t_i requis dans la formulation décomposée.
6. Les contraintes de consistance : Des contraintes de type $a_i - t_i = 0$ doivent être appliquées à l'optimiseur chargé de résoudre le couplage interdisciplinaire.

Enfin, pour les méthodes multi-niveaux,

7. La définition de fonction coût et des contraintes disciplinaires pour chaque optimiseur : Les fonctions de mérite ou contraintes à imposer à chaque optimiseur (maître ou disciplinaire) dépendent de la décomposition utilisée.

En conclusion, il apparaît que la configuration automatique d'une plate-forme d'intégration ainsi que d'optimiseurs en fonction des méthodes de décomposition constitue un défi de grande importance pour la réalisation de PMDO. À notre connaissance, aucune publication scientifique ne fait part de réalisations concrètes de ce genre. Notre approche, s'appuyant sur certains concepts introduits très récemment par Alexandrov et Lewis (2004), constitue probablement l'un des premiers essais de programmation d'un outil flexible de plate-forme de MDO à configuration semi-automatique. Les chapitres 4 et 5 suivants apporteront une description détaillée du prototype développé et de son principe de fonctionnement.

CHAPITRE 4

CONCEPTION ET FONCTIONNEMENT DE PMDO

Ce chapitre est consacré aux descriptions précises de la structure et du mode de fonctionnement de PMDO. Il présente donc l'outil logiciel conçu en fonction des requis rassemblés dans la première phase du projet. PMDO doit s'adapter à de nombreux cas d'utilisation. La flexibilité par rapport au système intégré, au problème traité et à la méthode de décomposition à utiliser constitue un important défi de la conception. De plus, PMDO doit également être transférable d'un système d'exploitation à l'autre ainsi que suffisamment évolutif pour accepter l'ajout ultérieur de disciplines et de nouvelles méthodes de décomposition. La conception et le développement du prototype constituent le cœur du travail de maîtrise.

Étant donné l'ampleur du projet, la réalisation de certains éléments logiciels, précisément définis au préalable, a été confiée à d'autres étudiants de génie (informatique et mécanique) qui en ont fait leur travail de Projet de Fin d'Études (PFE). Leurs contributions sont à chaque fois soulignées.

Sont également précisées ci-après les spécificités et limites d'utilisation du prototype actuel.

Faisant l'objet d'une contribution méritant d'être traitée en détails, la présentation explicite des mécanismes de configuration automatisée de PMDO en fonction d'une méthode de décomposition fera l'objet du chapitre 5.

4.1 Le développement d'un prototype de PMDO

4.1.1 Les spécificités du prototype

Compte tenu du temps imparti et de restrictions quant à l'utilisation des outils d'analyse de Pratt & Whitney Canada, rappelons que l'objectif de cette maîtrise recherche n'est pas de produire directement un nouvel outil opérationnel. Il s'agit plutôt de concevoir un prototype novateur faisant office de preuve de concept et soumis à de nombreux tests (jugé tant objectivement du point de vue de l'efficacité des optimisations que de celui de l'ergonomie, par des futurs utilisateurs chez P&WC) avant modifications et implantation finale. Le prototype présenté ci-après est bâti en fonction des requis (objectifs et obligations) établis initialement.

Une partie importante des choix de conception posés découle notamment de l'obligation d'utiliser le logiciel iSIGHT de Engineous Software en guise de moteur d'optimisation. De plus, bien que conçue pour s'accommoder de tous les types d'outils nécessaires (notamment des exécutables interagissant avec des fichiers texte), la plate-forme d'intégration constituée ne comporte, pour l'instant, que des feuilles de calcul Excel.

4.1.2 La réalisation du projet

Si les phases d'étude, d'expérimentations et enfin de conception constituèrent un travail de recherche effectué seul, la réalisation de la version actuelle du prototype est un travail d'équipe. Mon rôle dans cette ultime étape fut de concevoir le logiciel PMDO dans son ensemble, d'en programmer la partie reliée à l'optimisation (tous les codes de iSIGHT) tout en coordonnant et orientant le travail de 4 étudiants de baccalauréat (en Génie Informatique et Génie Mécanique) dont la programmation

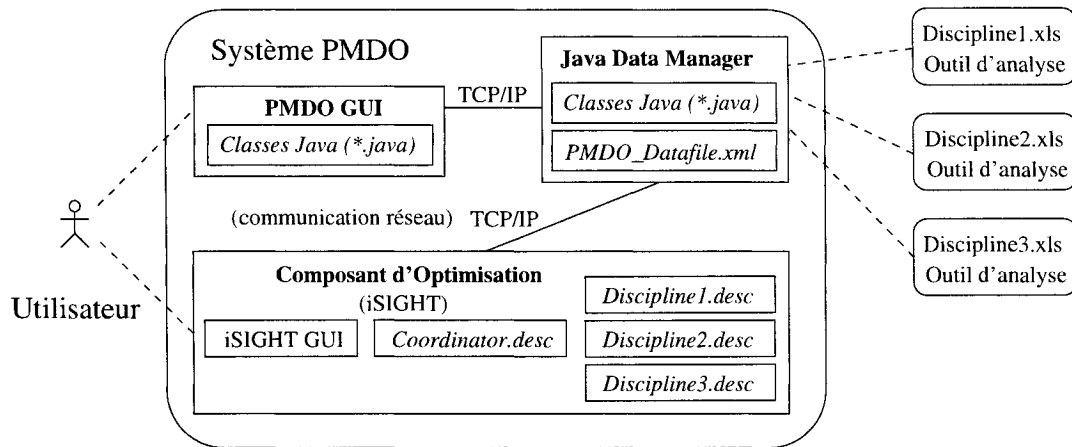


FIGURE 4.1: Architecture générale du prototype

de sous-ensembles constituait le travail de PFE. Les noms de ces étudiants (Harish Ganoory, Simon Labarre, Jean Maurais et Daniel Tardif) seront rappelés dans les sections traitant des composants logiciels auxquels ils ont travaillé.

4.2 L'architecture de PMDO

4.2.1 Principe général

4.2.1.1 Une structure à trois composants

Le logiciel PMDO comporte trois composants autonomes : le Java Data Manager (JDM), l'interface graphique PMDO_GUI (Graphical User Interface) et le composant d'optimisation. L'ensemble interagit avec des outils d'analyse indépendants présents sur le réseau. La figure 4.1 présente l'architecture générale du prototype.

Ces trois composants peuvent être installés sur des ordinateurs distincts. La communication entre eux s'établit à distance selon le principe clients/serveur et uti-

lise le protocole TCP/IP. Au cœur de la plate-forme d'intégration, un fichier *PMDO_Datafile.xml* sert de relais pour le partage des données : constamment remis à jour au cours de l'optimisation et accessible uniquement au JDM, ce fichier regroupe toute l'information décrivant un design donné.

4.2.1.2 Une exécution en trois phases

Le déroulement d'une session d'utilisation de PMDO comporte trois étapes séquentielles : le pré-traitement, l'exécution et enfin le post-traitement.

Le pré-traitement est l'étape commençant par le recueil de l'information décrivant le problème posé, le système auquel il s'applique et la méthode à utiliser pour sa résolution. Compte tenu de ces données, PMDO passe alors par une très importante phase de configuration automatique. Le MDA de travail est alors constitué par l'intégration automatique de tous les outils d'analyse nécessaires. Un fichier texte associé à chaque discipline, le *Properties_File* (par exemple *Discipline1.properties*), est alors utilisé pour établir automatiquement la liste des variables de design correspondant au problème MDO en cours de configuration. La phase de configuration se termine par la définition automatisée, dans le logiciel iSIGHT, de toutes les tâches d'analyse ou d'optimisation nécessaires à la résolution du problème. Dès lors, PMDO s'est entièrement configuré pour l'exécution.

La phase d'exécution est orchestrée entièrement par iSIGHT. Il s'agit habituellement d'un processus itératif au cours duquel plusieurs centaines d'analyses disciplinaires sont automatiquement effectuées en réponse aux requêtes du (des) optimiseur(s). À moins d'une intervention de l'utilisateur, la fin de la phase d'exécution est déterminée par le composant d'optimisation en fonction de critères de convergence. La fin de l'exécution entraîne immédiatement le lancement de l'étape de

post-traitement.

Consécutive à l'ouverture d'une interface graphique adaptée, l'étape de post-traitement est celle au cours de laquelle le(s) résultat(s) de l'exécution est(sont) présenté(s) à l'utilisateur sous forme de chiffres ou de graphiques tel que la coupe simplifiée de certains composants du moteur. Par l'intermédiaire de cette même interface, l'utilisateur de PMDO peut décider de sauvegarder chaque solution ou bien de relancer le processus afin d'améliorer encore un design.

Faisant suite à ces descriptions préliminaires, les sections suivantes comportent les détails relatifs à la constitution et au fonctionnement de chaque composant logiciel et de chacune des étapes d'exécution.

4.2.2 Le composant Java Data Manager

4.2.2.1 Programmation

Le JDM joue un rôle central dans la plate-forme d'intégration. Ce composant logiciel a été programmé en langage Java afin de bénéficier de plusieurs atouts :

- C'est un langage portable (Java est disponible et installé sur toutes les plateformes susceptibles d'être utilisées chez Pratt & Whitney Canada, un même code Java peut donc être utilisé sur chacune d'entre-elles;
- Le nombre de bibliothèques disponibles (gratuitement ou avec licence) et utiles au projet est très élevé;
- C'est un langage largement diffusé, déjà utilisé chez P&WC et ayant fait ses preuves.

La programmation des classes Java constituant le JDM a été confiée à Harish Ganoory (multiserveur) et Simon Labarre (exécution des analyses Excel), tous deux

étudiants en génie informatique. Le JDM a été développé selon les directives issues de la conception de PMDO dans son ensemble.

4.2.2.2 Le multiserveur

Le JDM comporte tout d'abord un multiserveur, baptisé *MultiServer.java* (correspondant à la classe *MultiServer.java*) dont la fonction est de permettre la communication avec le composant d'optimisation ainsi que l'interface graphique PMDO_GUI. Le multiserveur est à l'écoute d'un port unique (choisi une fois pour toutes par un administrateur) par lequel il peut échanger des messages avec plusieurs clients simultanément. Les clients et le multiserveur utilisent le protocole TCP/IP pour constituer des messages selon une syntaxe convenue à l'avance.

4.2.2.3 Le fichier unique de partage de données

Au cœur de la plate-forme d'intégration, le fichier XML (eXtensible Markup Language) *PMDO_Datafile.xml* est l'unique document dans lequel toutes les données décrivant le MDO sont regroupées, conservées puis redistribuées vers les clients utilisateurs. Ce fichier unique permet ainsi le partage de l'information entre les différents outils d'analyse intégrés dans le MDA ainsi que les autres composants (tels que les optimiseurs ou les interfaces graphiques). Constitué au cours de l'étape de pré-traitement, ce fichier est constamment mis à jour au fil des itérations de la phase d'exécution.

XML est un métalangage informatique, actuellement très utilisé (entre autres pour les applications Internet), permettant de construire des documents dont le contenu est identifié par des balises définies au préalable. Contrairement au protocole HTML

(Hyper Text Meta Language), qui est un langage figé, avec un nombre de balises limité, XML peut être considéré comme un métalangage permettant de définir d'autres langages, c'est-à-dire de créer de nouvelles balises pour décrire le contenu d'un fichier. Il permet de structurer, de poser le vocabulaire et la syntaxe des données qu'il va contenir.

Dans le cadre de PMDO, le standard XML a été retenu pour plusieurs de ses qualités intrinsèques :

- XML permet de constituer automatiquement des fichiers de données standardisées qui peuvent facilement être partagés par plusieurs applications tout en restant lisibles pour les humains. Cette qualité facilite grandement les étapes de développement et de tests, mais elle n'est toutefois pas essentielle pour l'utilisateur final pour qui le fichier XML reste caché en tout temps.
- Il existe plusieurs librairies Java permettant de constituer et de manipuler des fichiers XML avec une grande efficacité. Les étapes de recherche de données, de lecture ou d'écriture sont donc simplifiées.
- Certains logiciels existants permettent de produire automatiquement des documents de présentation (rapports et graphiques par exemple) à partir de fichiers XML contenant les données.
- Enfin, les fichiers XML peuvent être validés par comparaison avec des schémas contenant des règles de syntaxe et d'orthographe associées au contenu. Ceci permet de détecter des erreurs dans l'ordonnancement des données ou bien dans leur nature même (par exemple un nombre réel ou un caractère placé erronément dans un champ devant contenir un entier).

Une structure originale ainsi qu'un ensemble de balises spécifiques ont été définis pour le fichier *PMDO_Datafile.xml*. Ainsi, les données sont agencées par groupes (appelés *Blocks*) et sous-groupes en fonction de leur origine ou de leur utilisation. Les quatre types de groupes constituant le fichier *PMDO_Datafile.xml* sont :

- Le *GULBlock* regroupe l'ensemble des informations recueillies dans l'interface graphique PMDO_GUI. Il s'agit par exemple de la configuration du moteur, de la liste des disciplines incluses ainsi que des adresses des fichiers d'entrées correspondants, du type de problème traité (Optimisation ou DOE) et de la méthode de décomposition utilisée pour sa résolution;
- Plusieurs *Discipline_Blocks* (un pour chacune des disciplines incluses pour l'exécution en cours) contenant toute l'information relative aux variables utilisées, ou produites, par une discipline donnée. Les *Discipline_Blocks* rassemblent donc des variables décrites chacune par quinze attributs tels que le nom, la valeur, la nature (*input* ou *output*) ou bien la catégorie (voir ci-dessous pour une description détaillée des attributs);
- Un unique *Variables_Block*, dont la structure et les attributs sont semblables à ceux des *Discipline_Blocks*, mais contenant la liste complète des variables du système ou MDA constitué;
- Un *PPGULBlock*, acronyme de Post Processing Graphical User Interface Block, reçoit l'ensemble de l'information relative à la dernière étape de post-traitement. Il s'agit alors essentiellement de commentaires.

De plus, des attributs sont utilisés pour définir chaque variable du *Variables_Block* (il existe quelques différences mineures dans le cas des *Discipline_Blocks*). La liste suivante en recense les principaux :

- *key* : il s'agit là de l'identifiant unique de chaque variable du MDO. En effet, une même variable peut malheureusement exister sous des noms différents selon l'outil d'analyse dans lequel elle est utilisée (le nom ne peut donc pas servir à reconnaître un couplage automatiquement); de plus, certains noms utilisés par les ingénieurs comportent des caractères spéciaux, tels que "-", "_", "(" ou bien "/" , ce qui entraîne des difficultés de traitement pour certains algorithmes de PMDO.

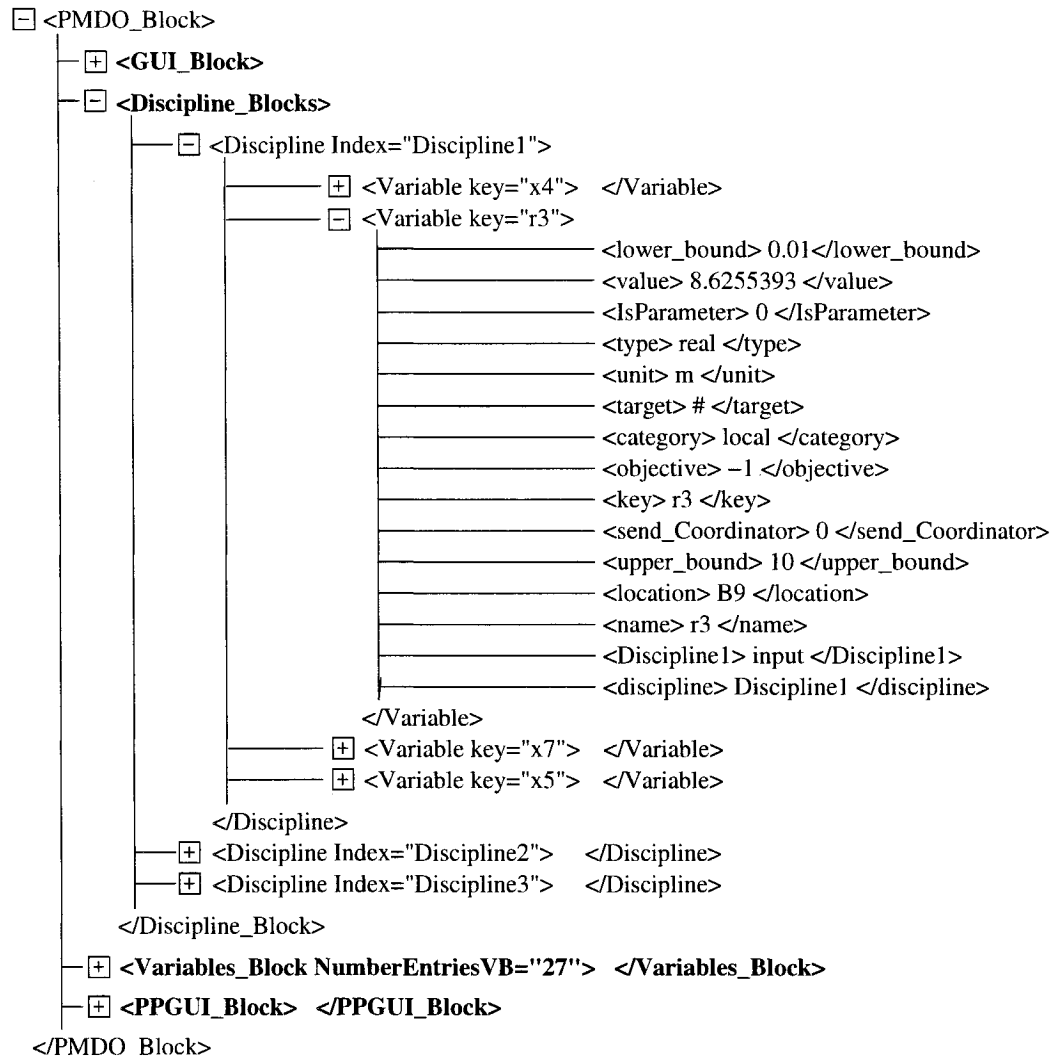
- *name*, *value*, *unit* et *type* : ces attributs servent à décrire chaque variable.
- *category* et *discipline_i* : le champ *category* précise le statut *local*, *shared* ou *coupled* de chaque variable. Les attributs *discipline_i* (*i* étant le numéro d'une discipline intégrée) en indiquent quant à eux la nature (*input* ou *output*).
- *objective*, *target*, *lower_bound*, *upper_bound* et *IsParameter* : Ces attributs sont tous relatifs à l'optimisation dans laquelle peut intervenir la variable concernée. Ils précisent, entre autres, les bornes de l'ensemble de design.

La figure 4.2 est un exemple de l'arborescence montrant l'agencement des différents groupes (ou *blocks*) de données au sein d'un fichier *PMDO_Datafile.xml*.

La classe *Datafile_Manager.java* du JDM comporte les méthodes permettant de construire le fichier *PMDO_Datafile.xml* au cours du pré-traitement, puis d'accéder (en lecture et écriture) à l'information s'y trouvant. Le JDM est l'unique composant de PMDO qui puisse utiliser ce fichier, il reçoit donc (par l'intermédiaire de messages TCP/IP échangés via le réseau et transmis par l'objet *MultiServer*) les requêtes des autres composants (interface graphique PMDO_GUI et composant d'optimisation). Ce mode de fonctionnement comporte, entre autres, l'avantage de minimiser le risque de corruption des données en les centralisant. La classe *Datafile_Manager.java* utilise la librairie DOM4J pour constituer les méthodes XML.

4.2.2.4 L'exécution des analyses disciplinaires

Pour PMDO, le Java Data Manager est également l'interface unique avec tous les outils d'analyse intégrés. En réponse à une requête du composant d'optimisation, le JDM a la responsabilité de commander une analyse disciplinaire utilisant en entrée certaines des variables du fichier *PMDO_Datafile.xml*. Les résultats de l'analyse effectuée doivent alors, eux aussi, être écrits dans un *Discipline_Blocks* du fichier XML. La séquence d'exécution est détaillée dans la section 4.3.2.1.

FIGURE 4.2: Structure interne d'un fichier *PMDO_Datafile.xml*

Le composant JDM doit donc comporter les classes et bibliothèques nécessaires pour interagir avec chacun des nombreux outils intégrés à PMDO. À l'heure actuelle et pour le prototype décrit, seule a été mise en place l'interface avec les feuilles de calcul Excel. Ce travail a fait l'objet du PFE de Simon Labarre (élève de génie informatique).

Après évaluation de plusieurs outils ou algorithmes, la solution retenue et mise en place utilise une bibliothèque Java Spreadsheet Component¹ (JSC) pour laquelle une licence d'exploitation a été achetée par Pratt & Whitney Canada.

Un fichier texte descriptif, appelé *Properties_File* et portant l'extension *.properties*, est établi pour chacun des outils d'analyse disponibles sous forme de feuille de calcul Excel. Créé une fois pour toutes, par l'administrateur, au moment de l'intégration de la discipline dans PMDO, ce fichier texte comporte la liste exhaustive des variables nécessaires à l'analyse (variables d'entrée et de sortie) ainsi que certaines informations associées telles que les coordonnées de la cellule correspondante dans la feuille de calcul. Lu au cours du pré-traitement, le fichier *Disciplinei.properties* sert au JDM pour établir la liste des variables requises pour l'exécution de l'analyse disciplinaire *i*. Cette information sert tout d'abord à constituer et initialiser le *Discipline_Blocks* correspondant à la discipline *i*. Enfin, la bibliothèque JSC contient les méthodes appelées pour lire et interpréter la feuille de calculs *Disciplinei.xls* afin de créer un objet Java reproduisant les opérations de l'analyse disciplinaire. Au terme de la phase de pré-traitement, le Java Data Manager crée un objet *Disciplinei.java* pour chaque discipline incluse pour l'exécution en cours. L'emploi de la bibliothèque Java Spreadsheet Component permet donc d'effectuer, sur la machine virtuelle Java, les mêmes calculs que le fichier **.xls* d'origine sans pour autant utiliser le programme Excel (logiciel difficilement disponible sur les plate-formes

¹Voir une description du produit à l'adresse <http://jsc.tomschuetz.com/> .

Unix et Linux). Il existe toutefois certaines restrictions quant aux opérations que la librairie JSC est capable de reproduire. Les fichiers Excel contenant certaines macros incompatibles avec la version actuelle de JSC doivent être adaptés avant leur intégration à PMDO.

Des développements ultérieurs devraient permettre au prototype d'intégrer, en sus des feuilles de calcul Excel, des outils du type exécutables. Ces programmes utilisent des fichiers textes (actuellement établis à la main par les ingénieurs utilisateurs) pour connaître les valeurs des variables d'entrée. Une nouvelle classe Java est donc requise pour générer automatiquement ces fichiers, au format complexe, à partir de l'information contenue dans le fichier *PMDO_Datafile.xml*. Un mécanisme similaire doit aussi être mis en place pour la lecture automatisée des fichiers textes de résultats.

4.2.3 Le composant d'interface graphique PMDO_GUI

4.2.3.1 Organisation et fonctionnement de l'interface graphique

Destiné à devenir un outil de design pour les ingénieurs de Pratt & Whitney Canada, PMDO doit comporter des interfaces graphiques permettant une utilisation aisée et conviviale. Le composant d'optimisation iSIGHT offre une interface suffisante pour les ajustements fins devant être effectués avant la phase d'exécution : modification des bornes de l'ensemble de design ou choix des algorithmes d'optimisation. Des interfaces supplémentaires doivent cependant être développées, notamment pour recueillir l'information relative au système MDO ou pour présenter des résultats sous forme de dessins de coupe moteur. Ainsi, une interface Java a été développée par Jean Maurais, étudiant de génie informatique, en réponse à des spécifications s'appuyant sur l'étude des requis effectuée en début de maîtrise.

Devant évoluer au fil de l'intégration de nouvelles disciplines dans PMDO (soit au fur et à mesure que des outils d'analyse préliminaire 0D/1D adaptés sont développés par chaque département de P&WC), l'interface graphique voit sa configuration établie automatiquement, à chaque utilisation, en fonction de l'information contenue dans un fichier texte *pmdo.properties*. Établi par l'administrateur, ce fichier de configuration comporte, par exemple, la liste complète des disciplines intégrées (et donc susceptibles d'être incluses pour une exécution donnée) ainsi que celle des champs à remplir pour chaque discipline. L'interface graphique PMDO_GUI est organisée en une succession de pages et d'onglets, chacun consacrée à une analyse disciplinaire.

Une fois recueillie, l'information nécessaire à l'exécution est transmise au Java Data Manager. Elle y est alors mise en forme à l'aide de balises XML pour constituer le *GULBlock* et donc le tout premier élément du *PMDO_Datafile.xml*. Cette étape correspond au début de la phase de configuration automatique du JDM (la séquence d'exécution est détaillée dans la section 4.3.1.2). L'interface graphique PMDO_GUI devient alors inactive jusqu'à la phase de post-traitement.

Les sections suivantes sont des présentations plus complètes des fonctions et de la structure de l'interface graphique en fonction des phases d'exécution.

4.2.3.2 Description du système et du problème

L'interface graphique (voir figure 4.3) est le premier composant avec lequel interagit l'utilisateur de PMDO. Au cours du pré-traitement, elle est utilisée avant tout pour décrire le système étudié, c'est-à-dire pour spécifier, parmi les disciplines intégrées, celles qui sont incluses ou utilisées pour l'exécution en cours. Les différentes pages de l'interface PMDO_GUI sont présentées ci-dessous dans l'ordre dans lequel elles

apparaissent.

Une première page générale permet de définir la configuration du moteur sur lequel porte le problème de MDO (type de moteur, nombre et type d'étages de compresseur, type de chambre de combustion, etc.). Cette information sert immédiatement à adapter l'interface graphique par la création d'une page dite *DisciplinePage* pour chacune des disciplines susceptibles de correspondre à la configuration décrite. Par exemple, une page sera offerte pour l'analyse structurelle d'une pale de soufflante si le moteur est de type turbosoufflante. De même, une discipline d'analyse structurelle sera incluse pour chaque étage de compresseur ou de turbine. La première page du PMDO_GUI comporte aussi des champs permettant d'identifier l'utilisateur et de nommer le projet en cours.

Dans chaque *DisciplinePage*, les champs suivants doivent être remplis par l'utilisateur :

- Le *Running_Mode* - Parmi les trois choix offerts, l'utilisateur choisira *Optimize* si les variables de design associées à la discipline considérée peuvent être modifiées au cours d'une optimisation du système. Il préférera *Use_As_Fixed* pour identifier une discipline pour laquelle le design initial ne peut-être changé en cours d'exécution (la discipline ainsi fixée applique donc une contrainte pour l'optimisation d'autres composants partageant certaines variables de design). Le choix *Do_Not_Use* sera utilisé lorsqu'une discipline ne doit pas être incluse au MDA en cours de constitution (pour une optimisation disciplinaire, par exemple, l'utilisateur choisira *Optimize* pour une discipline et *Do_Not_Use* pour toutes les autres).
- L'adresse d'un fichier correspondant au design initial - Il s'agit généralement du fichier d'entrée (texte) pour l'analyse disciplinaire considérée. C'est de ce fichier, fourni par l'utilisateur, que seront tirées les valeurs initiales des variables de

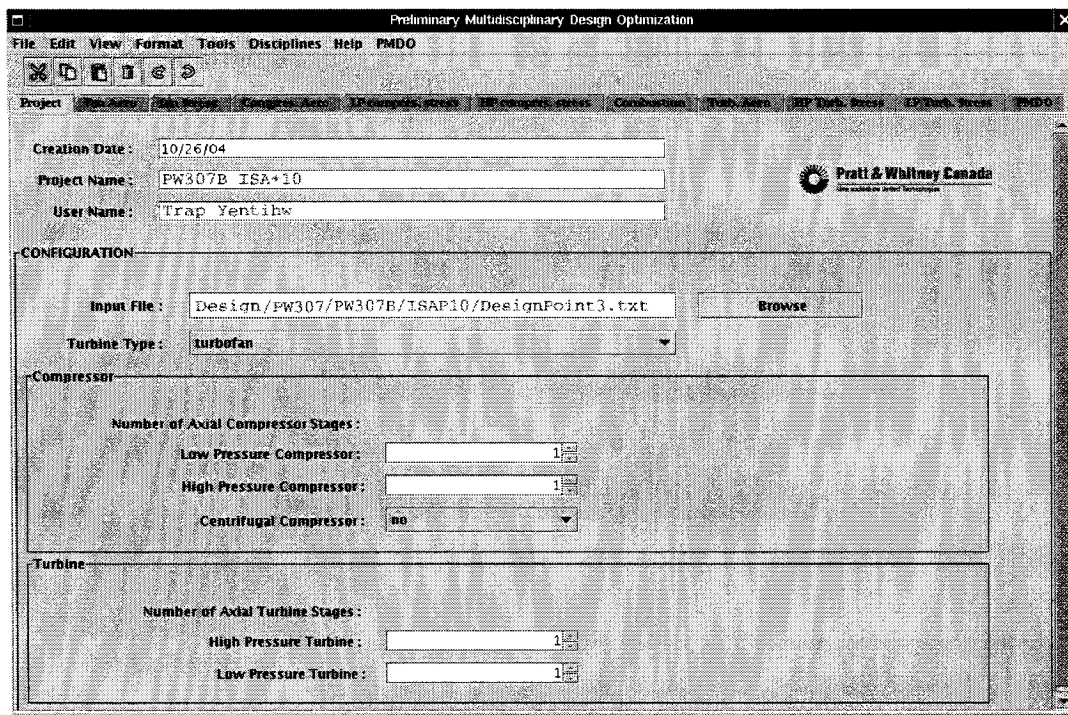


FIGURE 4.3: Exemple d'interface PMDO_GUI lors de la phase de pré-traitement

design pour la discipline considérée.

- Valeurs des bornes par défaut - Ces deux valeurs sont exprimées en pourcentages de différence par rapport à la valeur initiale de chaque variable. Dans le cas d'une optimisation, elles permettent un calcul automatique des valeurs inférieures et supérieures délimitant l'intervalle de design pour la discipline considérée.

Enfin, la dernière page de l'interface PMDO_GUI sert au choix du *Working_Mode*, c'est-à-dire de la nature du problème posé (optimisation, recherche de design faisable ou bien pour réaliser un plan d'expérience (DOE)(se reporter à la section 2.2.2 pour définition) et de la méthode de décomposition à utiliser.

Afin d'accélérer cette étape de définition du cas traité, un fichier XML issu d'une exécution PMDO passée peut être utilisé pour pré-remplir tous les champs de l'interface.

Une fois tous les champs remplis, l'utilisateur lance la phase de configuration automatique du JDM en cliquant sur le bouton prévu à cet effet. Cette action a aussi pour effet de fermer PMDO_GUI pour le remplacer par l'interface du composant d'optimisation iSIGHT.

4.2.3.3 Présentation et sauvegarde des résultats

Au cours du post-traitement, l'interface PMDO_GUI redevient active pour présenter les solutions sous forme de chiffres clefs et de graphiques (par exemple les coupes de composants ou bien une coupe générale du moteur). Un système de pages à onglets (voir la figure 4.4 pour illustration) permet de présenter simultanément les éventuels n designs faisables trouvés lors de l'exécution de PMDO. Pour chacune de ces n solutions, l'utilisateur a le choix d'effectuer une sauvegarde sous forme d'un fichier XML issu de *PMDO_Datafile.xml*. Comportant la solution obtenue et les entrées y ayant conduit, ces fichiers garantissent la traçabilité du processus de design conceptuel. Ils pourront servir à reprendre une exécution semblable (le fichier XML est alors chargé lors du pré-traitement) ou bien, à moyen terme (des développements sont en cours), à générer une géométrie Catia V5² (logiciel de conception assistée par ordinateur utilisé chez Pratt & Whitney Canada) servant de base pour les étapes de conception mécanique ultérieures.

²Voir <http://www.3ds.com/products-solutions/brands/CATIA> pour une description du logiciel.

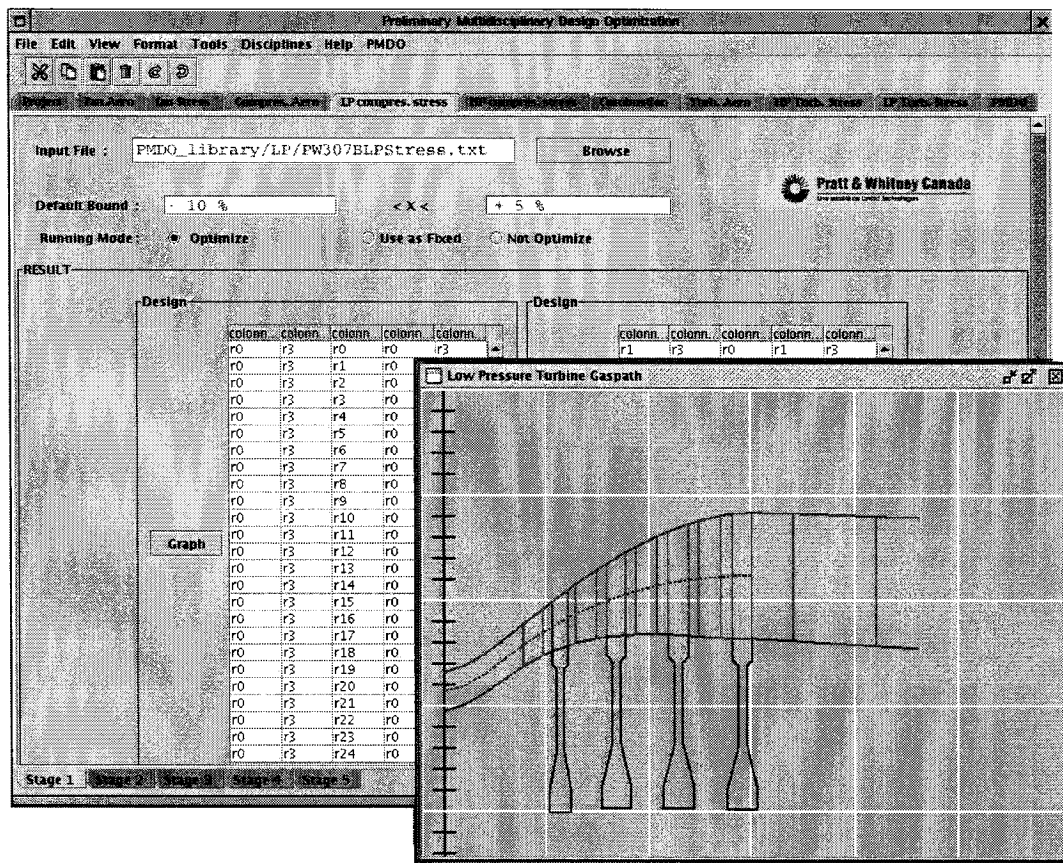


FIGURE 4.4: Exemple d'interface PMDO_GUI lors de la phase de post-traitement

4.2.4 Le composant d'optimisation

4.2.4.1 Le rôle de iSIGHT dans PMDO

Si le Java Data Manager est l'exécutant qui doit gérer le partage des données ainsi que les analyses disciplinaires, le composant d'optimisation est alors l'entité logicielle chargée de commander et coordonner l'exécution de PMDO.

Programmé pour constituer le composant d'optimisation, iSIGHT est utilisé pour les services suivants :

- Les optimiseurs : le logiciel comporte une variété d'algorithmes d'optimisation (appartenant à chacune des trois classes décrites dans le chapitre 3) applicables à la MDO. C'est parmi eux que sont choisis ceux utilisés pour PMDO.
- Un « advisor » : iSIGHT comporte également un algorithme capable de bâtir un plan d'optimisation adapté (dans une certaine mesure) à chaque problème de MDO. Ce conseiller se base pour cela sur certaines caractéristiques du système intégré (nombre de variables de design, nombre de variables discrètes, etc.)
- Un moteur de DOE : un module de iSIGHT peut orchestrer l'exécution automatique d'un plan d'expérience.
- Une base de données interne : offrant plusieurs options de configuration, cet outil permet de garder trace de la progression des optimiseurs dans l'ensemble de design ainsi que d'interrompre, sauvegarder puis reprendre toute exécution.
- Une interface : enfin, iSIGHT comporte une série d'interfaces graphiques relatives aux paramètres de design, aux plans d'optimisation ou bien à la progression de l'optimisation en cours d'exécution.

Cependant, et tel qu'expliqué précédemment, l'intégration des disciplines entre elles ou avec les optimiseurs n'est pas effectuée par iSIGHT. En effet, bien qu'offertes

par le logiciel de Engineous Software, ces fonctions ne sont pas utilisées car elles se sont avérées peu adaptées ou complexes. De plus, la dissociation relative des tâches d'intégration et d'optimisation est renforcée par la volonté de garder une certaine indépendance de PMDO vis-à-vis d'un logiciel commercial.

Des routines « client », programmées en utilisant le langage « Tool Command Language » (TCL), ont été intégrées au composant d'optimisation. Leur exécution par iSIGHT permet un échange de messages TCP/IP entre le Java Data Manager et le composant d'optimisation. C'est par l'intermédiaire de ces clients, et du JDM uniquement, que les tâches d'optimisation ou d'analyse échangent des données. Il n'y a aucun « mapping » direct entre les tâches de iSIGHT.

4.2.4.2 Structure du composant d'optimisation

Le composant d'optimisation est structuré en tâches et sous-tâches (*Tasks* et *Sub-tasks* dans la terminologie iSIGHT) indépendantes associées chacune à une fonction à remplir pour un sous-système donné. Les quatre fonctions utilisées pour PMDO sont l'analyse disciplinaire, le DOE, l'optimisation ou l'exécution d'une sous-tâche.

Le nombre des sous-tâches du composant d'optimisation dépend du MDO traité et donc des disciplines incluses dans le système. Dans tous les cas de figure, une tâche de haut niveau hiérarchique, le Coordinateur (*Coordinator*), est présente. Sa fonction première est de coordonner l'exécution des sous-tâches dans le but d'atteindre la solution du problème formulé. Une sous-tâche *Discipline*i** est également nécessaire pour chacune des analyses disciplinaires *i*, ou disciplines, incluse au système défini dans le PMDO_GUI (les disciplines incluses sont celles dont le *Running_Mode* est fixé à *Optimize* ou bien à *Use_As_Fixed*). La figure 4.5 illustre le cas d'un MDO à trois disciplines comportant donc trois sous-tâches.

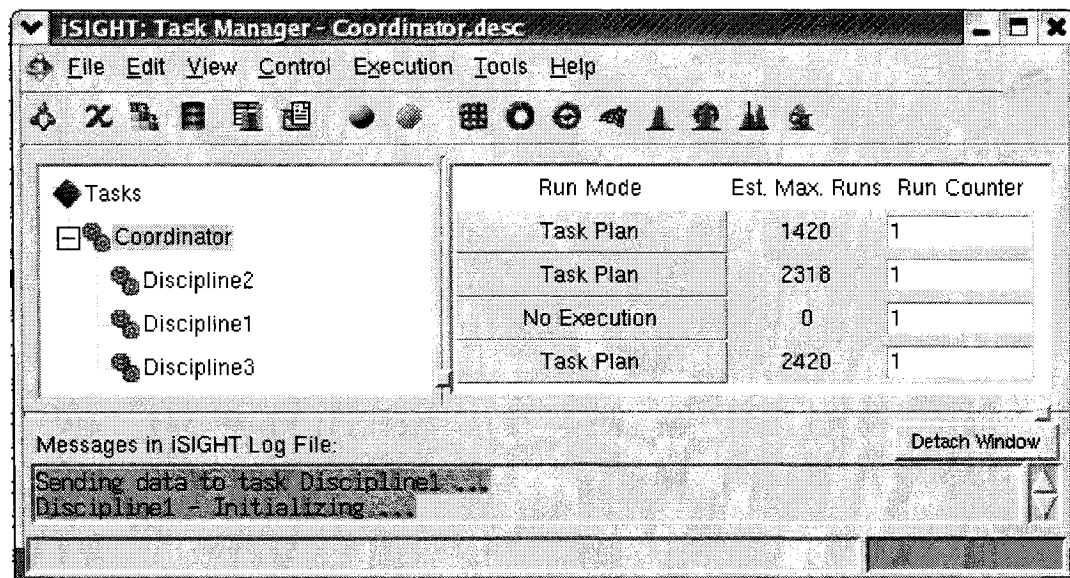


FIGURE 4.5: Exemple de l'interface iSIGHT-PMDO pour un MDO à trois disciplines

La séquence d'exécution des sous-tâches (exécutions séquentielles ou en parallèle) ainsi que leur fonction (aussi appelée *Evaluation Mode* dans la syntaxe iSIGHT) dépendent également de la méthode de décomposition utilisée (voir le chapitre 5 pour les détails relatifs à la configuration des sous-tâches en fonction de la méthode de décomposition).

4.2.4.3 Programmation de iSIGHT

Très complexe, la description exhaustive des principes de programmation et d'utilisation de iSIGHT ne se justifie pas dans le présent document. La section suivante comporte néanmoins une brève présentation des mécanismes de fonctionnement de ce logiciel commercial afin d'illustrer par la suite le fonctionnement du composant d'optimisation de PMDO.

L'utilisation de iSIGHT nécessite la production de *description file* écrits en MDOL. Le Multi Disciplinary Optimization Language MDOL est un langage propriétaire développé par Engineous Software et qui, associé à des routines TCL, sert à constituer ces fichiers textes (reconnaissables à l'extension **.desc*) pour décrire complètement une tâche (*Task*) que iSIGHT devra ensuite exécuter. Ces fichiers sont de deux types : statiques ou dynamiques.

Formés de plusieurs blocs d'instructions MDOL et TCL, les fichiers *description file* statiques contiennent les instructions complètes pour la résolution d'un problème unique (ils ne peuvent donc pas être réutilisés pour un cas de figure différent). Dans le cas d'une optimisation, les *description file* statiques décrivent une tâche d'analyse (assimilable à une boîte noire sur laquelle est basée une fonction coût) et des méthodes d'optimisation à utiliser. La tâche faisant l'objet de l'optimisation est définie par les blocs d'informations suivants :

- Une liste des variables - Noms, types (réel, entier, etc.) et valeurs initiales pour les variables d'entrée et de sortie.
- Une description de blocs de calculs - Les blocs de calcul *Calculation Blocks* servent à décrire, sous forme de routines, des opérations arithmétiques devant être effectuées dans iSIGHT (sans interaction avec d'autres programmes) au cours de l'exécution de la tâche. Dans certains cas, les *Calculation Blocks* peuvent suffire à constituer l'analyse nécessaire à l'évaluation de la fonction coût.
- Des *Procedures* - Les *Procedures* sont des routines TCL exécutées par iSIGHT lorsqu'il est nécessaire d'interagir avec d'autres programmes (ces routines peuvent, par exemple, commander l'exécution d'un programme présent sur le réseau).

De même, l'information relative à la résolution est regroupée pour constituer les divers blocs ci-dessous :

- La liste des variables potentielles;
- La description des objectifs - La fonction coût est identifiée en précisant la (ou les) variable(s) de sortie devant être maximisée(s) ou minimisée(s);
- L'énoncé des contraintes - Une série d'égalités ou d'inégalités associées à certaines des variables de sortie du problème traité;
- La description du plan d'optimisation à utiliser.

Les fichiers *description file* dynamiques comportent, en plus des blocs ci-dessus, des routines TCL qui ne sont exécutées que lors du pré-traitement de iSIGHT. Ces codes (les *Perform Block* par exemple) servent à générer des *description file* statiques adaptés à une configuration donnée. Ainsi, contrairement aux fichiers statiques, les *description file* programmés en MDOL dynamique sont flexibles et peuvent s'adapter à plusieurs cas de figures similaires. Cependant, ils doivent être établis manuellement, à l'aide d'un logiciel éditeur de texte, car ils ne peuvent pas être générés par iSIGHT. Lorsque chargé dans iSIGHT, un *description file* dynamique est interprété (il s'agit alors d'une phase de pré-traitement du composant d'optimisation) afin de générer un fichier statique utilisable directement.

Dans le cas du prototype PMDO, un fichier MDOL dynamique a été établi pour chacune des tâches et sous-tâches : les fichiers *Coordinator.desc* ainsi que *Disciplinei.desc* peuvent donc s'adapter automatiquement à tous les cas d'utilisation, toutes les configurations moteurs ainsi qu'à plusieurs méthodes de décomposition. Ces *description file* utilisent une procédure TCL faisant office de client capable de dialoguer avec le Java Data Manager. C'est au moyen de cette procédure que sont échangés les messages entre le JDM et le composant d'optimisation. Les *Perform Block* de *Coordinator.desc* et des *Disciplinei.desc* font appel à la procédure client pour lire le contenu des *GULBlock* et *Variables_Block*. Cette information est alors traitée pour constituer des listes TCL (liste des variables d'entrée, liste des variables de sortie, etc.) correspondant au MDO en cours. Les listes TCL permettent ensuite

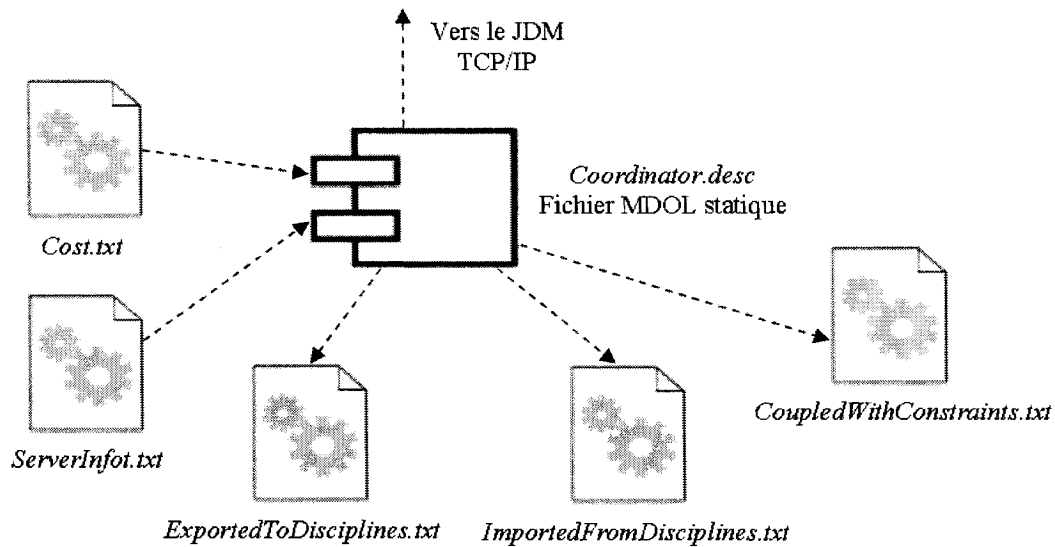
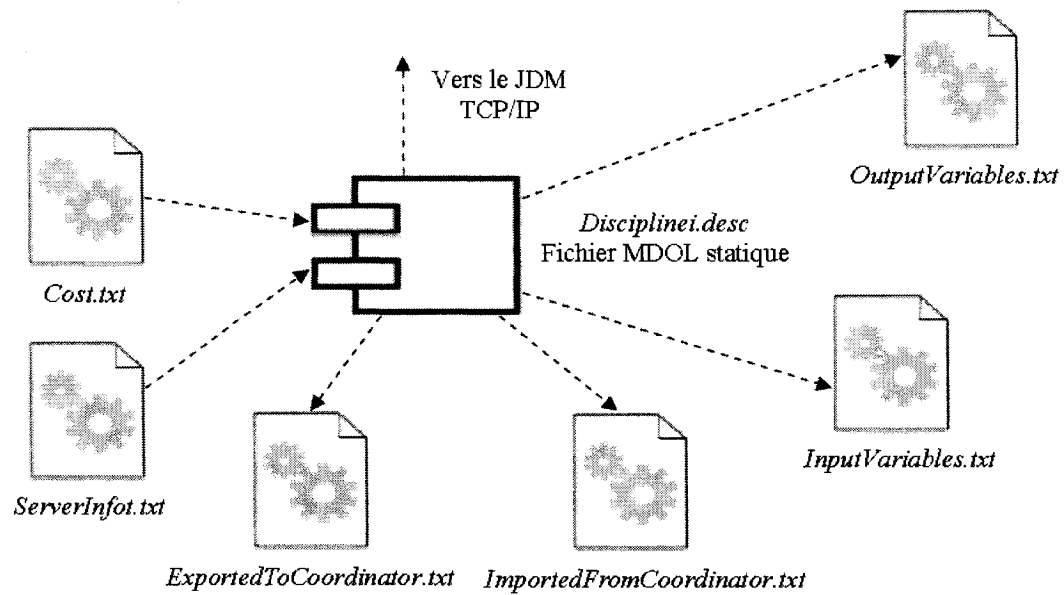


FIGURE 4.6: Diagramme de composants associé au Coordinateur

de constituer automatiquement les différents blocs d'un *description file* statique.

Les fichiers dynamiques *Coordinator.desc* et *Disciplinei.desc* sont donc « instanciés » au cours de l'étape d'initialisation du composant d'optimisation et en tenant compte d'informations obtenues en interrogeant le *Java Data Manager*. Ceci conduit à la création de fichiers **.desc* statiques ainsi que de plusieurs fichiers textes associés et correspondant spécifiquement au cas d'utilisation en cours. Les diagrammes de composants des figures 4.6 et 4.7 représentent l'ensemble des fichiers utilisés puis créés au cours de l'initialisation. Les fichiers **.desc* statiques sont destinés à être exécutés par iSIGHT tandis que ceux portant l'extension **.txt* sont associés spécifiquement aux procédures TCL utilisées lors de l'optimisation (notamment pour exporter /importer des données vers/depuis le *JDM*).

La section 4.3.1.3 fournit une description plus détaillée du pré-traitement et donc du processus permettant de passer d'un *description file* dynamique à sa version statique.

FIGURE 4.7: Diagramme de composants associé à la discipline i

4.3 Séquence d'exécution détaillée

La section ci-après est une présentation plus détaillée de la séquence d'exécution de PMDO. Le diagramme de séquence simplifié de la figure 4.8 permet d'en reconnaître les trois phases principales : le pré-traitement, l'exécution et le post-traitement. Le cas illustré correspond à l'optimisation d'un design de moteur, ce qui signifie que le problème MDO comporte plusieurs disciplines et que l'utilisateur fournit un design initial pour chacune d'entre elles. Le design initial constitué n'est pas nécessairement faisable.

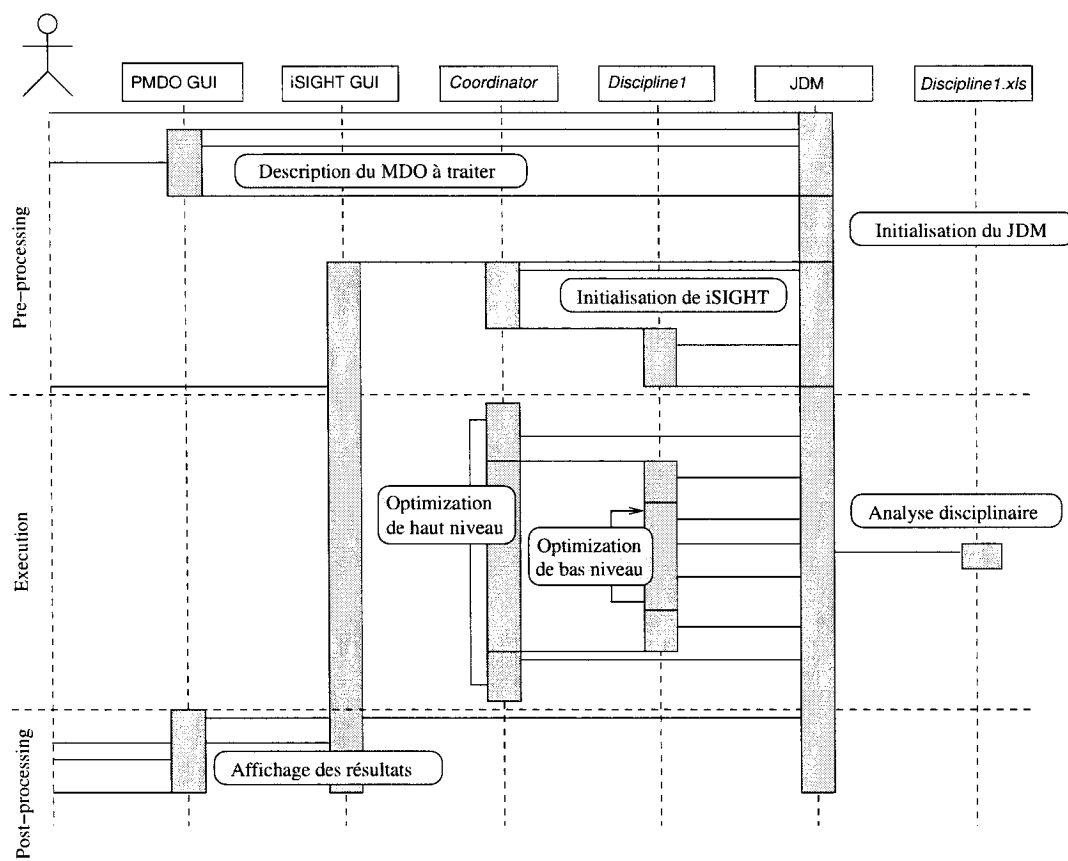


FIGURE 4.8: Diagramme de séquence simplifié

4.3.1 Phase de pré-traitement

4.3.1.1 Remplissage de l'interface graphique PMDO_GUI

La première étape de l'exécution de PMDO est l'ouverture automatique et simultanée (éventuellement sur deux ordinateurs différents) du Java Data Manager et de l'interface PMDO_GUI. Cette dernière se configure au fur et à mesure que l'utilisateur y décrit le problème MDO à résoudre.

Après avoir rempli les pages relatives à une discipline (choix du *Running_Mode* et d'un fichier d'entrée décrivant le design initial) ainsi que celles offrant le choix des options de résolution (telles que la méthode de décomposition à utiliser), l'utilisateur passe à l'étape suivante en cliquant sur un bouton de validation des données entrées. L'interface PMDO_GUI établit alors la communication TCP/IP avec le JDM à qui elle transfère l'intégralité des informations fournies par l'utilisateur.

4.3.1.2 Initialisation du Java Data Manager

Dès réception du contenu du PMDO_GUI, celui-ci est mis en forme pour constituer le *GULBlock* et commencer ainsi un fichier *PMDO_Datafile.xml*.

L'étape suivante est celle de création des *Discipline_Blocks* pour chacune des disciplines incluses. Dans le cas général, ceci doit nécessiter la lecture des fichiers *Properties File* et d'entrées fournis pour chaque discipline (et dont l'adresse est issue du *GULBlock*) afin d'en extraire automatiquement la liste des variables disciplinaires, des attributs associés ainsi que les valeurs initiales. Pour les disciplines représentées par une feuille de calcul Excel et donc n'utilisant pas de fichier texte en entrée, le fichier *Properties File* le remplace, car il comporte des valeurs initiales

pour chacune des variables. Par défaut, la valeur de l'attribut *category* pour toutes les variables des *Discipline_Blocks* est fixée à *local* (les variables sont donc tout d'abord considérées locales).

La constitution du *Variables_Block* est la dernière phase de la création du *PMDO_Datafile.xml* requis pour l'exécution. Tel que précisé dans la section 4.2.2.3, le *Variables_Block* contient la liste exhaustive de toutes les variables du système constitué par l'intégration des disciplines. Ce bloc XML a une structure quasiment identique à celle des *Disciplines_Blocks* puisque seuls sont différents les attributs relatifs aux disciplines utilisant chaque variable. L'algorithme de constitution du *Variables_Block* comporte une phase de tri des variables, puis celle de création des variables copies requises pour la définition des contraintes de couplage.

Le tri se fait en parcourant successivement tous les *Discipline_Blocks*, dans un ordre précis (cette information est contenue dans la liste *hierarchy* du *GUL_Block*), pour y lire chaque variable et déterminer (par comparaison des champs *key*) si elle se trouve déjà dans le *Variables_Block*. Dans le cas négatif, la variable lue dans l'un des *Disciplines_Blocks* est alors ajoutée au *Variables_Block* (les valeurs de tous les attributs sont dupliquées). Dans le cas d'une variable figurant déjà dans le *Variables_Block* (parce que déjà lue dans l'un des *Disciplines_Blocks* hiérarchiquement plus haut placé), elle n'est pas dupliquée mais certains attributs de l'élément correspondant dans le *Variables_Block* sont alors mis à jour (notamment la liste des disciplines utilisant ou produisant la variable considérée). Ce processus permet de déterminer la *category* à laquelle appartient chacune des variables du système : *local* (variables locales), *shared* (variables partagées) ou *coupled* (variables couplées).

Le *Variables_Block* est enfin terminé en dupliquant toutes les variables couplées (c'est-à-dire celles pour lesquelles l'attribut *category* a la valeur *coupled*). Pour chacune d'entre elles, la copie créée prend pour *key* le nom obtenu en concaténant

le préfixe *CopyI* au contenu du champ *key* de la variable couplée d'origine. Par exemple, la variable couplée *T4* sera associée à une nouvelle variable *CopyIT4*. Il faut préciser que la création des variables copies s'accompagne de modifications de certains des attributs de la variables d'origine.

Une fois le *Variables_Block* constitué, le fichier *PMDO_Datafile* est complet et correspond au design initial pour le MDA constitué.

4.3.1.3 Initialisation du composant d'optimisation

Commandée par le JDM, l'initialisation du composant d'optimisation débute par le lancement du logiciel d'optimisation iSIGHT, puis le chargement automatique du fichier *Coordinator.desc*. Le chargement du *Coordinator* commence par l'exécution du *Perform Block* figurant au début du *Coordinator.desc*. Les actions commandées par ce script TCL sont :

- iSIGHT établit une connexion TCP/IP avec le JDM. L'adresse de l'ordinateur sur lequel se trouve le Java Data Manager ainsi que le numéro du port réservé à la communication inter-composants sont lus dans un fichier texte établi par l'administrateur au moment de l'installation de PMDO.
- iSIGHT envoie au JDM une série de requêtes relatives au contenu du *GUL_Block*. En retour, le JDM renvoie des messages de réponse. Ces messages portent par exemple sur la méthode de décomposition à utiliser ou la liste des disciplines incluses pour l'exécution en cours.
- En une boucle *For*, iSIGHT envoie alors une succession des messages afin de recevoir en retour le contenu du *Variables_Block* lu élément par élément.
- Au fur et à mesure de la lecture et pour chacune des variables reçues, iSIGHT effectue une série de tests conditionnels en fonction des valeurs de différents

- attributs. Ces tests permettent, par exemple, de trier les variables en fonction de leur utilisation (variable d'entrée ou de sortie pour une discipline donnée) ou de la *category* à laquelle elles appartiennent. Les tests utilisés, et donc l'affectation des variables, dépendent aussi de la méthode de décomposition choisie. En fonction des résultats de ces tests, chaque variable peut-être placée dans une liste TCL établie pour la tâche en cours de chargement (le *Coordinator* dans le cas présent). Ces listes, pour la plupart, correspondent chacune au contenu de l'un des blocs de *description file* statique tels que présentés à la section 4.2.4.3. Par exemple, sont ainsi constituées la liste des variables d'entrée (*inputs*), des variables potentielles (*Potential variables*) et des variables de sortie (*outputs*). Quelques listes TCL additionnelles (liste des variables exportées vers les autres tâches par exemple) sont aussi constituées au fur et à mesure de la lecture du *Variables_Block*. Dans le cas des variables créées par le JDM pour la résolution du couplage (et facilement reconnaissables à leur attribut *key* commençant par le préfixe *CopyI*), iSIGHT définit un nom absolu (correspondant au champ *key* dans le JDM) et un nom local propre à une discipline (sans le préfixe *CopyI*) (les détails de l'utilisation des noms absolus et locaux seront présentés à la section 4.3.2.2).
- Les listes ainsi constituées n'étant plus accessibles une fois l'initialisation de iSIGHT terminée, certaines d'entre elles sont utilisées pour générer des fichiers texte utilisés au cours de l'exécution ultérieure. C'est le cas des listes des variables exportées vers les autres tâches (citons les fichiers *Exported.txt* et *Imported.txt*)
 - Les listes TCL à peine constituées sont alors utilisées pour générer le contenu de chacun des champs dynamiques du *description file*. Le MDOL dynamique est alors remplacé par un code MDOL statique établi à partir des listes TCL, et donc spécifique au système correspondant au MDO en voie de résolution. Ainsi, des contraintes de couplage sont automatiquement générées en utilisant les variables couplées et celles reconnues par le préfixe *CopyI* dans leur nom (i.e. le couple

des noms locaux et absolus).

À la fin de cette étape, le fichier dynamique d'origine *Coordinator.desc* a été transformé en un *description file* statique adapté à l'exécution à venir.

La même séquence est alors automatiquement effectuée pour chacune des sous-tâches incluses. À leur tour, les tâches *Disciplinei* établissent une connection avec le JDM, lisent le *PMDO_Datafile.xml* pour générer des listes TCL, des fichiers temporaires et enfin des codes MDOL statiques.

Une fois constitués, les *description file* statiques sont fusionnés en un seul fichier qui peut être exécuté directement par iSIGHT. Ce code est adapté au système considéré, aux *Running_Mode* choisis pour chaque discipline ainsi qu'à la méthode de décomposition sélectionnée. L'étape d'initialisation de iSIGHT est alors terminée.

4.3.1.4 Derniers ajustements, modifications optionnelles et lancement de l'exécution

Par l'intermédiaire de l'interface de iSIGHT, l'utilisateur a alors la possibilité de vérifier ou préciser certaines informations.

Une interface présente un tableau regroupant toutes les informations liées aux variables associées à une tâche donnée. L'utilisateur l'exploitera éventuellement pour modifier une valeur initiale, changer les bornes de l'ensemble de design (les valeurs présentes à l'ouverture de iSIGHT sont celles calculées par le JDM en fonction de pourcentages spécifiés par l'utilisateur lorsqu'il remplissait l'interface PMDO_GUI), retirer une variable de la liste de celles accessibles à l'optimiseur, imposer des contraintes sur certaines variables de sortie ou bien encore définir un nouvel objectif d'optimisation.

Les fonctions coût et équations de contraintes peuvent également être éditées, par l'intermédiaire de l'interface de iSIGHT, en fonction de la méthode de décomposition utilisée (voir le chapitre 5.3.3 pour plus de détails).

S'il renonce au plan d'optimisation établi par défaut par iSIGHT, l'utilisateur peut également en spécifier un autre au moyen d'une interface iSIGHT spécifique.

Enfin, des tableaux de suivi ainsi que des graphiques peuvent être bâtis via l'interface dédiée. Ils permettront, en cours d'exécution, de suivre en temps réel la convergence de l'optimisation.

Ces ajustements étant faits, la commande de lancement de l'exécution de iSIGHT signifie la fin de la phase de pré-traitement et le début de l'exécution proprement dite.

4.3.2 Phase d'exécution

4.3.2.1 Séquence détaillée

Commençant dès l'instant où l'utilisateur ordonne l'exécution de la tâche *Coordinator*, l'optimisation est une étape longue, qui requiert de nombreuses analyses et qui est conduite sans la moindre intervention de l'utilisateur.

La séquence d'exécution dépend grandement de la méthode de décomposition choisie par l'utilisateur. L'exemple suivant illustre le cas d'une optimisation conduite à l'aide la méthode dite Distributed Optimizations (DO, se référer au chapitre trois pour des détails). Le *Coordinator* comporte un optimiseur maître (ou optimiseur système) qui gère les variables systèmes, i.e. couplées et partagées, ainsi qu'une fonction coût et des contraintes globales tandis que les optimiseurs locaux, dans

chaque tâche *Disciplinei*, ont la responsabilité des variables locales. La séquence correspondant à une itération du *Coordinator* est l'enchaînement coordonné des 13 étapes suivantes :

1. L'optimiseur maître du *Coordinator* fixe la valeur de toutes les variables systèmes.
2. L'itération précédente est alors testée par appel d'un bloc de calcul appelé *TestLastIteration*. Ce code TCL permet de reconnaître les itérations ayant abouti à un design faisable ou optimal puis, le cas échéant, de commander au JDM la mise à jour de fichiers XML de résultats. En effet, si le design obtenu est faisable, une commande TCP/IP est envoyée au JDM pour lui ordonner de dupliquer le fichier *PMDO_Datafile.xml* courant, de le renommer, puis de le déplacer dans un dossier des designs faisables. De plus, si le design est faisable et meilleur que tous les précédents, le *Coordinator* envoie une instruction pour mettre à jour le fichier *Optimum.xml* en y copiant le contenu de *PMDO_Datafile.xml*.
3. Une procédure TCL est exécutée pour exporter ces valeurs vers le JDM qui met à jour les champs correspondants dans le *Variables_Block*.

Les actions suivantes sont effectuées simultanément pour toutes les disciplines incluses :

4. Les tâches *Disciplinei* exécutent une procédure TCL pour importer, depuis le JDM, toutes les valeurs des variables systèmes qui sont requises pour l'analyse disciplinaire (la sous-tâche est donc mise à jour avec l'information exportée par le *Coordinator*).

Commencent alors les optimisations locales.

5. L'optimiseur local fixe les valeurs de toutes les variables locales.

6. Une procédure TCL est exécutée pour exporter au JDM les valeurs de toutes les variables disciplinaires. Le JDM met à jour les champs correspondants dans le *Discipline_Blocks*.
7. iSIGHT envoie au JDM l'ordre d'exécuter l'outil d'analyse disciplinaire en utilisant les valeurs de variables tout juste mises à jour.
8. Le JDM exécute l'analyse en utilisant les valeurs de variables lues dans le *Discipline_Blocks*. Les valeurs de variables de sortie sont alors mises à jour dans le *Discipline_Blocks*.
9. Une autre routine TCL est utilisée pour importer les valeurs de variables de sortie depuis le *Discipline_Blocks*. Une fois de plus, ceci requiert l'établissement d'une communication TCP/IP inter-composants. Se référer à la figure 4.9 pour une illustration graphique du flux de données entre les composants PMDO (les numéros entre crochets correspondent aux étapes de cette énumération).
10. La fonction coût de la tâche i est calculée et l'optimiseur local évalue si les contraintes sont satisfaites.

Ces 6 dernières étapes sont répétées jusqu'à convergence de l'optimisation locale (ou, en cas de non convergence, jusqu'à ce que le nombre maximum d'itérations permises soit atteint).

11. Dernière action avant la fin de l'exécution de la sous-tâche, la procédure TCL d'exportation est appelée pour mettre à jour le *Variables_Block* avec les résultats de l'optimisation locale.

Une fois toutes les optimisations locales terminées, le *Coordinator* récupère, via le JDM, les résultats consignés dans le *PMDO_Datafile.xml*.

12. Le *Coordinator* exécute la procédure TCL d'importation afin de lire les valeurs du *Variables_Block*.

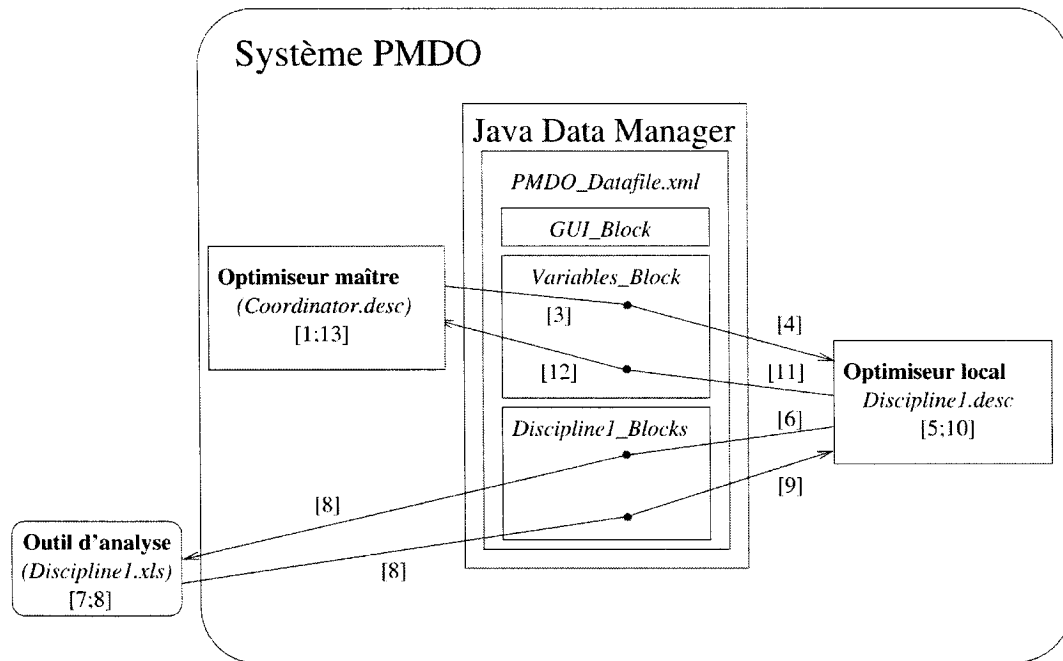


FIGURE 4.9: Flux de données au cours de la phase d'exécution (décomposition DO)

13. Le *Coordinator* évalue les contraintes et la fonction coût du système en exécutant les *Calculation Blocks CouplingConstraints* et *CostFunction*.

Selon le problème décrit par l'utilisateur, les étapes 1 à 13 sont répétées jusqu'à ce que n designs aient été trouvés ou bien jusqu'à ce que l'optimiseur maître ait convergé à un optimum.

4.3.2.2 Mécanismes de partage de données entre les tâches

Tous les transferts de données entre les composants de PMDO se font par appel à deux routines TCL associées à des fichiers texte *Import.txt* et *Export.txt* créés au cours de la phase de pré-traitement. Afin de garantir un meilleur entretien du code, les procédures *Import* et *Export* sont rigoureusement identiques dans tous les

description file de PMDO. Les fichiers textes temporaires utilisés sont des listes de paires de noms de variables : le nom absolu, qui correspond au champ *key* du *Variables_Block*, et le nom local, utilisé par iSIGHT pour identifier chaque variable. Le nom local diffère du nom absolu dans le cas des variables couplées et dupliquées, reconnues automatiquement par le préfixe *CopyI*.

Lors de l'exécution de la procédure *Import*, une connection client-serveur est établie entre iSIGHT et le JDM, puis le fichier texte temporaire *Import.txt* est ouvert et lu ligne par ligne. Pour chaque paire lue, par exemple *CopyIT4-T4*, le nom absolu (*CopyIT4*) est concaténé au mot *GetVariableValue* pour formuler une requête adressée au JDM par l'une des tâches du composant d'optimisation. En réponse, le JDM renvoie un message contenant la valeur numérique demandée (telle que lue dans le champ *CopyIT4.value* du *Variables_Block*), puis iSIGHT l'utilise pour mettre à jour la valeur de la variable locale *T4*. En conséquence, ce mécanisme de traduction des noms de variables permet de masquer aux tâches de bas niveau *Disciplinei* l'existence de la variable couplée *CopyIT4*. Illustré par la figure 4.10, ce mode de fonctionnement constitue l'une des bases de la méthode que nous avons mis en place pour gérer automatiquement le couplage interdisciplinaire. La procédure *Export* fonctionne de façon semblable pour le transfert des résultats depuis le composant d'optimisation vers le JDM. Moyennant le remplacement d'un mot clef du message envoyé, les mêmes procédures sont utilisées pour lire et écrire dans les *Discipline_Blocks* et le *Variables_Block*.

Le *Variables_Block* est utilisé pour le transfert de données entre tâches (*Coordinator* et *Disciplinei*, voir la figure 4.9). Il est mis à jour deux fois à chaque itération de l'optimiseur maître, pour la publication des variables systèmes, puis des résultats des optimisations locales.

Les *Discipline_Blocks*, par contre, servent pour le transfert de données en vue de

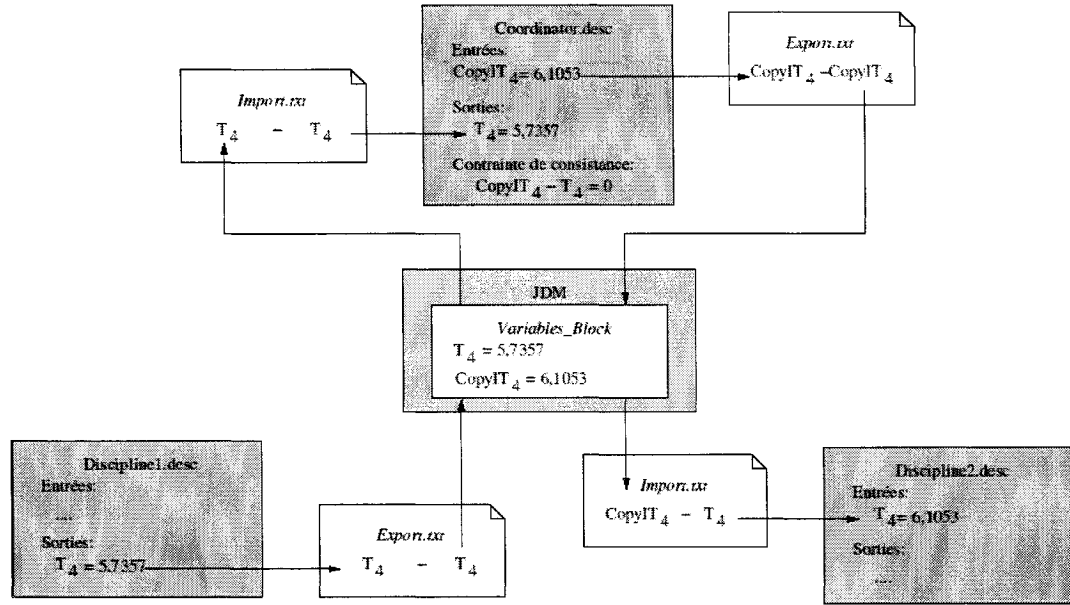


FIGURE 4.10: Mécanismes de gestion automatisée du couplage interdisciplinaire par l'emploi de variables globales dupliquées et de contraintes de consistance

l'exécution d'une analyse disciplinaire (voir la figure 4.9). Ils sont mis à jour deux fois à chaque itération de l'optimiseur local, pour la publication des valeurs de variables d'entrée, puis de celles des variables de sortie.

Enfin, des mécanismes de communication inter-tâches propres à iSIGHT sont utilisés pour la synchronisation des exécutions ainsi que pour faire « remonter » au *Coordinator* les statuts (résultats) d'exécution des tâches disciplinaires.

4.3.3 Phase de post-traitement

En fin d'exécution et en cas de succès, l'interface PMDO_GUI redevient automatiquement active pour la présentation des résultats (voir la capture d'écran de la figure 4.4). L'interface permet d'éditer des commentaires relatifs aux designs obte-

nus, puis de sauvegarder le fichier XML correspondant. L'information qu'il contient peut, par la suite, servir de départ pour une autre optimisation.

CHAPITRE 5

MÉCANISMES DE CONFIGURATION AUTOMATISÉE EN FONCTION DE LA DÉCOMPOSITION

L'objectif visé lors de la conception du prototype a été de produire un outil qui soit flexible, entre autres, afin de s'adapter à la méthode de décomposition utilisée pour la résolution du problème d'optimisation multidisciplinaire. Il est donc nécessaire d'automatiser, pendant le pré-traitement, toutes les opérations nécessaires pour passer de la formulation ingénieur à celle décomposée. L'objectif est de limiter au maximum l'intervention de l'utilisateur dans ce processus.

La conception d'un processus de configuration automatisée et son déploiement par le prototype constituent une contribution importante de ce travail de maîtrise. Dans ce chapitre sont présentés les défis posés par la configuration automatique et les mécanismes utilisés pour satisfaire ces exigences. Également, s'ajoutant au cas DO utilisé en exemple dans la section 4.3.2.1, une présentation détaillée de la phase d'exécution pour FIO et DAO permet de juger les résultats de la configuration automatisée.

5.1 Les défis de la configuration automatique

Dans sa formulation naturelle, c'est-à-dire telle qu'elle apparaît aux ingénieurs, le problème d'optimisation est décrit par les éléments suivants (voir le Chapitre 3) :

- un système couplé (MDA) de n analyses, décrit par la liste des disciplines incluses ainsi que des *Running_Mode* et design initial associés à chacune,

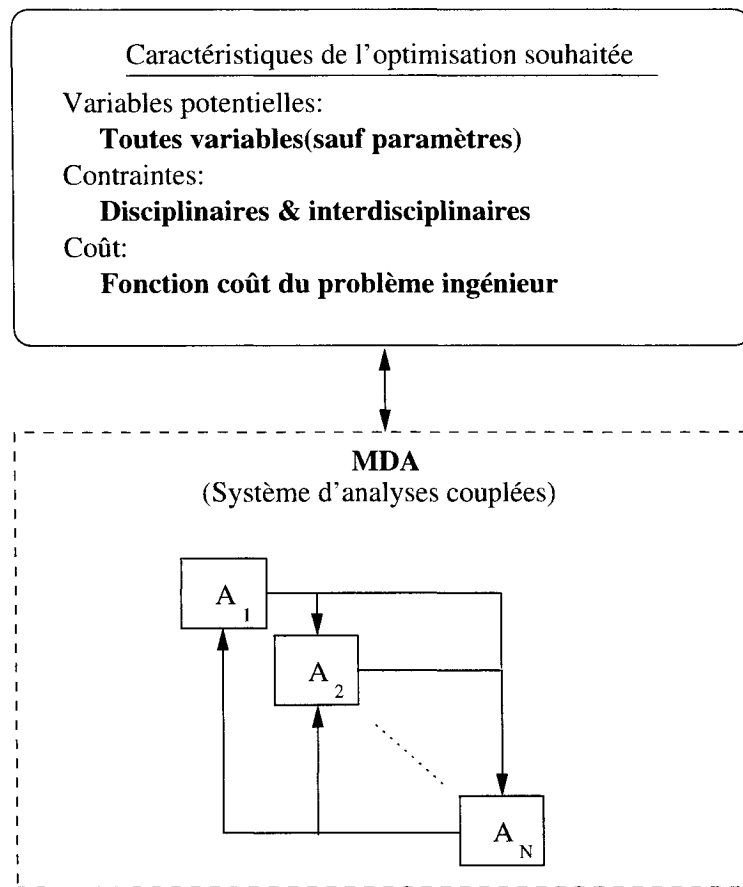


FIGURE 5.1: MDO originel

- une liste de variables potentielles et de leurs bornes respectives définissant l'ensemble de design,
- un ensemble de contraintes disciplinaires et interdisciplinaires,
- une fonction coût associée à un objectif d'optimisation.

Rassemblés sous forme schématique à la figure 5.1, ils constituent la formulation originelle, dite aussi « ingénieur », du besoin de l'utilisateur de PMDO.

La transformation du problème ingénieur en une forme décomposée équivalente est un problème non trivial. En effet, les opérations requises sont nombreuses et

complexes :

1. Classification des variables : les variables de design du MDA doivent être triées pour identifier celles partagées, couplées ou locales à une discipline.
2. Création de variables additionnelles : dans certains cas, des variables additionnelles t doivent être ajoutées.
3. Répartition des variables : dans le cas des décompositions multi-niveaux, les variables de design doivent être réparties entre l'optimiseur maître et les n optimiseurs disciplinaires.
4. Assignation des tâches pour chaque discipline : selon la méthode de décomposition, la définition de n optimiseurs disciplinaires peut-être requise. Dans le cas contraire, une tâche d'analyse correspond à chaque discipline.
5. Définition de la fonction coût globale : la fonction coût de l'optimiseur maître doit être précisée car elle n'est pas nécessairement identique à celle de la formulation originelle. Certaines méthodes de décomposition requièrent l'ajout de termes de pénalité.
6. Répartition des contraintes : les contraintes du problème originel doivent être réparties entre les optimiseurs mis en place. Dans certains cas, des contraintes de consistance additionnelles doivent s'ajouter.
7. Mise en place d'une méthode de résolution du couplage : pour les décompositions n'utilisant pas de contraintes de couplage ou de pénalité, une autre méthode de résolution du couplage interdisciplinaire doit être prévue.

Devant être capable de décomposer et résoudre le problème ingénieur selon différentes méthodes (voir l'inventaire des requis au chapitre 2), PMDO doit donc pouvoir se configurer automatiquement. Cette configuration implique toutes les opérations citées ci-dessus. Elle doit se faire au cours du pré-processing et avec un minimum d'interventions de la part d'un utilisateur non spécialiste des méthodes de

décomposition. Les développements ci-après illustrent les mécanismes développés et testés pour permettre une configuration automatisée en fonction des informations recueillies dans l'interface PMDO_GUI.

5.2 Choix de conception et principes de programmation

Le détail des sept opérations de transformation montre que seule la première est inchangée quelle que soit la décomposition utilisée. Bien que non requise, en théorie, pour appliquer FIO, il a été choisi de traiter la seconde de la même façon. Ainsi, les opérations de classification et de création de duplicata des variables couplées, toutes deux effectuées lors de l'initialisation du JDM, deviennent indépendantes de la décomposition. Ce choix permet de regrouper dans le composant d'optimisation toute la variabilité associée à la configuration automatique. L'exécution des JDM et de l'interface graphique PMDO_GUI sont inchangées quelle que soit la décomposition utilisée. Ce choix permet de faciliter l'entretien et les évolutions ultérieures du logiciel.

De plus, définissant l'exécution du composant d'optimisation, des fichiers *description file* dynamiques s'adaptant à tous les cas de figure ont été préférés à une bibliothèque de codes consacrés chacun à une méthode uniquement. Ainsi, les deux fichiers *Coordinator.desc* et *Disciplinei.desc* doivent être configurables en fonction de chacune des méthodes proposées. Cette flexibilité a été obtenue en faisant les trois choix suivants :

- Les *description file* incluent des codes dynamiques (MDOL et TCL) pour générer le contenu des différents blocs du fichier statique issu de l'initialisation de iSIGHT (se reporter à la section 4.2.4.3 pour une description des blocs utilisés pour programmer iSIGHT). Ces codes dynamiques sont exécutés lors de l'initia-

lisation du composant d’optimisation et comportent des tests de type *IF* et des instructions propres à une décomposition. En fin d’initialisation, le *description file* produit est statique et propre à une décomposition uniquement.

- Les instructions d’exécution sont fractionnées en plusieurs opérations élémentaires placées dans des *Calculation Blocks* indépendants. Certains sont propres à une forme de décomposition et inclus au besoin. Cela permet de décrire de façon condensée (par une succession d’exécution de *Calculation Blocks*) les étapes d’exécution pour chaque décomposition.
- Les routines TCL utilisées par iSIGHT (autrement dit les *Procedures* servant à l’interaction avec le JDM) sont génériques (leur exécution est conditionnée par l’emploi de nombreux arguments), identiques dans tous les fichiers *Coordinator.desc* et *Disciplinei.desc*, et leur exécution n’est aucunement reliée à la décomposition choisie.

Ces principes de programmation ont permis de construire un processus de configuration automatisée de PMDO s’articulant en trois étapes, tel que décrit ci-dessous.

5.3 Les trois étapes de la configuration automatisée

Cette section reprend la description de certaines phases du pré-traitement, déjà présenté en détails dans le chapitre 4, en mettant l’accent sur les étapes du processus de configuration automatisée. La figure 5.2 en représente schématiquement la progression.

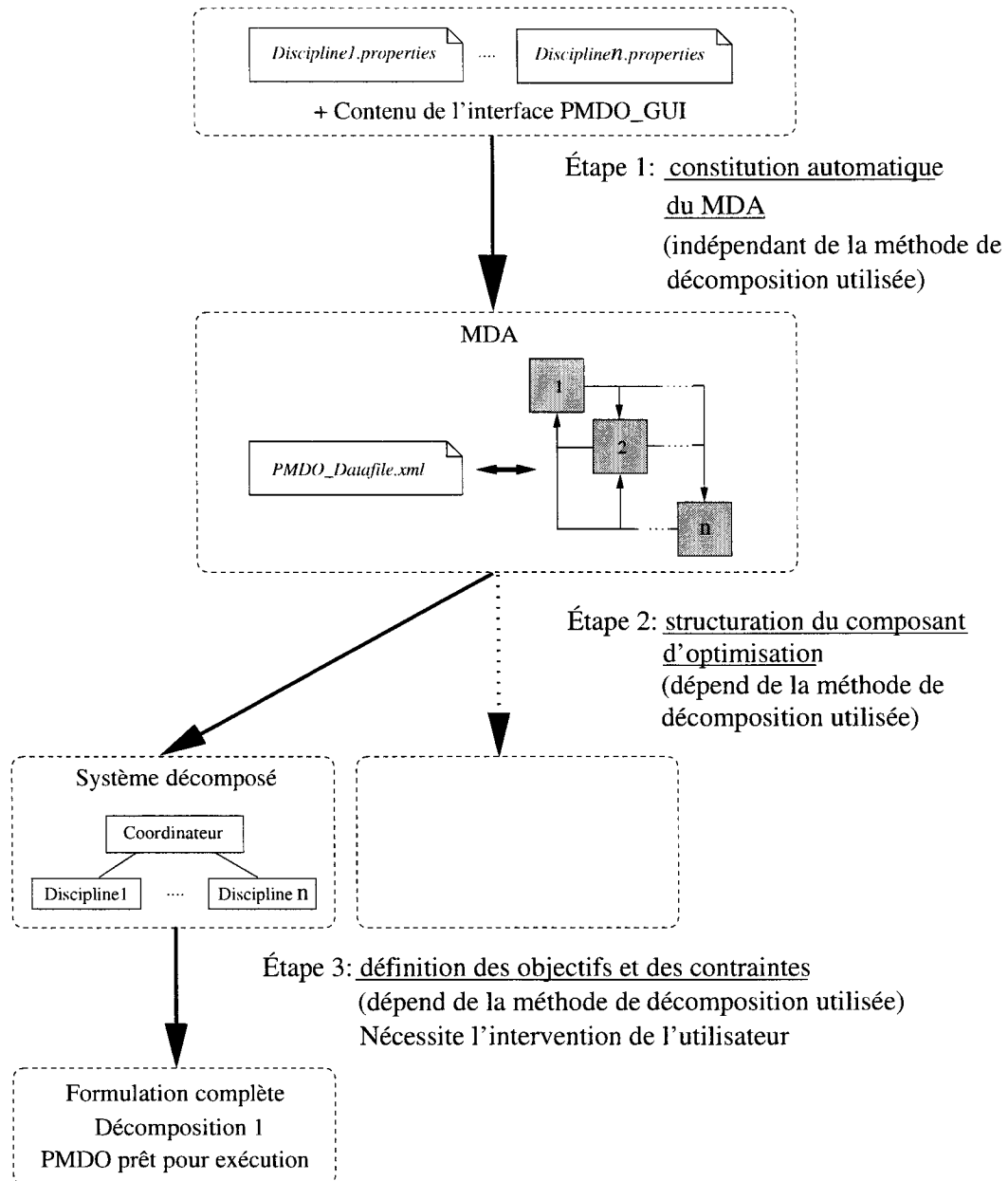


FIGURE 5.2: Étapes du processus de configuration automatisée

5.3.1 Constitution automatique du MDA

Cette première étape est effectuée pendant l'initialisation du JDM. Elle correspond aux opérations (1) et (2) de la section 5.1 et utilise l'information recueillie dans le PMDO_GUI (liste du *Running_Mode* et lien vers un fichier de design initial) ainsi que celle extraite des *Properties_File* (qui dénombrent et décrivent les variables d'entrée et de sortie requises pour chaque discipline, indépendamment de son inclusion dans un MDA) pour constituer le *Variables_Block*, associé au MDA originel, selon l'algorithme suivant :

- Pré-remplissage du *Variables_Block* par lecture successive de tous les *Discipline_Block* et comparaison, à chaque fois, de l'attribut *Key* identifiant chaque variable. Le rôle (*input*, *output* ou *#*) est identifié à chaque fois et pour chaque discipline *i* par un attribut *disciplinei*.
- Détermination de la catégorie (*local*, *shared* ou *coupled*) de chaque variable du *Variables_Block* en fonction des champs *disciplinei* pour toutes les disciplines incluses.
- Création d'une variable additionnelle pour chaque variable couplée. L'identifiant *Key* de chaque variable dupliquée est obtenu à partir de celui d'origine en y concaténant le préfixe *CopyI*. Les champs *disciplinei* et *category* de ces deux variables sont à chaque fois modifiés.
- Imposition de blocages (au moyen de l'attribut *IsParameter*) pour empêcher que les variables intervenant dans une discipline fixée (*Running_Mode=UseAsFixed*) ne soient ultérieurement modifiées par un optimiseur.

5.3.2 Structuration du composant d'optimisation

Seconde étape de la configuration automatisée, la structuration du composant d'optimisation correspond à l'initialisation de iSIGHT et à la réalisation des opérations (3), (4) et (7) (voir la section 5.1). Elle s'effectue en cinq temps :

- Lecture, via le réseau, du *Variables_Block* pour constituer ensuite automatiquement le contenu des blocs des *Coordinator.desc* et *Disciplinesi.desc*. C'est alors que sont assemblées, par exemple, la liste des variables potentielles pour chaque optimiseur de iSIGHT.
- Éventuelle imposition automatique de contraintes de consistance à chaque fois que le préfixe *CopyI* est reconnu dans l'identifiant *Key* d'une variable.
- Éventuelle inclusion, pour le *Coordinator.desc* uniquement, d'une méthode itérative de résolution du couplage (décrite en détails dans la section 5.4.2 ci-après). Cette méthode est alors automatiquement affectée à la détermination des variables dont l'attribut *category* a la valeur *coupled*. Lesdites variables ne font alors pas partie de la liste de celles accessibles aux optimiseurs.
- Pour chaque *description file* inclus, l'un des trois *Evaluation Mode* suivants (nous utilisons ici la terminologie propre à iSIGHT) est déterminé en fonction de la méthode de décomposition :
 - *taskplan* - Un optimiseur est utilisé.
 - *single* - L'analyse correspondante n'est exécutée qu'une seule fois à chaque appel de la sous-tâche.
 - *donotrun* - Bien que prise en compte au cours du pré-traitement, la sous-tâche est ignorée et l'analyse correspondante n'est jamais exécutée.
- Les fichiers texte créés automatiquement en fin d'initialisation de iSIGHT et utilisés lors des transferts de données entre le JDM et le composant d'optimisation (appel notamment aux procédures *Export* et *Import*) font le lien entre le nom

global et le nom local de chaque variable.

5.3.3 Définition des objectifs et contraintes

L'ultime phase de la configuration automatisée demande l'intervention de l'utilisateur. En effet, la formulation du problème se termine par la spécification des fonction coût et contraintes, soit les opérations (5) et (6). Ceci est effectué simplement, pour chaque optimiseur et par l'intermédiaire d'interfaces graphiques spécialisées de iSIGHT. Dans le cas de la fonction coût, l'utilisateur définit librement une expression mathématique évaluée lors de l'appel d'un *Calculation Block* intitulé *CostFunction* (voir par exemple la figure 5.3 montrant les interfaces d'édition de la fonction coût).

Dans le cas des décompositions FIO et DAO n'utilisant qu'un seul optimiseur, cette opération n'est qu'une simple transcription des contraintes et fonctions coût du problème ingénieur. La création des contraintes de couplage est entièrement automatique lors de la structuration du composant d'optimisation. En conclusion, la configuration automatique pour FIO et DAO ne requiert aucune connaissance des méthodes de décomposition de la part des utilisateurs de PMDO.

Dans le cas de la décomposition DO, par contre, la fonction coût originelle du problème ingénieur doit être fractionnée en n fonctions coût locales avant d'être répartie entre $n + 1$ (n optimiseurs disciplinaires et 1 optimiseur maître) tâches d'optimisation. De même, les contraintes locales doivent être imposées aux optimiseurs disciplinaires tandis que les contraintes interdisciplinaires devront être précisées dans le *Calculation Blocks* nommé *EngineeringConstraints* au niveau du Coordinateur. Bien que ne posant aucune difficulté particulière, cette opération nécessite tout de même que l'utilisateur de PMDO connaisse les contraintes liées à

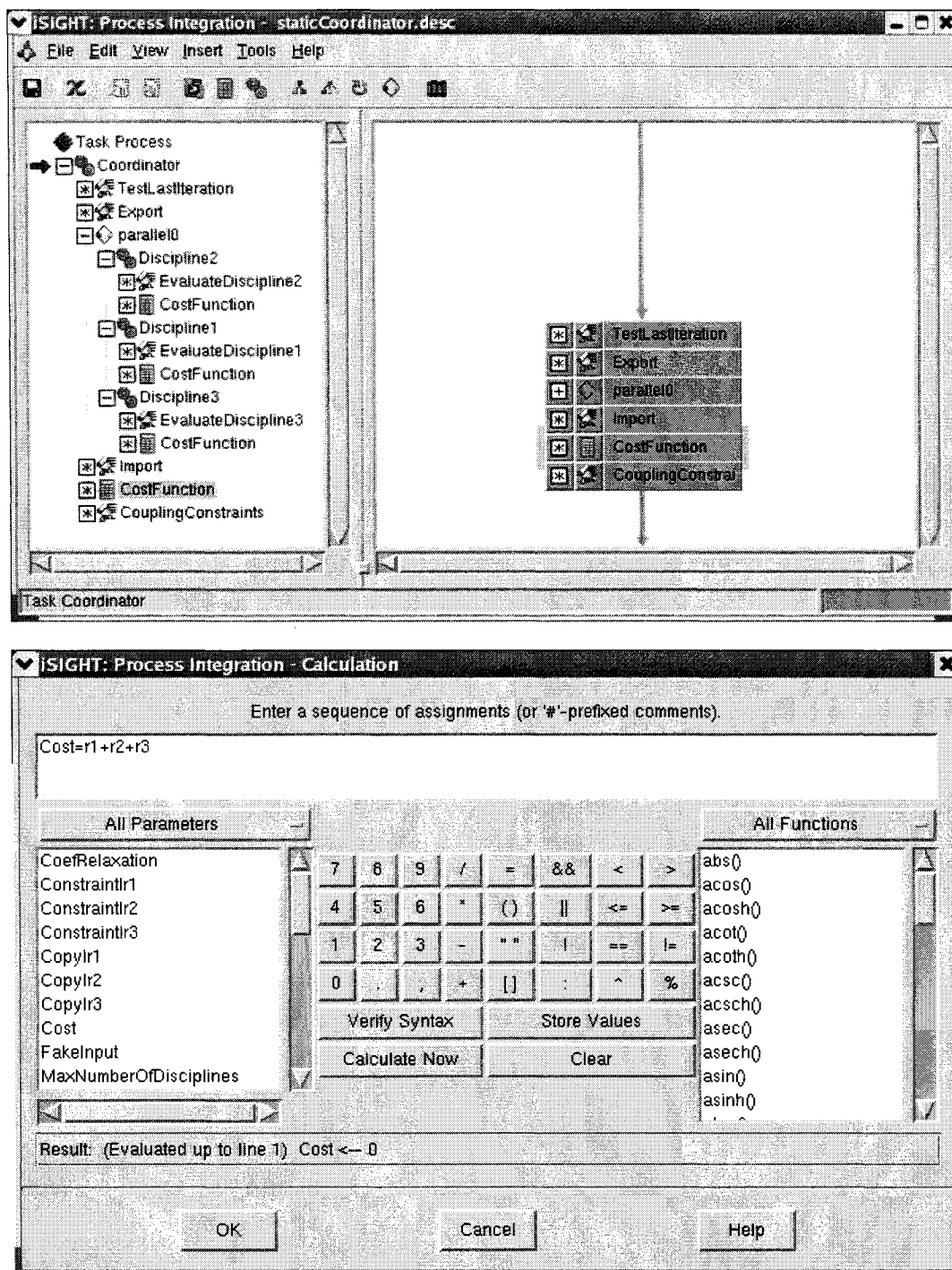


FIGURE 5.3: Interfaces iSIGHT pour l'édition de la fonction coût

la décomposition DO.

C'est l'intervention humaine de cette dernière étape qui permet de dire que si elle est automatisée, **la configuration de PMDO n'est pas entièrement automatique.**

Dans les trois sections suivantes, seront présentées les structures et séquences d'exécution de PMDO pour les trois décompositions mises en place dans le prototype. Ces informations seront à chaque fois résumées sous la forme de diagrammes schématiques (voir figures 5.4, 5.8 et 5.10) et de séquence (voir figures 5.7, 5.9 et 5.11).

5.4 Configuration automatisée selon FIO

5.4.1 Éléments du MDO configuré pour FIO

Dans le cas de la méthode de résolution FIO, le *Coordinator* est associé à une tâche d'optimisation tandis que les sous-tâches *Disciplinei* sont configurées pour des analyses disciplinaires simples ou bien ne sont pas exécutées.

Le *Coordinator* gère directement toutes les variables non fixées du système, exceptées celles qui sont couplées. La valeur de ces dernières est alors calculée de façon itérative par l'exécution de codes MDOL et TCL dynamiques (ces codes sont une mise en application de la méthode du point fixe vectoriel présentée ci-dessous).

La figure 5.4 résume la structure et les composantes du MDO originel décomposé selon FIO.

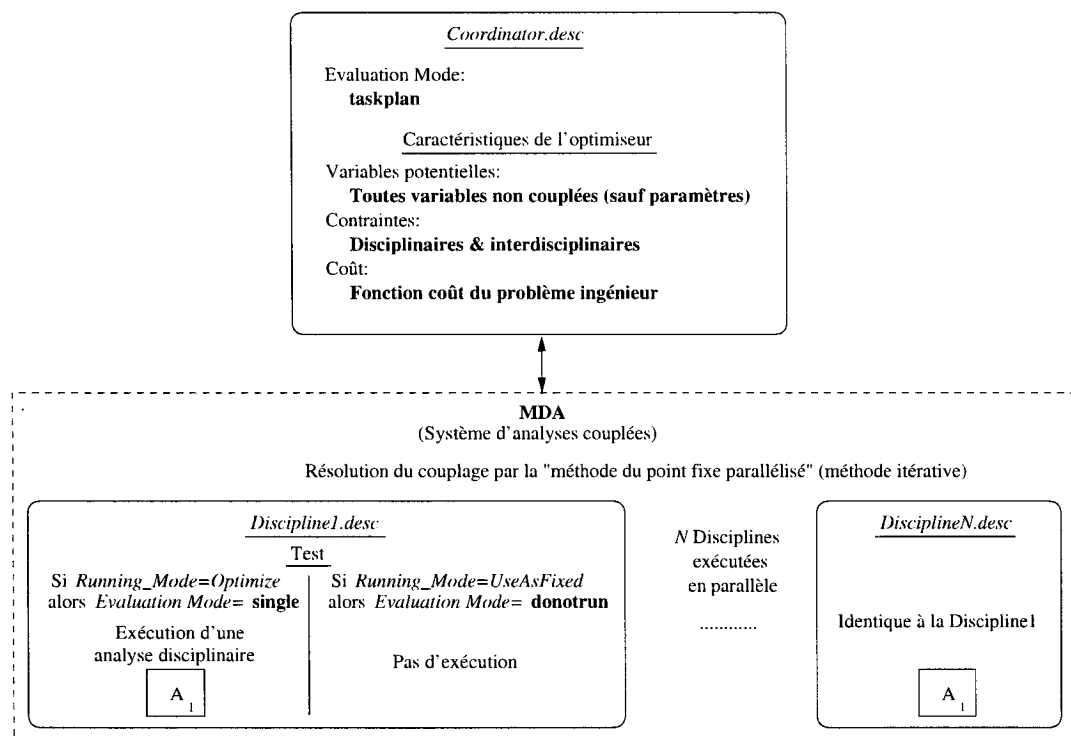


FIGURE 5.4: MDO originel reformulé selon la décomposition FIO

5.4.2 Détails de l'algorithme de résolution du couplage

La méthode du point fixe est un algorithme itératif permettant de déterminer une valeur approchée de l'une des solutions de l'équation à une dimension $x = g(x)$. Son utilisation requière le choix d'une valeur initiale x_0 de x ainsi que d'un seuil de convergence ϵ et d'un nombre maximum d'itérations N_{max} . La convergence de la méthode dépend de caractéristiques mathématiques de la fonction g (notamment de sa continuité et de la norme de ses dérivées en x , voir Fortin (1995) pour les développements mathématiques explicatifs). Il existe des cas de figure (dépendants de g et du choix de x_0) pour lesquels la méthode du point fixe diverge. L'algorithme se termine alors parce que le nombre n d'itérations effectuées atteint la valeur maximale permise N_{max} . La précision de la valeur approchée de la solution dépend du critère de convergence ϵ choisi. La convergence est obtenue lorsque le carré de la différence entre deux valeurs de x , pour deux itérations successives, est inférieur ou égal à ϵ .

Les étapes de l'algorithme du point fixe sont :

- Étape 1 : Poser $n = 0$,
- Étape 2 : Tant que $n < N_{max}$, faire les étapes 3 à 6,
- Étape 3 : Poser $x = g(x_0)$,
- Étape 4 : Si $(x - x_0)^2 \leq \epsilon$, alors afficher x et terminer,
- Étape 5 : Poser $n = n + 1$,
- Étape 6 : Poser $x_0 = x$,
- Étape 7 : Afficher un message d'échec pour nombre d'itérations trop important et terminer.

Dans le cas d'un vecteur de design x à n dimensions associé à un MDA de plusieurs disciplines, deux cas de figure peuvent être considérés.

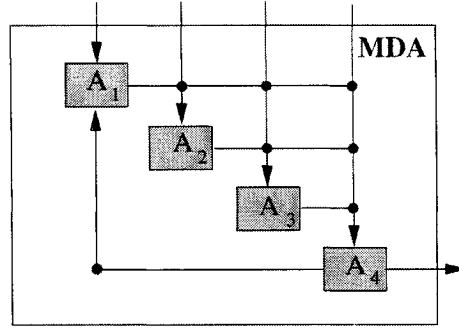


FIGURE 5.5: Exemple de MDA de quatre disciplines ne comportant qu'une seule boucle de retour

Dans le premier, le système d'analyse ne comporte qu'une seule boucle d'itérations (voir figure 5.5). Il existe donc une façon d'ordonnancer les disciplines, de 1 à N , de façon à ce qu'aucune d'entre elles n'utilise en entrée des résultats d'une analyse ultérieure dans la séquence. Avec les notations définies dans le chapitre 2 :

$$a_1 = A_1(s, l)$$

$$a_2 = A_2(s, l, c_1)$$

$$a_i = A_i(s, l, c_1, c_2, \dots, c_{i-1})$$

$$a_N = A_N(s, l, c_1, c_2, \dots, c_{N-1})$$

soit :

$$c_1 = c_1(s, l)$$

$$c_2 = c_2(s, l, c_1)$$

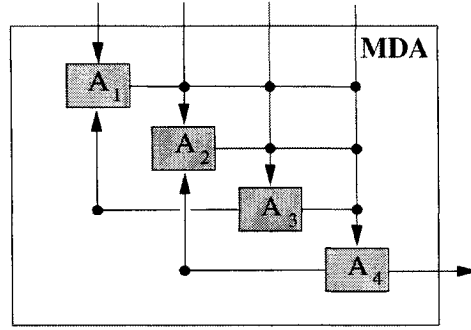


FIGURE 5.6: Exemple de MDA de quatre disciplines comportant deux boucles de retour imbriquées

$$c_i = c_i(s, l, c_1, c_2, \dots, c_{i-1})$$

$$c_N = c_N(s, l, c_1, c_2, \dots, c_{N-1})$$

L'enchaînement successif des analyses de A_1 à A_N permet alors de définir une fonction g équivalente :

$$a = g(s, l, a) = A_N(A_{N-1}(\dots(A_2(A_1(s, l, a))))))$$

Ce cas de figure permet de résoudre le couplage par application directe de la méthode du point fixe et donc en exécutant séquentiellement toutes les analyses disciplinaires, de 1 à N .

Toutefois, d'autres MDA comportent plusieurs boucles d'itérations imbriquées (voir figure 5.6) qu'aucun agencement des disciplines ne peut supprimer. On pourra néanmoins remarquer que certains outils, tels que DeMaid (d'après Park et Choi (2003), Rogers (1992) et Rogers et Bloebaum (1994)), permettent de réorganiser les analyses au sein d'un MDA afin de minimiser le nombre et l'importance des

boucles imbriquées. Dans ce cas de figure, l'ensemble des analyses ne peut pas être assimilé à une fonction g et donc l'algorithme du point fixe ne peut s'appliquer directement.

Devant définir un algorithme de résolution du couplage qui s'applique à tous les cas de figure, nous n'avons pu nous contenter de l'algorithme du point fixe. Une variation de ce dernier, appelée « point fixe en dimension n » (voir Fortin (1995) au chapitre 4.4 pour une présentation de cet algorithme et des théorèmes s'y rapportant), a donc été utilisée. Son principe est de ne plus exécuter séquentiellement les disciplines (puisque'il n'existe plus de séquence naturelle) mais bien simultanément : les valeurs déterminées, pour les variables couplées, à l'itération n sont utilisées pour calculer celles de l'itération $n + 1$. Ainsi :

$$(a_i)_{n+1} = A_i(s, l, (c_1)_n, \dots, (c_{i-1})_n)$$

Le nouvel algorithme de calcul des variables couplées devient donc :

- Étape 1 : Poser $n = 0$,
- Étape 2 : Tant que $n < N_{max}$, faire les étapes 3 à 6,
- Étape 3 : Pour $i \in [1 : N]$, poser $a_i = A_i(s, l, (c_1)_0, (c_2)_0, \dots, (c_{i-1})_0)$ et en extraire la composante c_i ,
- Étape 4 : Si $\sum_i ((c_i) - (c_i)_0)^2 \leq \epsilon$, alors afficher c et terminer,
- Étape 5 : Poser $n = n + 1$,
- Étape 6 : Pour $i \in [1 : N]$, poser $(c_i)_0 = (c_i)$,
- Étape 7 : Afficher un message d'échec pour nombre d'itérations trop important et terminer.

Par rapport à la méthode du point fixe applicable au premier cas de figure, l'al-

gorithme choisi pour PMDO comporte l'inconvénient de décaler d'une itération la prise en compte d'une valeur de c_i à peine calculée, ce qui ralentit probablement la convergence du processus. Cependant, cette nouvelle formulation a l'avantage de permettre d'exécuter les N analyses en parallèle, ce qui, en fin de compte, est susceptible de faire gagner du temps de calcul.

Pour finir, il faut souligner que, comme pour le cas unidimensionnel, la convergence de la méthode du point fixe en dimension n dépend de caractéristiques propres au système (norme de la matrice associée) ainsi que du choix du vecteur initial. L'algorithme mis en place ne convient pas nécessairement à tous les cas d'utilisation de PMDO.

5.4.3 Séquence d'exécution

Commandée par le bloc *Task Process Control* du *description file Coordinator.desc*, l'exécution d'une itération FIO de PMDO fait appel à plusieurs routines *Calculation Blocks* spécifiques. La description exhaustive de leur fonctionnement serait laborieuse et injustifiée dans le présent document. Toutefois, les principales étapes de la séquence d'exécution FIO sont indiquées ci-après à la manière de ce qui a déjà été fait pour DO.

1. En début de chaque itération système, c'est-à-dire de chaque itération de l'optimiseur *Coordinator*, ce dernier fixe tout d'abord la valeur de toutes les variables potentielles non couplées.
2. L'itération précédente est alors testée par appel d'un bloc de calcul appelé *TestLastIteration*. Ce code TCL permet de reconnaître les itérations ayant abouti à un design faisable ou optimal puis, le cas échéant, de commander au JDM la mise à jour des fichiers XML de résultats.

Les étapes suivantes visent à résoudre le couplage pour l'itération en cours. L'algorithme de la méthode du point fixe modifié est programmé en TCL dans un *Calculation Block* nommé *IterateFixedPoint*. Son exécution (étapes 3 à 9) est organisée autour d'une boucle de type *While* comportant deux tests. Cette boucle est exécutée tant que le nombre maximum d'itérations de résolution reste inférieur au maximum fixé par l'administrateur et que le critère de convergence n'a pas été atteint.

3. Après incrémentation d'un compteur des itérations de résolution, la procédure TCL *Export* est exécutée pour exporter les valeurs de variables non couplées vers le JDM qui met à jour les champs correspondants dans le *Variables_Block*.
4. Les sous-tâches *Disciplinei* sont toutes exécutables simultanément.

L'exécution d'une tâche de type *Disciplinei* se limite à trois étapes simples :

5. Importation des valeurs de variables depuis le *Variables_Block*,
6. Exécution d'une analyse disciplinaire (via le JDM et les *Discipline_Blocks*),
7. Exportation des résultats écrits eux aussi dans le *Variables_Block*.

Suite aux analyses disciplinaires et afin d'analyser les résultats d'analyse, le *Coordinator* effectue les opérations ci-dessous :

8. Le *Coordinator* exécute la procédure TCL *Import* afin de lire les valeurs du *Variables_Block*.
9. Le *Calculation Block IterateFixedPoint* est utilisé pour calculer la nouvelle valeur de chacune des variables couplées ainsi que la valeur de la mesure de convergence (somme des carrés des différences de valeurs des variables couplées entre le début et la fin de l'itération de résolution courante).

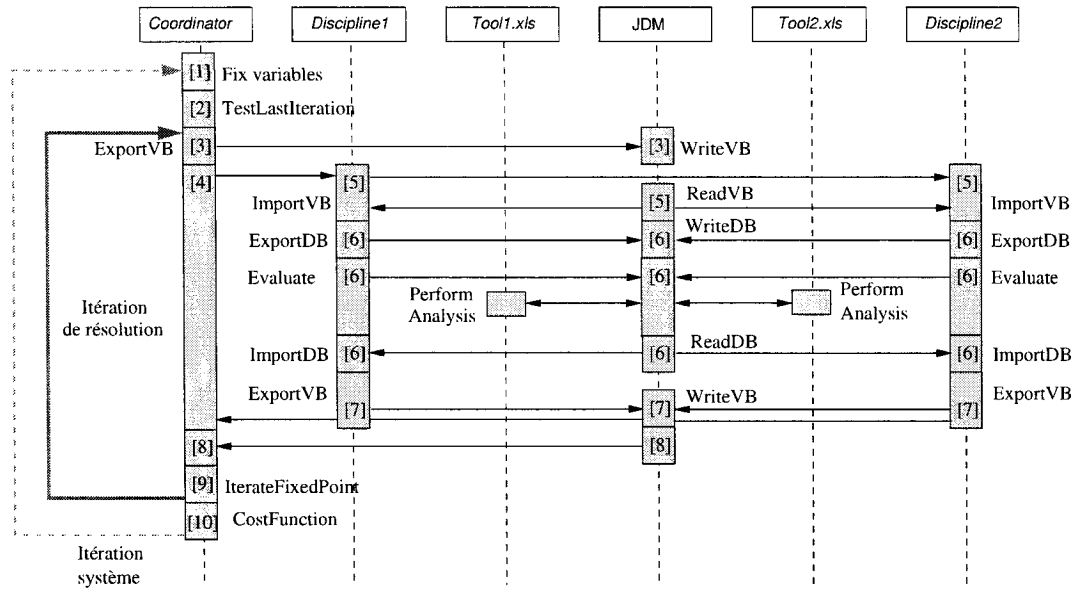


FIGURE 5.7: Séquence d'exécution détaillée pour une itération du *Coordinator* lors de la résolution d'un MDO de deux disciplines par la méthode FIO

Cette étape 9 correspond à la fin de la boucle *While*.

10. En cas de succès de la méthode du point fixe (sortie de la boucle *While* par le critère de convergence et non pas par échec), la fonction coût associée à l'itération effectuée est alors calculée par appel du *Calculation Block* baptisé *CostFunction*. En cas d'échec (si la méthode est divergente ou bien si le nombre maximal d'itérations est atteint), une valeur de fonction coût arbitrairement grande est alors imposée (un *Calculation Block* nommé *PenaliseCostFunction* est exécuté à cette fin) pour signifier à l'optimiseur qu'il emprunte une direction erronée.

La figure 5.7 résume graphiquement les étapes précédentes en un diagramme de séquence qui correspond à une itération FIO dans le cas d'un MDO à deux disciplines. Les numéros repris sur ce diagramme correspondent aux étapes précédentes.

5.5 Configuration automatisée selon DAO

5.5.1 Éléments du MDO configuré pour DAO

Dans le cas où la méthode DAO est utilisée, un optimiseur maître reste bien entendu nécessaire au niveau du *Coordinator* tandis que les sous-tâches *Disciplinei* sont de simples analyses disciplinaires : pour des disciplines dont le *Running_Mode* a été fixé à *Optimize*, l'*Evaluation Mode* est fixé à *single*. Seul optimiseur dans le système, le *Coordinator* gère donc toutes les variables du système, y compris celles qui sont couplées. Le couplage interdisciplinaire est dans ce cas traité par l'imposition de contraintes de consistance et la création de variables dupliquées *CopyI*.

La décomposition DAO permettant de dissocier les disciplines (voir la figure 5.8), les analyses *Disciplinei* sont donc exécutées en parallèle (et potentiellement sur plusieurs ordinateurs distincts).

5.5.2 Séquence d'exécution

Ici encore, les principales étapes de la séquence d'exécution DAO sont énumérées ci-après puis reprises à l'aide d'un diagramme de séquence simplifié (voir figure 5.9).

1. En début de chaque itération de l'optimiseur maître *Coordinator*, ce dernier fixe la valeur de toutes les variables potentielles.
2. L'itération précédente est testée par appel du bloc de calcul *TestLastIteration*.
3. La procédure TCL *Export* est exécutée pour exporter les nouvelles valeurs des variables potentielles vers le JDM qui met à jour les champs correspondants dans le *Variables_Block*.

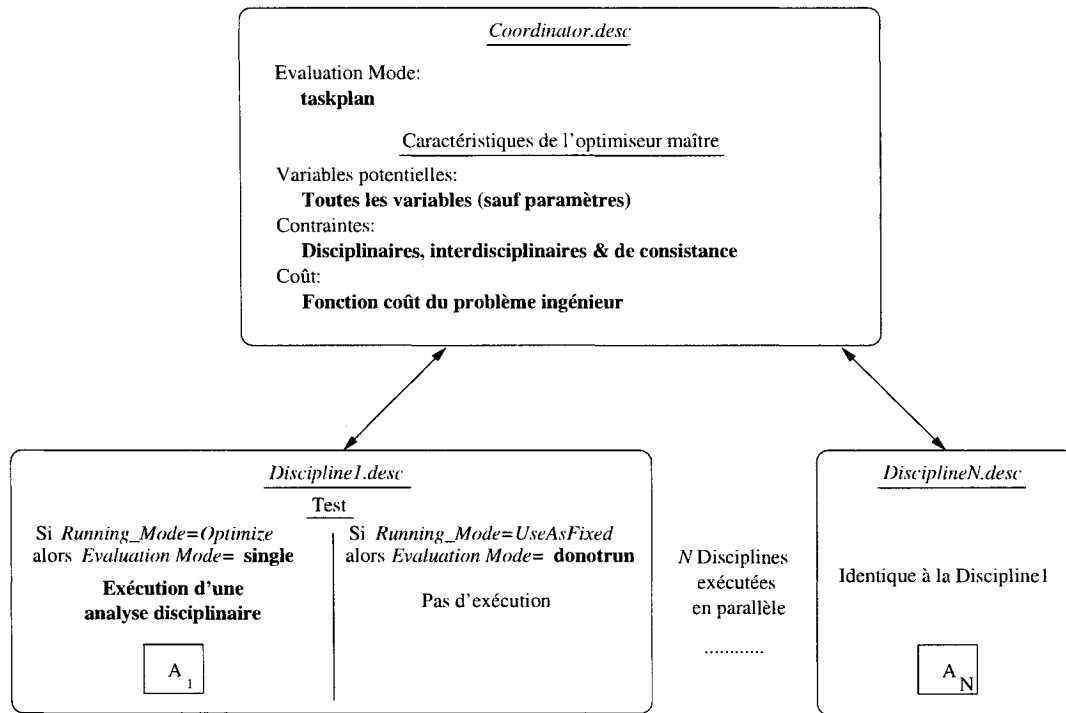
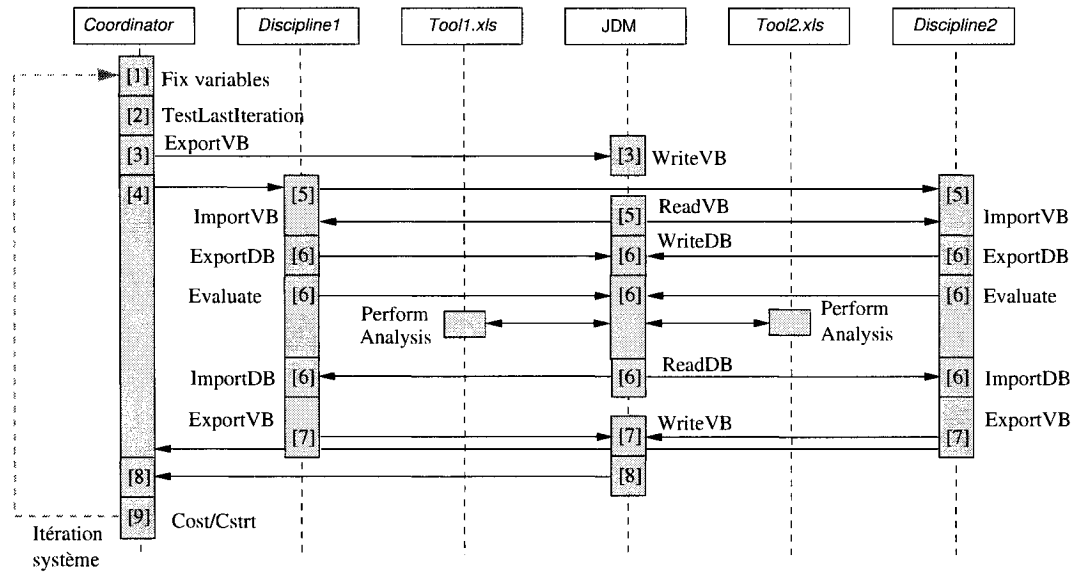


FIGURE 5.8: MDO originel reformulé selon la méthode DAO

FIGURE 5.9: Séquence d'exécution détaillée pour une itération du *Coordinator* lors de la résolution d'un MDO de deux disciplines par la méthode DAO

4. Les sous-tâches *Disciplinei* sont toutes exécutables simultanément.

Pour toutes les disciplines incluses avec un *Running_Mode* fixé à *Optimize*, les trois étapes de la sous-tâche sont exécutées en parallèle :

5. Importation des valeurs de variables requises depuis le *Variables_Block*,
6. Exécution d'une analyse disciplinaire (via le JDM et les *Discipline_Blocks*),
7. Exportation des résultats écrits eux aussi dans le *Variables_Block*.

Suite aux analyses disciplinaires et afin d'exploiter les résultats, le *Coordinator* effectue les opérations ci-dessous :

8. Le *Coordinator* exécute la procédure TCL *Import* afin de lire les valeurs du *Variables_Block*.
9. Le *Coordinator* évalue les contraintes et la fonction coût du système en exécutant les *Calculation Blocks CouplingConstraints* et *CostFunction*.

5.6 Configuration automatisée selon DO

5.6.1 Éléments du MDO configuré pour DO

En complément de l'optimiseur maître, un optimiseur disciplinaire est associé à chaque discipline locale (confère figure 5.10). Les optimiseurs disciplinaires gèrent les variables locales tandis que le *Coordinator* à la responsabilité des variables systèmes. Le couplage entre les disciplines parallélisées est résolu au moyen de contraintes de consistance appliquées à l'optimiseur maître.

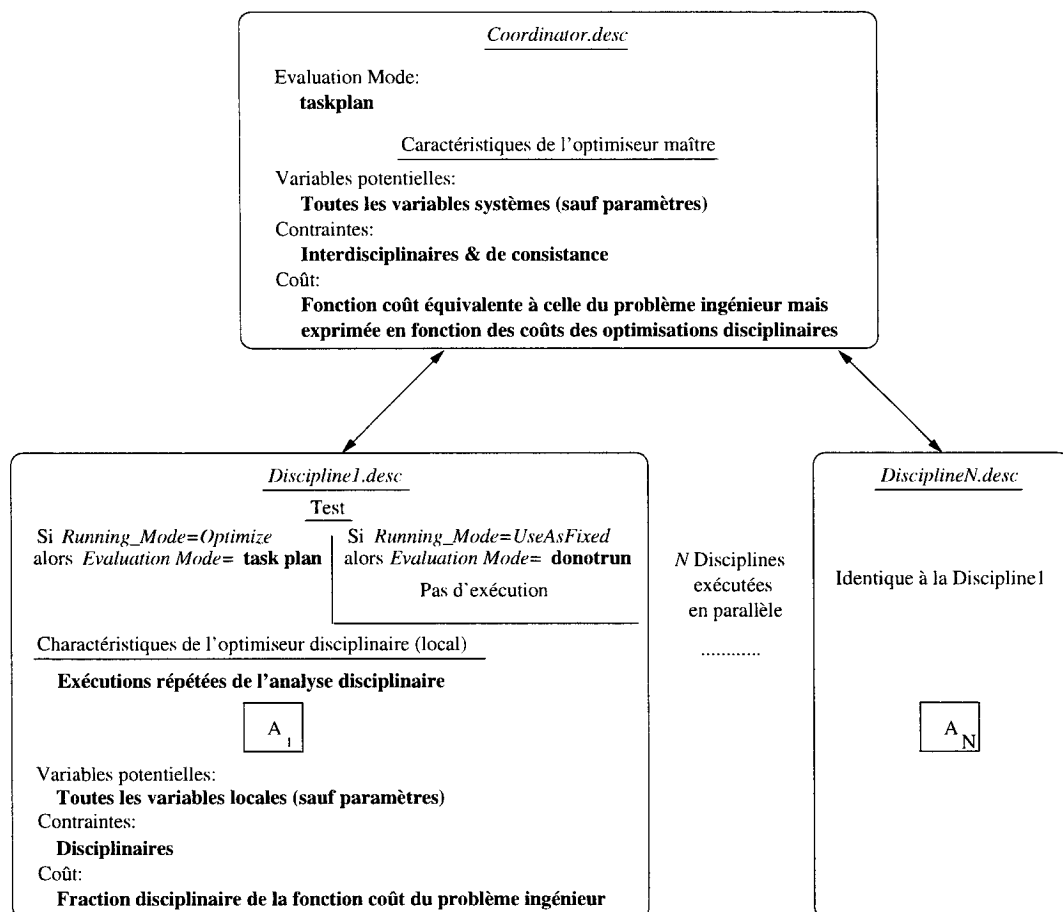


FIGURE 5.10: MDO original reformulé selon la méthode DO

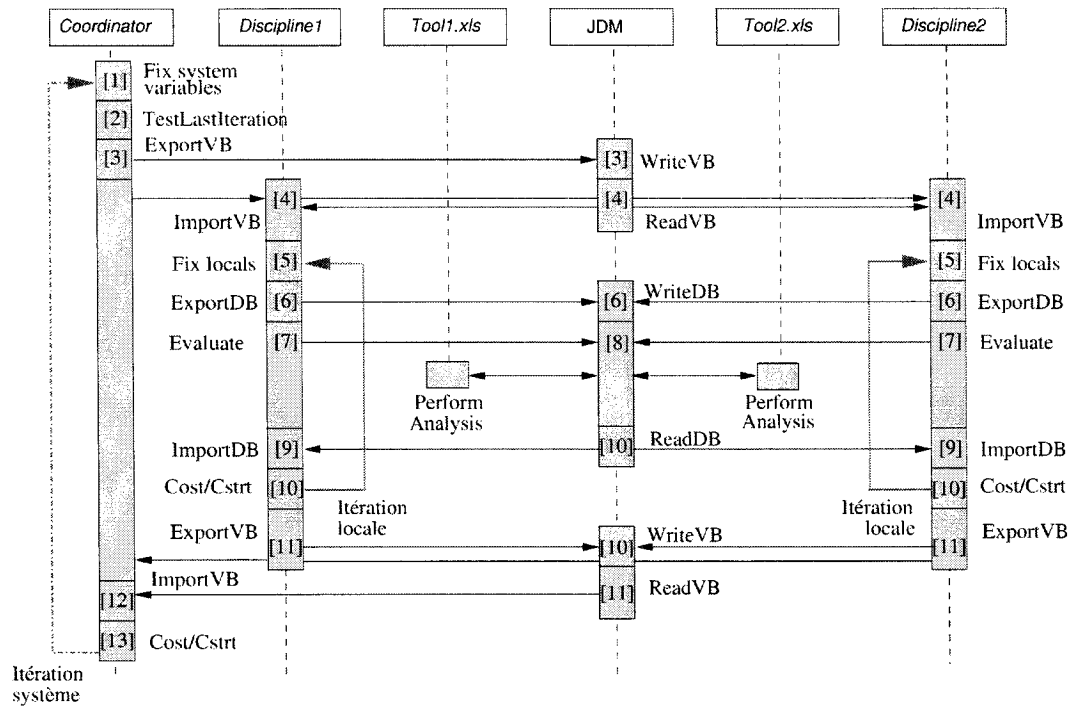


FIGURE 5.11: Séquence d'exécution détaillée pour une itération du *Coordinator* lors de la résolution d'un MDO de deux disciplines par la méthode DO

5.6.2 Séquence d'exécution

Les étapes d'exécution d'une itération DO ayant déjà été énoncées à titre d'exemple (voir section 4.3.2.1), elles ne seront pas rappelées ici. La figure 5.11 les résume néanmoins graphiquement au moyen d'un diagramme de séquence qui correspond à une itération DO du *Coordinator* dans le cas exemple d'un MDO à deux disciplines couplées. Tel que cela apparaît clairement, les optimisations locales sont découplées et peuvent être effectuées simultanément sur plusieurs machines. Le passage de l'étape 10 à la 11 n'est possible qu'une fois toutes les sous-tâches terminées.

5.7 Ajout d'une méthode de décomposition

Le prototype actuel a été développé initialement avec la méthode de décomposition DO. Dans un deuxième temps, les méthodes FIO puis DAO ont été rajoutées.

L'ajout d'une méthode supplémentaire nécessitera l'intervention d'un administrateur PMDO ayant le droit de modifier les fichiers *Coordinator.desc* et *Disciplinei.desc*. Il s'agira alors de rajouter des blocs dynamiques pour inclure les nouvelles instructions propres à la décomposition. Cette étape impliquera peut-être également la définition de nouveaux blocs *Calculation blocs*. Enfin, l'édition du fichier texte *pmdo.properties* de commande de l'interface permettra d'ajouter un nouveau choix dans la liste des méthodes de décomposition offertes.

CHAPITRE 6

VALIDATION DU PROTOTYPE PMDO À L'AIDE DE CAS TESTS

Ce dernier chapitre présente les principaux tests effectués pour valider le prototype de PMDO en terme de flexibilité, de déroulement du processus d'optimisation et de performances.

La flexibilité est mise à l'épreuve en variant les méthodes de décomposition utilisées, le MDO défini (par l'inclusion d'un nombre variable de disciplines et par l'imposition de *Running_Mode* changeants) ainsi que les conditions d'exécution (système d'exploitation et nombre d'ordinateurs) des tests.

Le bon déroulement des exécutions est validé en utilisant PMDO pour résoudre un problème test TP300 issu de la littérature. Cet exemple repose sur l'évaluation de trois fonctions analytiques couplées. Il est associé à une optimisation convergeant vers une solution unique et aisément comparable à celle obtenue par l'exécution de PMDO.

Enfin, les mesures effectuées au cours des essais permettent une évaluation de certaines performances de l'outil PMDO: comparaison de la vitesse de convergence ou bien de la précision de résolution du couplage selon la décomposition .

Les développements suivants exposent les méthodes et conditions des essais, présentent le cas test TP300 choisi et résument les résultats obtenus. Une importante voie d'améliorations futures ressort de l'analyse des performances.

6.1 Nature et conditions de réalisation des tests

6.2 Conditions de test

Ces tests ont pour objet de valider la conception et la réalisation du prototype. Afin de s'assurer qu'il corresponde effectivement aux attentes en terme de flexibilité, il est nécessaire de valider la configuration automatisée pour les différentes méthodes de décomposition, les mécanismes d'intégration et d'inclusion d'une discipline ainsi que la portabilité d'un système d'exploitation à un autre. Il est également important de tester que les mécanismes d'optimisation permettent effectivement de converger vers un optimum. Enfin, quelques tests permettront d'évaluer les performances du prototype en terme de vitesse de convergence et de qualité de résolution du couplage interdisciplinaire. L'objectif des tests présentés est donc de démontrer que l'architecture de PMDO et son mode de fonctionnement constituent une approche intéressante qu'il est judicieux de développer ultérieurement jusqu'à la mise en place chez Pratt & Whitney Canada. Par contre, l'objectif n'est pas de raffiner les méthodes de résolution pour un problème donné. Pour cette raison, les tests ont tous été faits en laissant iSIGHT choisir les algorithmes d'optimisation utilisés, les critères d'arrêt et pas de différentiation. Aucune analyse de sensibilité n'a été effectuée ni aucun choix des variables de design. Les temps de résolution obtenus ne doivent donc être interprétés que comme des ordres de grandeur.

Les tests ont été conduits sur plusieurs plateformes d'exploitation, dans les locaux de Pratt & Whitney ainsi qu'à Polytechnique. Le composant JDM a été testé avec succès et utilisé sur Windows et Linux. Le fonctionnement du composant d'optimisation a été testé sur des ordinateurs UNIX et Linux (mais pas sur Windows, faute de licence pour le logiciel iSIGHT). Au cours des essais, plusieurs configurations ont été testées avec 1, 2 ou 3 ordinateurs simultanément :

1. tous les composants sur l'ordinateur servant à l'affichage,
2. le JDM sur un ordinateur tandis qu'iSIGHT et l'affichage sont sur une autre machine,
3. le JDM, iSIGHT et l'affichage sur 3 ordinateurs distincts du même réseau.

L'interface PMDO_GUI n'a pas été utilisée pour effectuer les tests décrits. Elle fut remplacée par une routine lisant directement les informations nécessaires dans un fichier XML *Input_PMDO.xml* contenant le GULBlock établi au préalable. Cette modification ne change rien au reste de l'exécution de PMDO, mais permet d'utiliser des scripts **.sh* pour automatiser l'exécution des tests. Chacun des essais effectués étant reproductible (une même exécution de PMDO effectuée plusieurs fois de suite conduit toujours au même résultat avec le même nombre d'itérations) mais tributaire de l'encombrement du réseau ou des ordinateurs utilisés (tel que nous le mesurerons plus loin), ces scripts servent à reproduire plusieurs fois chaque essai en enregistrant à chaque fois les durées d'exécution (temps réel, temps CPU et temps Système) pour ensuite en calculer une valeur moyenne.

Tel que précisé précédemment, seules des feuilles de calculs Excel ont été intégrées dans le prototype. Les fichiers **.xls* correspondant sont interprétés par le JDM (au moyen d'une librairie spéciale appelée JSC ou Java Spreadsheet Component) et les opérations équivalentes à celles constituant la feuille de calcul sont alors exécutées sur la machine virtuelle Java à chaque fois qu'une analyse disciplinaire est requise. Le logiciel Excel n'est donc jamais utilisé pour l'exécution de PMDO.

Au moment où ces tests ont été conduits, le travail d'intégration des outils d'analyse de Pratt & Whitney Canada (qui devront constituer la première version de PMDO) n'était pas suffisamment avancé pour permettre des tests complets. En effet, les feuilles de calcul requises n'existant pas pour toutes les disciplines, un étudiant de génie mécanique, Daniel Tardif, consacre son PFE à les constituer à

partir de PMDO Light, un programme utilisé en R&D pour l'analyse préliminaire des moteurs. Ce long travail de préparation des fichiers Excel et de rédaction des *properties files* est actuellement en cours. Quand les tests furent nécessaires, seules les quinze disciplines suivantes étaient en cours d'intégration dans PMDO :

- *Discipline1* (Fan) : analyse de la soufflante (fan),
- *Discipline2* (Axial LPC) : analyse aérodynamique du compresseur basse pression,
- *Discipline3* (IC Duct) : calcul des pertes de charge dans la conduite d'entrée du compresseur,
- *Discipline4* (Axial HPC) : analyse aérodynamique du compresseur haute pression,
- *Discipline5* (Centrifugal) : analyse du compresseur centrifuge,
- *Discipline6* (Cent-Disk) : analyse aérodynamique du compresseur centrifuge,
- *Discipline7* (Axial Comb) : analyse aérodynamique de la chambre de combustion axiale,
- *Discipline8* (Rev Comb) : analyse aérodynamique de la chambre de combustion à flux inversé,
- *Discipline10* (HPT-Disk and LPT-Disk) : analyse structurelle (calcul des contraintes dans les rotors) pour les turbines hautes et basse pression,
- *Discipline12* (Shaft) : analyse structurelle des arbres de rotation,
- *Discipline13* (Nacelle Turbofan) : calcul de la nacelle englobant une turbosoufflante,
- *Discipline14* (Weight) : estimation du poids du moteur,
- *Discipline15* (Production Cost) : estimation des coûts de production,
- *Discipline16* (Maintenance Cost) : estimation des coûts de maintenance,
- et *Discipline17* (Noise) : estimation du bruit produit.

Des essais préliminaires ont permis de vérifier que toutes les étapes du pré-traitement s'effectuent correctement pour un nombre de disciplines intégrées allant

jusqu'à dix. Le MDA constitué automatiquement lorsque les 10 premières disciplines sont utilisées (*Running_Mode* fixé à *Optimise* pour toutes les dix) comporte 551 variables (ce qui signifie que le *Variables_Block* contient 551 éléments distincts) dont 20 couplées, associées à 20 autres variables dupliquées, ainsi que 25 partagées. Toutefois, ce MDA étant très volumineux et peu pratique pour des tests exhaustifs (il est difficile de déterminer le couplage interdisciplinaire pour chaque configuration envisageable des 10 disciplines et ensuite de le comparer au résultat de l'initialisation du fichier *PMDO_Datafile.xml*), il a été décidé de constituer un cas test présenté ci-après.

6.3 Présentation du cas test analytique utilisé

Le cas test analytique utilisé est créé à partir d'un exemple issu d'un article proposant plusieurs problèmes pertinents pour le test des méthodes d'optimisation. Parmi les exemples proposés par Schittkowski (1987), nous avons choisi celui intitulé TP300. Les équations le constituant forment un système analytique facilement transcribable à l'aide de feuilles de calcul Excel :

$$F_1 = (x_1^2 - 2x_1 + 1) + \sum_{i=2}^6 (2x_i^2 - 2x_i x_{i-1} + 1) \quad (6.1)$$

$$F_2 = \sum_{i=7}^{12} (2x_i^2 - 2x_i x_{i-1} + 1) \quad (6.2)$$

$$F_3 = \sum_{i=13}^{18} (2x_i^2 - 2x_i x_{i-1} + 1) \quad (6.3)$$

$$a_1 = \frac{F_1}{1 + \beta (a_2 + a_3)} \quad (6.4)$$

$$a_2 = F_2 \sqrt{1 + \beta (a_1 + a_3)} \quad (6.5)$$

$$a_3 = F_3 \sqrt{1 + \beta (a_1 + a_2)} \quad (6.6)$$

À ce système, est associé le problème d'optimisation (sans contrainte) suivant :

$$\underset{\{x_1 \dots x_{18}\}}{\text{minimise}} \ a_1 + a_2 + a_3 \quad (6.7)$$

L'ensemble de ces équations peut-être assimilé à un MDA de trois disciplines couplées. Les paires d'équations 6.1 et 6.2, 6.3 et 6.4 ainsi que 6.5 et 6.6 constituent respectivement les Discipline1, Discipline2 et Discipline3. Ce MDA exemple comporte un paramètre β et 24 variables (18 variables de design x et 6 variables de sortie a ou F) dont 2 partagées $s = \{x_6; x_{12}\}$ et 3 couplées $c = \{a_1; a_2; a_3\}$. Le paramètre β permet de varier la force de couplage entre les disciplines (pour $\beta = 0$, les trois disciplines sont totalement découplées). Le couplage peut facilement être résolu par la méthode du point fixe vectoriel donc le MDA s'accommode bien de la résolution par FIO.

Pour ce problème d'optimisation, les valeurs des composantes d'un vecteur solution X_{sol} unique sont connues avec précision pour le cas $\beta = 0.1$. Partant d'une valeur initiale de 1 pour chacune des variables ($X_{init} = (1; 1; \dots; 1)$), la recherche de cet optimum constitue le cas test que nous utilisons pour valider PMDO. Pour ce faire, il a donc été nécessaire de rajouter trois disciplines supplémentaires au prototype, selon la procédure exposée ci-après.

6.4 Procédure d'ajout de nouvelles disciplines

L'intégration d'une nouvelle discipline dans PMDO est une opération effectuée par l'administrateur uniquement. Très simple, elle se divise en 6 étapes :

1. Assignment d'un numéro i : la discipline sera identifiée par un numéro i choisi librement à condition qu'il ne serve pas déjà. Pour plus de clarté et pour les disciplines de Pratt & Whitney, un mécanisme de numérotation

de 2+2 chiffres a été proposé pour faciliter l'identification de la discipline, et éventuellement du rotor, en fonction du nombre. Cette nouvelle numérotation remplacera celle utilisée actuellement (de 1 à 17).

2. Création d'un dossier : un dossier *Disciplinei* est créé dans le répertoire de travail.
3. Constitution d'une feuille de calcul Excel : une feuille de calcul effectuant les opérations requises est créée et nommée *Disciplinei.xls*.
4. Constitution d'un properties file : un fichier descriptif *Disciplinei.properties* est constitué afin de permettre au JDM d'interpréter la feuille de calcul.
5. Création d'un fichier *description file* pour iSIGHT : le fichier *Disciplinei.desc* est obtenu très simplement à l'aide d'un exemple *Disciplinej.desc* et d'un éditeur de texte. Il suffit, dans le fichier exemple, de remplacer automatiquement la chaîne de caractères *Disciplinej* par *Disciplinei*, puis de le sauvegarder sous le nouveau nom.
6. Edition du fichier de description de l'interface PMDO_GUI : à l'aide d'un éditeur de texte, l'administrateur rajoute *Disciplinej* à la liste des disciplines intégrées (soit potentiellement incluses). Cela fait en sorte qu'une page supplémentaire dans l'interface graphique PMDO_GUI permettra ensuite à l'utilisateur d'inclure la nouvelle discipline pour les optimisations suivantes.

6.5 Résultats

6.5.1 Tests de validation de la flexibilité

Des essais ont été effectués avec les disciplines disponibles afin de vérifier le bon fonctionnement des procédures d'initialisation de PMDO pour plusieurs configurations et plusieurs problèmes.

En laissant iSIGHT effectuer le pré-traitement d'un système constitué des feuilles de calcul P&WC, il fut possible de vérifier en situation le bon déroulement de l'étape de création automatique d'un MDA de 551 variables de design. Cet essai permis également d'observer le bon déroulement de l'initialisation d'un composant d'optimisation comportant dix disciplines.

Par la suite, à l'aide des trois disciplines de l'exemple TP300, des essais supplémentaires ont été réalisés pour un plus grand nombre de configurations. Les tests furent entrepris en variant de un à trois le nombre des disciplines incluses, en en fixant une (*Running_Mode* fixé à *UseAsFixed*) et en changeant la méthode de décomposition. À chaque fois, les résultats du pré-traitement furent comparés aux attentes en vérifiant la liste des variables potentielles, les contraintes de consistance créées ainsi les *evaluation mode* de chaque tâche. Le fonctionnement de PMDO semblant conforme aux attentes, les trois disciplines du cas test TP300 furent ensuite employées à valider l'étape d'optimisation.

6.5.2 Tests de validation de l'optimisation

À plusieurs reprises, les trois disciplines constituant le problème test TP300 ont été incluses avec un *Running_Mode* fixé à *Optimize* afin de valider l'optimisation et la résolution du couplage pour chacune des méthodes de décomposition. Sont présentés ci-dessous les résultats de convergence pour FIO, DAO et enfin DO. Dans chaque cas, un premier graphique de convergence présente l'évolution des valeurs de fonction coût (*Cost*) et du critère de faisabilité (*Feasibility*)¹ en fonction

¹Le critère de faisabilité est une grandeur discrète, comprise entre 0 et 9, calculée par iSIGHT au terme de chaque itération. Les valeurs 0 à 6 caractérisent différents designs infaisables. Une faisabilité de 7 indique un design faisable, mais correspondant à une valeur de coût moins bonne que l'optimum en cours. la valeur 9, enfin, marque un design faisable meilleur que tout ceux trouvés précédemment. Au terme du processus d'optimisation, le résultat optimum est donc le

de l'itération du Coordinateur. Sur un second graphique, n'apparaissent que les itérations correspondant à une amélioration du design : la courbe de faisabilité est donc une fonction croissante dont le dernier point (*Feasibility* = 9) correspond à l'optimum du processus d'optimisation.

6.5.2.1 Résolution avec la décomposition FIO

Dans le cas de la décomposition FIO, la formulation du problème test est la suivante :

$$\underset{\{s, l_1, l_2, l_3\} \in \Gamma}{\text{minimise}} \quad a_1 + a_2 + a_3 \quad (6.8)$$

$$\text{avec : } s = \{x_6; x_{12}\},$$

$$l_1 = \{x_1; x_2; x_3; x_4; x_5\}, \quad l_2 = \{x_7; x_8; x_9; x_{10}; x_{11}\}, \quad l_3 = \{x_{13}; x_{14}; x_{15}; x_{16}; x_{17}; x_{18}\},$$

$$\Gamma = \{x \in \mathbb{R}^{18} \mid 0 \leq x_i \leq 10, i \in [1..18]\}$$

L'optimiseur unique a la charge de toutes les variables de design non couplées. La détermination des valeurs de ces dernières, par la méthode du point fixe, est un processus itératif dont la fin dépend de la tolérance de convergence ϵ (voir la section 5.4.2). Le choix de ϵ influence donc directement la résolution du couplage interdisciplinaire et donc la précision des analyses du MDA ainsi que le temps de calcul. La relation entre ϵ et la tolérance moyenne de résolution de couplage Δ_{moy} est fonction du nombre de variables couplées $N_{couplees}$:

$$\Delta_{moy} = \sqrt{\frac{\epsilon}{N_{couplees}}} \quad (6.9)$$

dernier design trouvé pour lequel la faisabilité est de 9.

Plusieurs exécutions de PMDO ont été effectuées en faisant varier le critère ϵ depuis $1.E-4$ à $1.E-10$. À chaque fois, l'optimisation a convergé en un point proche de l'optimum connu, mais toujours légèrement moins bon (fonction coût légèrement supérieure). Le tableau 6.1 résume succinctement les résultats obtenus. Les tendances observées sont conformes aux attentes :

- pour chaque design évalué par l'optimiseur, le nombre moyen d'itérations requises pour la convergence des variables couplées augmente de 1.63 à 4.84, de façon inversement proportionnelle au logarithme de ϵ (voir figure 6.1),
- le temps de calcul nécessaire à la convergence augmente au même rythme, passant de 4542 secondes pour $\epsilon = 1.E - 4$ à presque 11000 secondes si $\epsilon = 1.E - 10$ (voir figure 6.2 pour comparaison avec les temps mesurés par la suite pour les décompositions DAO et DO),
- enfin, la valeur de la fonction coût, évaluée à l'optimum trouvé, diminue aussi avec la tolérance de convergence (voir figure 6.3 pour comparaison avec l'optimum trouvé ensuite pour la décomposition DAO).

L'efficacité du processus d'optimisation diminue donc rapidement avec ϵ : les coûts en temps de calcul et en nombre d'évaluations des disciplines augmentent sans commune mesure avec l'amélioration du résultat. Les figures 6.4 et 6.5 résument le processus de convergence pour un compromis de $\epsilon = 1.E - 5$ soit une valeur moyenne de résolution du couplage de $\Delta_{moy} = \sqrt{\frac{1.E-5}{3}} \approx 2.E - 3$.

TABLEAU 6.1: Résultats de convergence FIO en fonction du paramètre ϵ , pour le cas test TP300

ϵ	1.E-4	1.E-5	1.E-6	1.E-7	1.E-8	1.E-9	1.E-10
Coût optimum	5,9232	5,7872	5,7645	5,7609	5,7420	5,7457	5,7414
Itérations <i>Coordinator</i>	1472	2344	2707	1470	2294	2252	1920
Itérations <i>Disciplinei</i>	2400	4949	6429	4464	7712	8569	9288
Moyenne convergence	1,63	2,11	2,37	3,03	3,36	3,81	4,84
Temps réel (s) ¹	4542	5492	7295	4696	8644	10066	10982
Temps iSIGHT (s) ²	9,56	20,34	31,11	18,42	30,65	34,34	36,82

¹ Afin de minimiser l'influence de l'encombrement du réseau et des machines, les valeurs de temps de calcul sont obtenues en effectuant la moyenne d'un minimum de deux essais identiques effectués sur les même ordinateurs.

² Le temps iSIGHT est la somme des temps CPU et System utilisés par le composant d'optimisation.

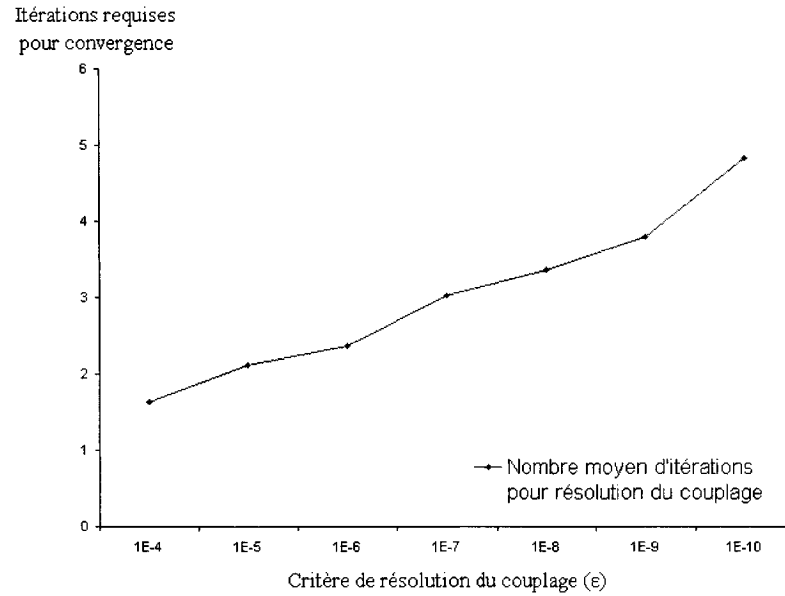


FIGURE 6.1: Décomposition FIO : nombre moyen d'itérations requises pour la résolution du couplage interdisciplinaire

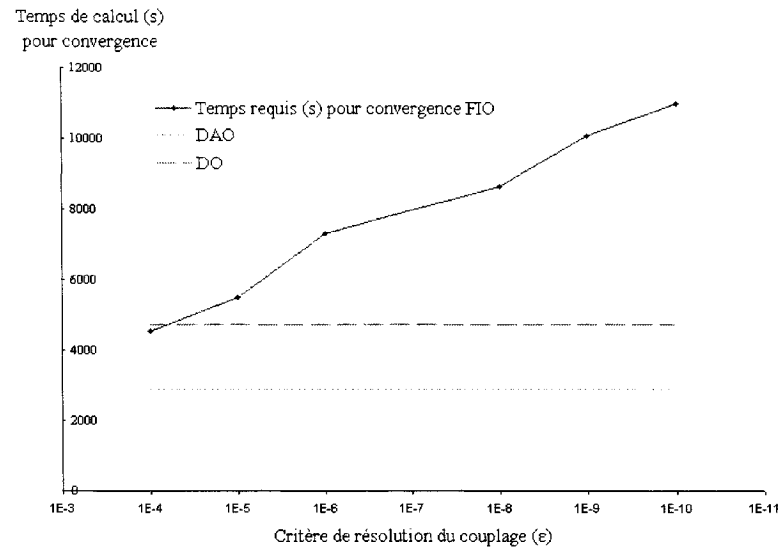


FIGURE 6.2: Décomposition FIO : temps de calcul requis (secondes) avant convergence

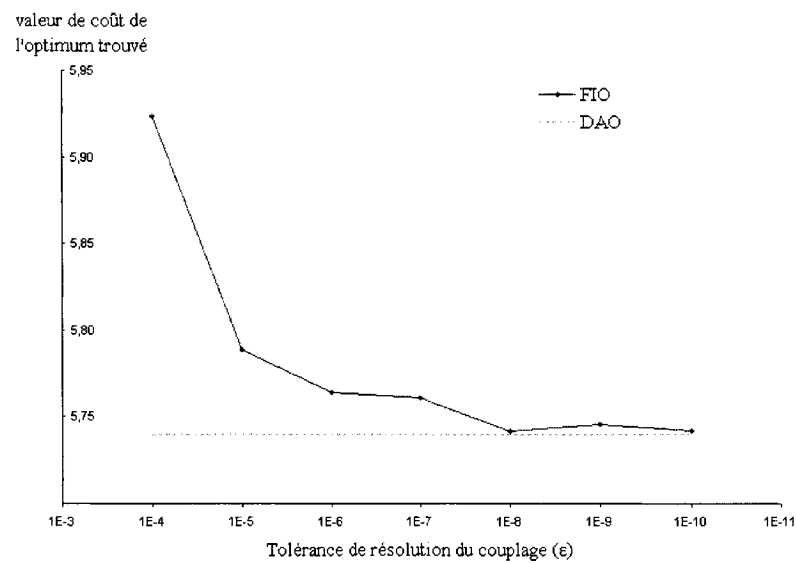


FIGURE 6.3: Décomposition FIO : valeur de la fonction coût à l'optimum, en fonction de ϵ

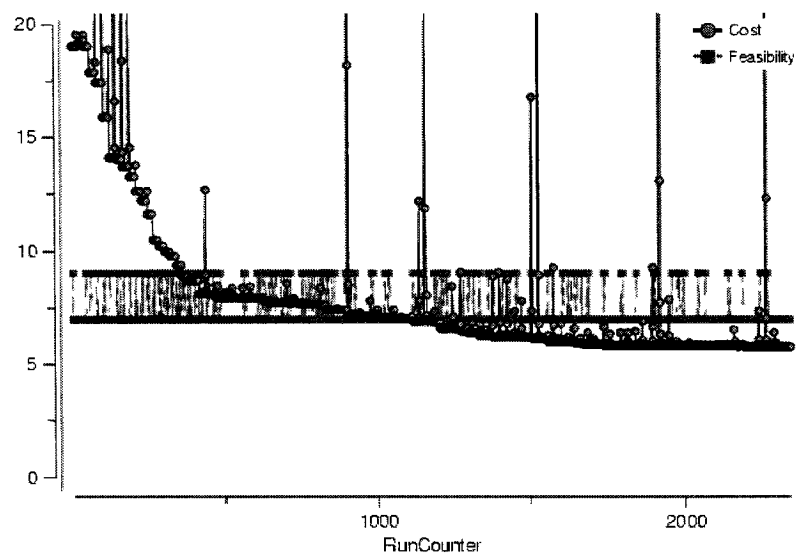


FIGURE 6.4: Progression de la convergence FIO

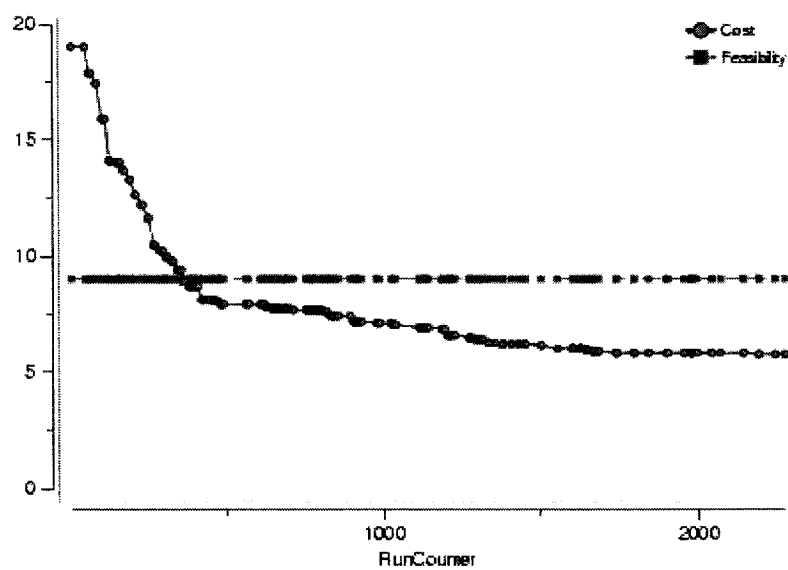


FIGURE 6.5: Améliorations de la convergence FIO

6.5.2.2 Résolution avec la décomposition DAO

Pour la décomposition DAO, la formulation du problème d'optimisation est :

$$\begin{aligned} & \underset{\{s,t,l_1,l_2,l_3\} \in \Gamma}{\text{minimise}} && a_1 + a_2 + a_3 \end{aligned} \tag{6.10}$$

$$\text{tel que : } a_1 - \text{CopyIa}_1 = 0,$$

$$a_2 - \text{CopyIa}_2 = 0,$$

$$a_3 - \text{CopyIa}_3 = 0,$$

$$\text{avec : } s = \{x_6; x_{12}\}, \quad t = \{\text{CopyIa}_1; \text{CopyIa}_2; \text{CopyIa}_3\},$$

$$l_1 = \{x_1; x_2; x_3; x_4; x_5\}, \quad l_2 = \{x_7; x_8; x_9; x_{10}; x_{11}\},$$

$$l_3 = \{x_{13}; x_{14}; x_{15}; x_{16}; x_{17}; x_{18}\},$$

$$\Gamma = \{x \in \mathbb{R}^{21} \mid 0 \leq x_i \leq 10, i \in [1..18], -10 \leq \text{CopyIa}_j \leq 10, j \in [1..3]\}$$

Par rapport à la formulation FIO, les trois variables dupliquées *CopyIa_1*, *CopyIa_2* et *CopyIa_3* s'ajoutent à la liste des variables potentielles déléguées à l'optimiseur unique. De plus, trois contraintes de consistance s'ajoutent à la formulation. La fonction coût est inchangée.

Les graphiques de convergence des figures 6.6 et 6.7 montrent que, parti d'un design initial infaisable, l'optimiseur a convergé en 2931 itérations. Le premier design faisable a été atteint au terme d'environ 800 itérations. La tolérance de satisfaction des contraintes de consistance, laissée à la valeur par défaut de iSIGHT, assure une résolution du couplage avec une précision Δ_{moy} de l'ordre de 1.E-7.

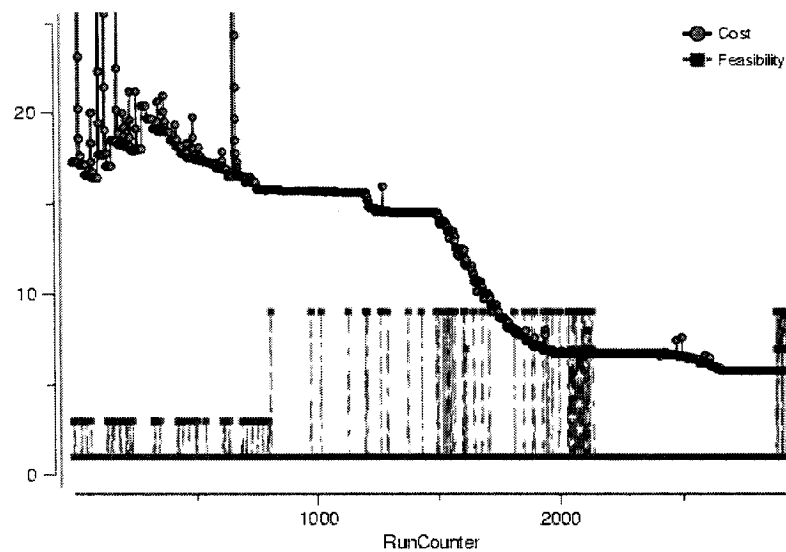


FIGURE 6.6: Progression de la convergence DAO

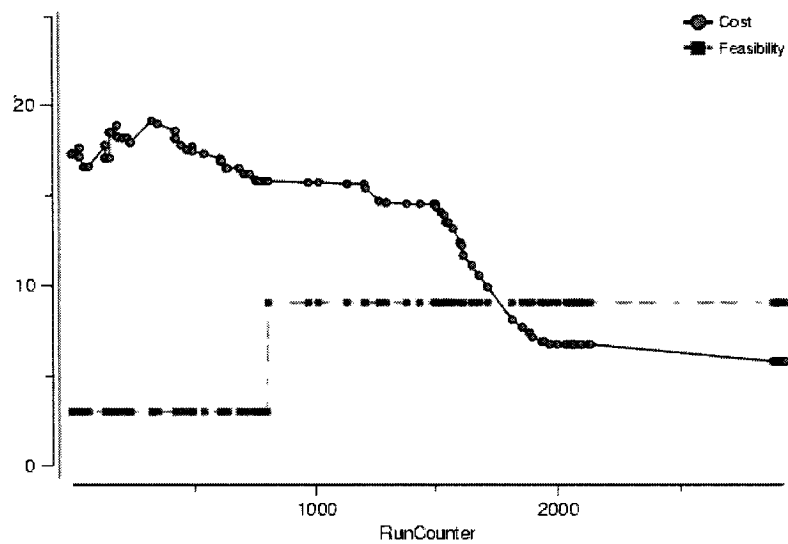


FIGURE 6.7: Améliorations de la convergence DAO

6.5.2.3 Résolution avec la décomposition DO

DO étant une méthode de décomposition multi-niveaux, la formulation équivalente à celle du problème originel requiert la coordination des 4 optimisations présentées ci-après.

Pour l'optimiseur maître (tâche *Coordinator*) tout d'abord (l'exposant ^{opt} identifie les solutions des sous-problèmes d'optimisation) :

$$\underset{\{s,t\} \in \Gamma}{\text{minimise}} \quad a_1^{opt} + a_2^{opt} + a_3^{opt} \quad (6.11)$$

$$\text{tel que : } a_1^{opt} - CopyIa_1 = 0,$$

$$a_2^{opt} - CopyIa_2 = 0,$$

$$a_3^{opt} - CopyIa_3 = 0,$$

$$\text{avec : } s = \{x_6; x_{12}\}, \quad t = \{CopyIa_1; CopyIa_2; CopyIa_3\},$$

$$\Gamma = \{x \in \mathbb{R}^{21} \mid 0 \leq x_i \leq 10, i \in [1..18], -10 \leq CopyIa_j \leq 10, j \in [1..3]\}$$

Pour l'optimiseur associé à la discipline 1 :

$$\underset{\{l_1\} \in \Gamma}{\text{minimise}} \quad a_1 \quad (6.12)$$

$$\text{avec : } l_1 = \{x_1; x_2; x_3; x_4; x_5\}$$

$$\Gamma = \{x \in \mathbb{R}^5 \mid 0 \leq x_i \leq 10, i \in [1..5]\}$$

Pour l'optimiseur associé à la discipline 2 :

$$\begin{aligned}
 & \underset{\{l_2\} \in \Gamma}{\text{minimise}} \quad a_2 & (6.13) \\
 & \text{avec : } l_2 = \{x_7; x_8; x_9; x_{10}; x_{11}\}, \\
 & \Gamma = \{x \in \mathbb{R}^5 \mid 0 \leq x_i \leq 10, i \in [7..11]\}
 \end{aligned}$$

Et enfin, pour l'optimiseur associé à la discipline 3 :

$$\begin{aligned}
 & \underset{\{l_3\} \in \Gamma}{\text{minimise}} \quad a_3 & (6.14) \\
 & \text{avec : } l_3 = \{x_{13}; x_{14}; x_{15}; x_{16}; x_{17}; x_{18}\}, \\
 & \Gamma = \{x \in \mathbb{R}^5 \mid 0 \leq x_i \leq 10, i \in [13..18]\}
 \end{aligned}$$

L'optimiseur maître détermine la valeur optimale des variables partagées et couplées (*CopyIa_j*, avec $j \in [1..3]$) tandis que les optimiseurs disciplinaires fixent les variables de design locales. Parti d'un design infaisable, PMDO a convergé vers un optimum pour lequel le couplage interdisciplinaire est résolu avec la précision de satisfaction des contraintes de consistance, soit 1.E-7. La convergence a nécessité 179 itérations du *Coordinator* pour un total de près de 12000 évaluations de chacune des analyses. Les graphiques de convergence produits par iSIGHT (voir les figures 6.8 et 6.9) montrent que 135 itérations du *Coordinator* furent nécessaires avant obtention du premier design faisable.

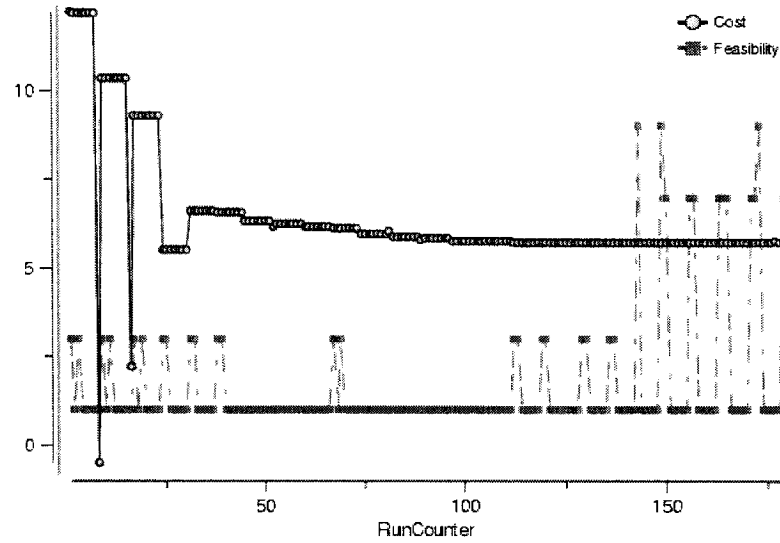


FIGURE 6.8: Progression de la convergence DO

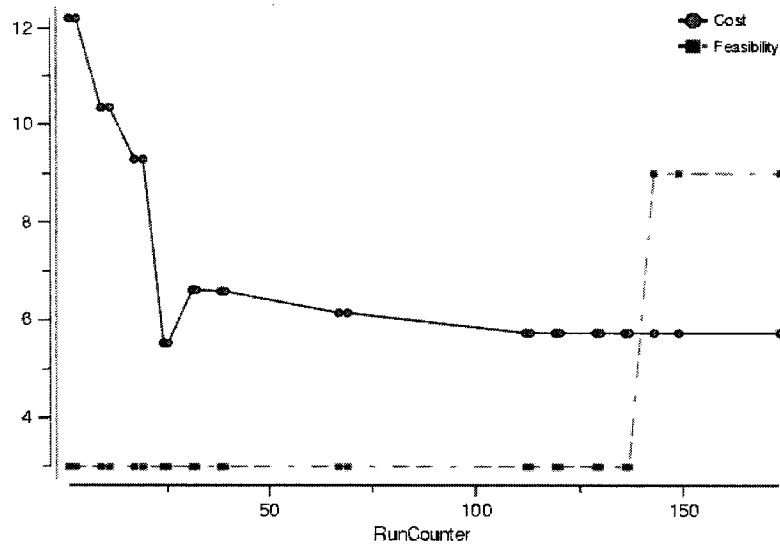


FIGURE 6.9: Améliorations de la convergence DO

6.5.3 Analyse

6.5.3.1 Choix d'une méthode de décomposition

Les tests effectués avec l'exemple TP300 sont concluants car, à chaque fois, l'optimisation a permis de converger vers un optimum très proche de celui attendu. Le tableau récapitulatif 6.2 présente le design optimum trouvé en fonction de la méthode de décomposition utilisée. Dans les 3 cas FIO, DAO et DO, le faible écart observé entre le résultat obtenu et l'optimum X_{sol} connu (indiqué en référence dans la quatrième colonne) est inversement relié au critère de résolution du couplage interdisciplinaire *CumulErreursCouplage*².

Plutôt que de faire ressortir une méthode de décomposition parmi les 3 essayées, ces essais illustrent que chacune a des spécificités la désignant pour certains problèmes. Pour cet exemple, FIO est moins efficace que DAO et DO pourtant cette méthode produit un grand nombre de designs faisables et sa convergence initiale est rapide. DAO et DO ont sensiblement les mêmes performances de résolution du couplage et convergent vers des optimums proches mais pour un temps de calcul variant de 2900 (DAO) à 4700 (DO) secondes. Partant du même design initial pour aboutir à des solutions comparables, les 3 processus d'optimisation testés varient donc grandement dans leur progression en fonction de la décomposition et des algorithmes de résolution (ces derniers sont, rappelons-le, choisis par iSIGHT en fonction du problème posé) utilisés. Le tableau 6.3 résume les coûts du processus d'optimisation en terme de temps de calcul et de nombres d'évaluations de chacune des analyses.

²La variable *CumulErreursCouplage* est la somme des carrés de l'erreur de couplage pour l'ensemble des variables couplées du problème considéré

TABLEAU 6.2: Solutions du cas test TP300 pour différentes méthodes de décomposition

	FIO ¹	DAO	DO	Référence ²
Itérations pour convergence	2272	2922	173	#
x12	0,662706505	0,739764222	0,740161115	0,74074
x10	1,372310845	1,614369661	1,615544321	1,61592
x7	2,839054569	2,927600541	2,928374309	2,92868
x11	1,003278193	1,177016731	1,177891641	1,17833
x6	3,357151672	3,365875131	3,366187786	3,36627
x8	2,316603767	2,489620654	2,490717915	2,49109
x9	1,812866983	2,051890857	2,053112631	2,05351
x4	5,238755995	5,360779268	5,363970251	5,36627
x2	7,168783896	7,357834475	7,361997056	7,36627
x1	8,158013346	8,357416313	8,361654375	8,36627
x3	6,196839898	6,358986769	6,362775359	6,36627
x5	4,295266748	4,363109826	4,365098151	4,36627
x14	0,455141319	0,527829739	0,528624445	0,529099
x17	0,149811359	0,211035451	0,211289254	0,21164
x13	0,540211716	0,633685545	0,634357392	0,634919
x15	0,3864225	0,422196136	0,422803163	0,423278
x16	0,284346072	0,316462392	0,317025582	0,31746
x18	0,168729436	0,105359825	0,105612584	0,10582
CopyIa3	5,676703052	5,576038652	5,575637277	5,57506
CopyIa2	-5,20113638	-5,264463148	-5,267098359	-5,26732
CopyIa1	5,311631571	5,42825128	5,431273183	5,43207
a3	5,676703052	5,576039039	5,575637201	5,57506
a2	-5,20113638	-5,264463264	-5,26709892	-5,26732
a1	5,311631571	5,428251898	5,431273331	5,43207
F2	-3,58989336	-3,632453783	-3,634045685	-3,63418
F1	5,565847319	5,59738293	5,598849251	5,59924
F3	5,645131012	5,530927987	5,530424196	5,52969
ConstraintIa3 ³	#	3,87E-07	-7,57E-08	#
ConstraintIa2	#	-1,16E-07	-5,61E-07	#
ConstraintIa1	#	6,18E-07	1,47702E-07	#
CumulErreursCouplage ⁴	5,54E-06	5,45532E-13	3,4185E-13	0,3907,E-13
Feasibility	9	9	9	#
Cost	5,787198243	5,739827673	5,739811613	5,739806986

¹ $\epsilon = 1.E-5$ ² Ceci constitue le meilleur optimum connu X_{sol} , d'après les données disponibles, pour la résolution du cas test TP300 avec $\beta = 0.1$.³ Contrainte de consistance $ConstraintIai = ai - CopyIai$ ⁴ Somme du carré de l'erreur de résolution du couplage pour l'ensemble du MDA, $CumulErreursCouplage = \sum_{i=1}^3 (ConstraintIai^2)$

TABLEAU 6.3: Récapitulatif des résultats pour le cas test TP300

Méthode de décomposition	FIO ¹	DAO	DO
Itérations Coordinator	2344	2931	179
Itérations Discipline1	4949	2931	11462
Itérations Discipline2	4949	2931	10992
Itérations Discipline3	4949	2931	11895
Fonction coût	5,788830	5,739829	5,739812
Δ_{moy} ²	1.E-3	1.E-7	1.E-7
Temps réel d'exécution (s)	5492	2901	4716
Temps machine iSIGHT (s)	20	13	12

¹ $\epsilon = 1.E-5$

² valeur moyenne de l'erreur de résolution du couplage pour chaque variable couplée

6.5.3.2 Fractionnement du temps de calcul

Au cours de ces essais, la résolution répétée (jusqu'à 5 fois dans certains cas) de chaque problème a toujours conduit au même résultat et requis un nombre toujours identique d'évaluations de chaque discipline. Le processus de résolution est donc entièrement reproductible (tout au moins pour les algorithmes d'optimisation utilisés). Par contre, le temps d'exécution de PMDO (temps écoulé entre le lancement de l'exécution et la convergence) varie en fonction de la puissance des ordinateurs disponibles ainsi que de l'encombrement du réseau.

En analysant la séquence d'exécution pour chaque décomposition puis en dénombrant à chaque fois l'information partagée d'un composant vers un autre, il fut possible d'évaluer le volume des dialogues (TCP/IP) entre iSIGHT et le JDM. Le nombre de messages échangés au cours de l'optimisation varie de 296000 pour DAO à près le 500000 pour DO (chaque message est une chaîne de caractères constituée

de 3 à 5 mots clefs concaténés). De même, le nombre d'évaluations disciplinaires (toutes disciplines confondues) varie de 8793 pour DAO à 34368 pour DO (voir le tableau 6.4 pour les détails). La comparaison de ces mesures à la valeur moyenne des temps d'exécution enregistrés fait ressortir une relation quasiment linéaire entre le nombre des messages échangés et la durée de calcul total (voir la figure 6.10). Dans le but de confirmer cette observation, une modification temporaire a été apportée au code de PMDO pour doubler le nombre des messages transitant sur le réseau : chaque requête du composant d'optimisation est adressée deux fois de suite au JDM qui, en retour, envoie 2 fois sa réponse. Toutes les autres étapes du processus de résolution restant par ailleurs inchangées, le nombre d'évaluations n'est pas affecté par cette modification. Ce code modifié a été utilisé pour résoudre, une fois de plus, le problème test TP300 par la méthode DAO. Le temps de calcul mesuré (voir la colonne « DAO double messages » du tableau 6.4) étant très peu inférieur au double du temps DAO de référence, cet essai confirme que le temps d'exécution de PMDO est conditionné avant tout par le nombre des messages échangés sur le réseau.

Bien entendu, le temps d'exécution de PMDO, de l'ordre de l'heure, n'est pas comparable à celui nécessaire pour résoudre le problème test par d'autres moyens. Certains programmes (tel *ampl* par exemple), conçus spécifiquement pour l'optimisation de problèmes définis par des fonctions analytiques, sont capables de converger en un peu plus d'une seconde vers la solution de référence X_{sol} . Toutefois, il faut préciser que ces programmes sont généralement exécutés en local, sans interface graphique, et évaluent directement la valeur des dérivées partielles de la fonction coût (ce qui permet de s'affranchir des estimations par la méthodes des différences finies et donc de converger plus rapidement et précisément en utilisant les algorithmes d'optimisation à gradient). Au contraire, PMDO est conçu pour sa flexibilité et pour pouvoir s'accommoder non seulement de ces mêmes fonctions

TABLEAU 6.4: Résumé du nombre des évaluations disciplinaires et des messages échangés

Décomposition	DAO	DAO doubles messages	FIO ¹	DO
Temps réel moyen (s) ²	2901	6382	5530	4716
Messages échangés	296031	592062	499825	421187
Evaluations disciplinaires	8793	8793	14847	34368

¹ $\epsilon = 1.E-5$

² Moyenne des résultats de 2 essais.

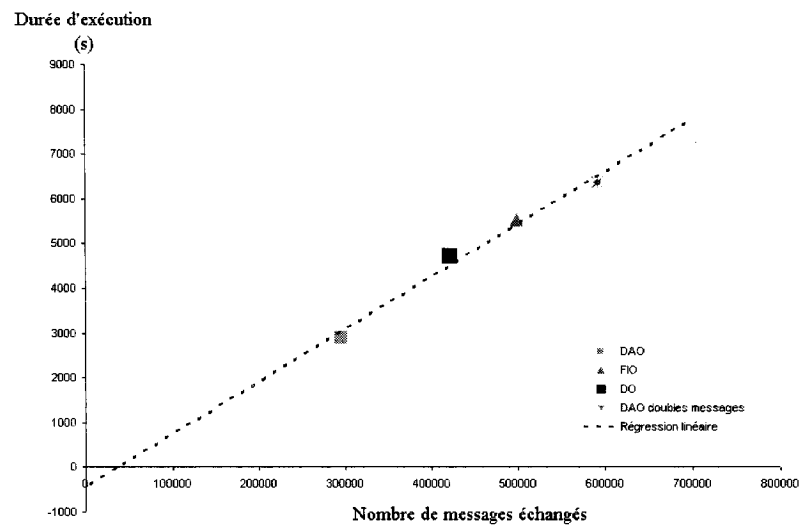


FIGURE 6.10: Temps d'exécution de PMDO en fonction du nombre de messages TCP/IP échangés

d'analyses analytiques mais aussi d'une grande variété de programmes d'analyses. Prédominants pour ce cas test simple, les coûts de calcul inhérents à la plate-forme PMDO seront relativement moindres lorsque les outils de design préliminaire seront intégrés.

6.5.3.3 Conclusions

En conclusion, ces tests ont permis de vérifier, sur différents systèmes d'exploitation et pour un exemple simple, que PMDO correspond aux principaux requis en terme de flexibilité, de configuration automatisée et d'optimisation. Ils illustrent également l'importance du choix de la méthode de décomposition pour résoudre un problème de MDO.

De plus, la vitesse d'exécution du prototype apparaît très dépendante de l'échange de messages entre iSIGHT et le JDM. Le mécanisme de communication intercomposants mis en place, basé le principe du serveur dialoguant avec plusieurs clients, est tout d'abord imposé par le composant d'optimisation iSIGHT. Ce choix ne sera vraisemblablement pas remis en cause (les alternatives compatibles avec iSIGHT ne sont pas nombreuses), néanmoins, certaines modifications (relativement aisées à mettre en place) des mécanismes de communication interdisciplinaire permettraient de diminuer significativement le temps de résolution des problèmes. Dans le cas du TP300, une modification du protocole de communication entre iSIGHT et le JDM permettrait de diviser par 10 environ le nombre des messages et par 3 le nombre de mots échangés sur le réseau.

CONCLUSION

Symboles de l'excellence technologique d'une époque, les réalisations de l'industrie aéronautique sont le fruit de beaucoup de travail, d'expertise ingénieur et de la mobilisation de moyens importants mis en œuvre dans un contexte où les contraintes de tout ordre sont omniprésentes. À ce titre, la conception des turbomachines est un parfait exemple de processus d'optimisation dans lequel l'analyse coordonnée de centaines de spécialistes doit mener au meilleur produit possible compte tenu des ressources disponibles. Depuis 20 ans, l'extraordinaire développement des ordinateurs permet d'augmenter les possibilités de design et de raccourcir les cycles d'analyse. L'enjeu est maintenant la mise à disposition, pour les ingénieurs, de méthodes de conception adaptées à ces nouveaux outils informatiques.

Dans ce contexte, le premier objectif de recherche de cette maîtrise, financée par le CRIAQ (voir §1.2) et par ses partenaires industriels, fut la définition d'un nouvel outil informatique baptisé PMDO (Preliminary Multidisciplinary Design Optimization) et destiné à accélérer le design conceptuel des turbomachines de Pratt & Whitney Canada. Dans un deuxième temps, la réalisation d'un prototype, puis sa validation, en ont constitué les réalisations les plus importantes.

La définition des requis associés à l'outil PMDO a rapidement montré que l'un des enjeux majeurs était la conception d'une plate-forme d'intégration flexible, transportable et modulaire, mais aussi capable d'automatiser au maximum les étapes de configuration automatique pour diverses méthodes de décomposition. Cette plate-forme devait également être conçue dans la perspective de constituer ensuite un outil de design d'utilisation aisée et intuitive. Dans le prolongement de la phase de conception, la programmation de certains modules de PMDO a été confiée à d'autres étudiants dans le cadre de leur projet de fin d'études. Le prototype ainsi

constitué est une contribution originale à la recherche sur la configuration automatisée de plates-formes de MDO flexibles. Il a fait l'objet de tests dont les résultats ont permis de valider la conception de PMDO, son initialisation ainsi que les mécanismes d'optimisation. Des travaux sont actuellement en cours pour poursuivre le développement du prototype installé chez Pratt & Whitney Canada.

Pour les étudiants qui poursuivront les recherches, les perspectives de développement et d'améliorations sont nombreuses. Tout d'abord, les essais effectués ont montré que, excluant les coûts d'analyse, le facteur limitant la vitesse d'exécution de PMDO est l'échange de messages TCP/IP entre ses composants. Une redéfinition des protocoles permettra d'accélérer les optimisations. Également dans le but d'améliorer les performances enregistrées, l'inclusion au JDM d'un module de calcul de gradients facilitera l'utilisation des algorithmes à gradient. De même, l'architecture clients/serveur de PMDO, associée aux méthodes de décomposition multi-niveaux (telle que DO), permettra de distribuer l'optimisation sur plusieurs ordinateurs travaillant simultanément pour diminuer encore le temps de calcul.

Le développement ultérieur des mécanismes de configuration automatisée permettra à PMDO de proposer une formulation complète et fonctionnelle d'un problème donné. L'utilisateur pourra ensuite l'adapter plus finement par rapport à l'objectif du design. La définition d'un algorithme de conseil capable de suggérer (par exemple dès l'interface PMDO_GUI) une méthode de décomposition pertinente (en fonction du nombre des variables de design et des analyses, du couplage interdisciplinaire et de la formulation du problème d'optimisation) est également un objectif de recherche ambitieux pour le futur.

Enfin, l'intégration en cours des feuilles de calcul Excel de Pratt & Whitney Canada permettra de calibrer le prototype PMDO et d'effectuer les premiers essais appliqués au processus de design conceptuel des moteurs. Par la suite, l'ajout de

nouveaux modules d'interface (édition automatisée de fichiers d'entrées et de sorties) au JDM étendra les possibilités d'intégration à des outils d'analyse d'un autre type (des exécutables par exemple).

En conclusion, l'expérience apportée par PMDO ouvre naturellement la voie aux futurs outils de résolution de problèmes MDO plus dispendieux (constitués par l'intégration d'outils d'analyse plus complexes). Les mécanismes d'intégration d'analyses, de gestion des variables de design, de configuration automatisée et d'optimisation validés avec PMDO pourront demain être appliqués au design détaillé de sous-ensembles de moteurs.

BIBLIOGRAPHIE

- ALEXANDROV, N. M. ET LEWIS, R. M. (1999). Comparative properties of collaborative optimization and other approaches to MDO. Rapport technique 99-24, ICASE — NASA Langley Research Center.
- ALEXANDROV, N. M. ET LEWIS, R. M. (2000a). Analytical and computational aspects of collaborative optimization. Rapport technique NASA/TM-2000-210104, NASA Langley Research Center.
- ALEXANDROV, N. M. ET LEWIS, R. M. (2000b). Analytical and computational properties of distributed approaches to MDO. *8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. No. AIAA-2000-4718, 1-12.
- ALEXANDROV, N. M. ET LEWIS, R. M. (2004). Reconfigurability in MDO problem synthesis, part 1. *10th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. Albany, New York, no. 2004-4307.
- BAILEY, M. W. (1999). Integrated multidisciplinary design. *ISABE paper No. 99-7174*.
- BRAUN, R. D. (1996). *Collaborative Optimization: An Architecture for Large-Scale Distributed Design*. Thèse de doctorat, Stanford University.
- CORDIER, M.-O. ET ROUSSET, M.-C. (2005). Systèmes Experts. Encyclopedia Universalis - <http://www.universalis-edu.com/>.
- CRAMER, E. J., DENNIS, JR., J. E., FRANK, P. D., LEWIS, R. M. ET SHUBIN, G. R. (1993). Problem formulation for multidisciplinary optimization. *4*, 754-776.

- DE MIGUEL, A.-V. (2001). *Two Decomposition Algorithms for Nonconvex Optimization Problems with Global Variables*. Thèse de doctorat, Stanford University.
- FORTIN, A. (1995). *Analyse Numérique pour Ingénieurs*.
- KROO, I. (1997). MDO for large-scale design. *Multidisciplinary Design Optimization — State-of-the-art. Proceedings of the ICASE/NASA Langley Workshop on Multidisciplinary Design Optimization, Hampton, VA, March 1995*. Society for Industrial and Applied Mathematics, 22–44.
- LYTLE, J. K. (1999). The numerical propulsion system simulation: A multidisciplinary design system for aerospace vehicles. ISABE, éditeur, *ISABE Paper No ISABE 99-7111*. ISABE, ISABE, vol. No ISABE 99-7111.
- MIGDALAS, A., PARDALOS, P. M. ET VARBRAND, P. (1998). *Multilevel optimization: Algorithms and applications*. Kluwer. Kluwer.
- MOUSTAPHA, H. (2003). *Small Gas Turbine Technology - Evolution and Challenges*. Pratt & Whitney Canada.
- NEOS (2003). Optimization Technology Center - NEOS Guide Optimization Tree. <http://www-fp.mcs.anl.gov/otc/Guide/OptWeb/>.
- PANCHENKO, Y., PATEL, K., MOUSTAPHA, H., DOWHAN, M. J., MAH, S. ET HALL, D. (2002). Preliminary multi-disciplinary optimization in turbomachinery design. *Proceedings of DETC/CIE Conferences*. Montréal, QC, no. ISD-34436.
- PARK, H.-W. ET CHOI, D.-H. (2003). An adaptive decomposition and scheduling of multidisciplinary design problems. ISSMO and University of Pavia (Italy), Italian Polytechnic Press, Milano, 33–34.

- PEREZ, R. E., LIU, H. H. T. ET BEHDINAN, K. (2004). Evaluation of multidisciplinary optimization approaches for aircraft conceptual design. No. AIAA-2004-4537, 1–11.
- PRADO, P., PANCHENKO, Y., MAH, S. ET HALL, D. (2003). PMDO: Project summary. Rapport technique, École Polytechnique de Montréal.
- ROGERS, J. L. (1992). DeMaid/GA an enhanced design manager's aid for intelligent decomposition. *4th AIAA/AF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*. Independence, Ohio, no. 92–4809.
- ROGERS, J. L. ET BLOEBAUM, C. L. (1994). Ordering design tasks based on coupling strengths. *Fifth AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. Panama City, Florida, no. 94–4326.
- RYAN, R., BLAIR, J., TOWNSEND, J. ET VERDERAIME, V. (1996). Working on the boundaries: Philosophies and practice of the design process. NASA, NASA, vol. 3642.
- SCHITTKOWSKI, K. (1987). More test examples for nonlinear programming codes. *Lecture Notes in Economics and Mathematical Systems*, 282.
- TRIBES, C., DUBÉ, J.-F. ET TRÉPANIÉ, J.-Y. (2004). Decomposition of multidisciplinary optimization problem: Formulations and application to a simplified wing design. *10th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. No. AIAA-2004-4544, 1–21.
- TURCOTTE, J. (2004). *Intération et optimisation multidisciplinaires d'une turbine à gaz au stade préliminaire de conception*. Mémoire de maîtrise, École Polytechnique de Montréal.

VANDERPLAATS, G. N. (1999). *Numerical Optimization Techniques for Engineering Design*. Vanderplaats Research & Development, Inc.