

Calcul de trajectoires optimales de lanceurs spatiaux réutilisables par une méthode de point intérieur

Julien Laurent-Varin

► To cite this version:

Julien Laurent-Varin. Calcul de trajectoires optimales de lanceurs spatiaux réutilisables par une méthode de point intérieur. Optimisation et contrôle [math.OC]. Ecole doctorale de polytechnique, 2005. Français. <tel-01579175>

HAL Id: tel-01579175

<https://tel.archives-ouvertes.fr/tel-01579175>

Submitted on 30 Aug 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

présentée à

L'ÉCOLE POLYTECHNIQUE

pour obtenir le titre de

DOCTEUR

Spécialité : Mathématiques Appliquées

par

Julien LAURENT-VARIN

Calcul de trajectoires optimales de lanceurs spatiaux réutilisables par une méthode de point intérieur

Soutenue le 23 Novembre 2005 devant le Jury composé de

J. Frédéric BONNANS

Directeur de thèse

Nicolas BÉREND

Examineur

Philippe CHARTIER

Rapporteur

Pierre-Louis LIONS

Examineur

Helmut MAURER

Rapporteur

Isabelle RONGIER

Examineur

Remerciements

Je tiens à remercier toutes les personnes m'ayant encadré et soutenu pendant ces travaux de thèse :

En premier lieu, j'adresse mes sincères remerciements à mon directeur de thèse Frédéric Bonnans qui m'a accepté en tant que thésard et m'a fait partager son expérience de la recherche durant ces trois ans,

je remercie mon encadrant de l'ONERA, Nicolas Bérend qui a partagé son bureau pour répondre à toutes mes questions,

je remercie Christophe Talbot du CNES pour sa participation pendant ces trois ans,

je remercie les rapporteurs qui ont fait des remarques très pertinentes et ont contribué à l'amélioration notable de ce document,

je remercie toutes les personnes ayant relu et corrigé ce manuscrit, et plus particulièrement Elisabeth Ottenwaelter,

je remercie enfin toutes les personnes qui m'ont soutenu et poussé dans mes études, mes parents, mes soeurs et toute ma famille,

enfin je remercie celle qui a dû supporter avec patience tous les rebondissements de ces travaux, ma chère fiancée Emilie.

Je vous remercie tous profondément et vous dédie ce document.

Julien LAURENT-VARIN

Table des matières

I	Introduction	5
II	Tour d’horizon en commande optimale et modèles physiques	9
1	Algorithmes de commande optimale	11
1.1	Panorama en commande optimale	11
1.1.1	Méthodes locales	12
1.1.2	Méthodes globales	15
1.1.3	Méthodes totalement analytiques	16
1.2	Méthodes de point intérieur	16
1.2.1	Optimisation linéaire	16
1.2.2	Optimisation non linéaire	16
1.2.3	Commande optimale	17
2	Définition du problème et modèles aéronautiques	19
2.1	Position du problème	20
2.2	Vecteur état	22
2.3	Angles d’attitude du véhicule (commande)	24
2.4	Dynamique	25
2.4.1	Modèles de gravité	25
2.4.2	Forces aérodynamiques	26
2.4.3	Forces d’entraînement et de Coriolis	30
2.4.4	Principe fondamental de la dynamique	31
2.4.5	Équation du mouvement en atmosphère	31
2.5	Contraintes sur le système	32
2.5.1	Limites thermiques	33
2.5.2	Limites d’équilibre	34
2.5.3	Limites en facteur de charge	35
2.5.4	Limites en pression dynamique	36
2.5.5	Couloir d’incidence	36

2.5.6	Couloir de rentrée	36
2.6	Jonctions entre les arcs	37
2.7	Exemple de concepts multiarcs	38
2.7.1	Monoétage de type SOH	38
2.7.2	Bi-étage en configuration siamoise	38
III	La discrétisation de Runge-Kutta	43
3	Méthode de Runge-Kutta en commande optimale	45
3.1	Introduction	45
3.2	La résolution numérique d'EDO et de systèmes partitionnés .	46
3.2.1	Définitions préliminaires	46
3.2.2	Méthodes de collocation	46
3.2.3	Méthodes de Runge-Kutta partitionnées	48
3.3	Condition d'ordre pour les schémas de Runge-Kutta	48
3.3.1	Motivation	48
3.3.2	Définitions formelles à propos des arbres	49
3.3.3	Séries B	50
3.3.4	Série B de la solution exacte d'une EDO	51
3.3.5	Série B de la solution numérique de Runge-Kutta . . .	51
3.4	Conditions d'ordre	51
3.4.1	En terme d'arbre	51
3.4.2	Méthode pratique pour le calcul des conditions d'ordre	51
3.4.3	Méthode de collocation	52
3.5	Précision de résolution et condition d'ordre pour les méthodes de Runge-Kutta partitionnées	53
3.5.1	Motivation	53
3.5.2	Définition formelle des arbres bicolores	54
3.5.3	Séries P	55
3.5.4	Série P de la solution exacte d'un système partitionné .	55
3.5.5	Série P de la solution numérique de Runge-Kutta par- titionnée	55
3.6	Conditions d'ordre pour les schémas partitionnées	56
3.6.1	En terme d'arbre	56
3.6.2	Méthode pratique pour le calcul des conditions d'ordre	56
3.7	Les méthodes de Runge-Kutta symplectique	56
4	Computation of order conditions for symplectic partitioned Runge-Kutta schemes with application to optimal control	59
4.1	Introduction	59

4.2	Discretization of unconstrained optimal control problems . . .	61
4.3	Order conditions for partitioned Runge-Kutta schemes	63
4.4	Calculus on oriented free trees	66
4.5	Implementation and computational results	69
4.5.1	Implementation	69
4.5.2	Computational result	70
IV	Méthode de point intérieur	87
5	An Interior-Point Approach to Trajectory Optimization	89
5.1	Introduction	89
5.2	Unconstrained problems: error estimates	92
5.3	Mesh Refinement	94
5.4	Linear algebra	97
5.5	Interior-point algorithms, constrained problems	98
5.5.1	Interior point	98
5.5.2	The Dogleg procedure	100
5.6	Application to Goddard's problem	100
5.7	Fuller's problem	101
5.8	Atmospheric Reentry	109
5.8.1	Model	109
5.8.2	Initialization procedure	109
5.8.3	Update of the penalty parameter	110
5.8.4	Numerical results for cross range maximization	110
5.8.5	Numerical results, for range maximization	112
5.9	Conclusion and perspectives	116
6	Compléments numériques monoarc	119
6.1	Retour de booster Everest	119
6.2	Rentrée atmosphérique SOH	122
V	Calcul de trajectoire optimale multiarc	125
7	Optimisation multiarc	127
7.1	Graphe de scénario, dynamique et critère	127
7.1.1	Motivations	127
7.1.2	Définitions préliminaires	128
7.1.3	Variables du problème	128
7.1.4	Conditions sur les arcs du graphe	129

7.1.5	Conditions sur les sommets	130
7.1.6	Lien entre variables d'arc et de sommet	131
7.1.7	Critère d'optimisation	131
7.1.8	Problème de commande optimale sur un graphe de scénario	131
7.2	Conditions d'optimalité	132
7.2.1	Cas sans contraintes d'inégalité	132
7.2.2	Cas avec contraintes d'inégalité	133
7.3	Résolution numérique du problème	134
7.3.1	Conditions d'optimalité discrètes	134
7.4	Algèbre linéaire	135
7.4.1	Algèbre linéaire récursive	135
7.4.2	Type inductif de matrice	136
7.4.3	Description détaillée	137
7.4.4	Factorisation et inversion des matrices axiomes et 4 blocs	139
7.4.5	Factorisation et inversion des matrices "Graphe"	140
7.4.6	Résolution du système réduit	142
7.5	Applications	143
7.5.1	Cas académique	143
7.5.2	Bi-étage en configuration siamoise	144
A	Détail de modélisation	153
A.1	L'atmosphère	153
A.1.1	Altitude <i>géocentrique</i> , <i>géodésique</i> et <i>géopotentielle</i> . . .	153
A.1.2	Modélisation de l' <i>atmosphère</i>	154
B	Détail du système linéaire	159
B.1	Résolution et algèbre-linéaire multi-arc	159
B.1.1	Détail des conditions d'optimalité discrétisées	159
B.1.2	Jacobienne du système	162
B.1.3	Détails de la jacobienne du système	164
C	Participation à des conférences et publications	171

Partie I

Introduction

Depuis le premier satellite artificiel Spoutnik, lancé par les Russes en 1957, les techniques spatiales ont fait de grands progrès. Et pourtant, la technique repose toujours sur la propulsion fusée chimique et le lanceur dit “consommable”, qui est détruit dans l’opération de lancement. Avec leur navette, les États Unis d’Amérique ont essayé de sortir de ce cadre en employant une solution semi-réutilisable. Mais la réussite ne peut être qualifiée de totale. Elle se heurte à la difficulté d’assurer la fiabilité du système et représente des coûts élevés, contrairement à ce qui avait été prévu.

Aujourd’hui, le CNES et l’ONERA veulent se doter d’outils de calcul de trajectoires pour les études précédant les choix de concepts. Dans ce cadre, cette thèse a pour objectif de mettre au point une méthode fiable et robuste pour calculer les trajectoires optimales de lanceurs totalement réutilisables. Les calculs portent donc aussi bien sur les trajectoires de montée (propulsées) que de rentrée (planeur hypersonique). La méthode explorée a été celle de point intérieur, associée à des modules spécifiques d’algèbre linéaire et à une stratégie de raffinement de discrétisation.

La thèse se découpe en quatre grandes parties.

Une première partie est découpée en deux chapitres présentant les méthodes actuellement existantes en commande optimale et en optimisation, ainsi que l’utilisation des méthodes dites de points intérieurs d’une part, et un second chapitre décrivant le problème à résoudre et les modélisations physiques effectuées.

La seconde partie est focalisée sur la discrétisation de Runge-Kutta que la méthode utilise. Le Chapitre 3 donne des rappels sur ces méthodes, et en particulier, la manière de construire les conditions d’ordre pour la classe de méthode “classique” et “partitionnée”. Le Chapitre 4, montre qu’en commande optimale, la méthode de Runge-Kutta doit être “symplectique” et décrit un calcul pour obtenir les conditions d’ordre de cette classe de méthodes.

La troisième partie décrit en détail la méthode de résolution numérique d’un point de vue mono-arc. Elle est composée de deux chapitres. Le Chapitre 5 présente les trois piliers de la méthode : l’aspect point intérieur, l’algèbre linéaire spécifique, et le raffinement optimal. En effet, la formalisation primale-duale perturbée des conditions d’optimalité provenant de la méthode de point intérieur, nous conduit à un système linéarisé bande. Pour celui-ci la factorisation QR est particulièrement avantageuse car en sus d’être rapide, elle ne dégrade pas le conditionnement du système lors de la résolution par factorisation orthogonale. De surcroît, pour un paramètre de point

intérieur fixé, une étude de l'erreur de discrétisation commise peut être effectuée. Ainsi une politique de raffinement est mise au point. Il s'agit de résoudre un problème d'optimisation en nombres entiers où nous minimisons le nombre de points à ajouter lors du raffinement sous la contrainte d'atteindre une erreur donnée. Finalement, le chapitre présente une application aux problèmes de Goddard, de Füller ainsi qu'à un ensemble de problèmes de rentrée atmosphérique. Le Chapitre 6 présente les résultats obtenus par l'application de la méthode à deux trajectoires de concepts de véhicules ré-utilisables, qui sont le retour booster d'un concept de bi-étage, et la rentrée atmosphérique du véhicule monoétage SOH.

La quatrième partie se concentre sur l'aspect multi-arcs, en décrivant le formalisme choisi pour décrire ce type de problème, puis calcule les conditions d'optimalité et décrit l'algèbre linéaire inductive associée à la résolution, et enfin applique l'outil à un cas simplifié d'un lanceur bi-étage en configuration siamoise inspiré du concept FSSC16-FR des études FESTIP.

L'annexe contient des compléments, en particulier sur la modélisation physique et sur le détail de la jacobienne des conditions d'optimalité.

Partie II

Tour d'horizon en commande optimale et modèles physiques

Chapitre 1

Algorithmes de commande optimale

Ce chapitre présente un panorama des méthodes de commande optimale ainsi que des méthodes de points intérieurs.

1.1 Panorama en commande optimale

Cette section est destinée à donner un aperçu, non exhaustif, des grandes classes de méthodes couramment utilisées en commande optimale. Nous décrirons leur principe, ainsi que les avantages et les inconvénients qu'elles apportent, et nous positionnerons la méthode que nous présenterons dans la suite de la thèse par rapport à ces familles de méthodes. Nous pouvons parler de deux classes principales de méthodes : globales, et locales. Les méthodes globales sont celles qui garantissent l'obtention de l'“optimum optimorum”. Le problème de ces méthodes est leur lourdeur et de ce fait leur limitation. Les méthodes locales sont des méthodes qui exhibent une solution locale du problème. Dans certains cas ce genre de méthode est suffisant, en particulier s'il n'existe qu'un seul optimum grâce à des propriétés de convexité par exemple. En aérospatiale la majorité des méthodes utilisées est locale. Notre méthode cherche à résoudre les conditions nécessaires d'optimalité locale.

Dans ce chapitre, nous considérerons le problème de commande optimale consistant à trouver la fonction du temps $u^*(t)$ minimisant un critère dépendant de l'état. Pour fixer les notations, nous choisirons un problème sous la

forme :

$$\begin{cases} \min_u \Phi(y(T)); \\ \dot{y}(t) = f(y(t), u(t)), & t \in [0, T]; & y(0) = y^0, \\ g(y(t), u(t)) \geq 0. \end{cases} \quad (P)$$

L'état est le vecteur y , la commande u , Φ sera le coût, et g est le vecteur des contraintes courantes sur l'état et la commande. Nous choisissons ce formalisme pour pouvoir expliquer chaque méthode par la suite.

1.1.1 Méthodes locales

Ces méthodes que nous nommons “locales” sont les méthodes qui sont actuellement les plus utilisées et risquent de le rester encore pour un certain temps, les méthodes globales étant beaucoup trop coûteuses pour des systèmes de dimension 6 ou plus. Parmi ces méthodes, nous pouvons distinguer trois familles assez différentes :

- Les *méthodes directes* qui cherchent à faire diminuer directement le coût tout en essayant de satisfaire les contraintes.
- Les *méthodes indirectes* qui cherchent à résoudre les conditions d'optimalité de Pontryaguine par la résolution d'un problème aux deux bouts.
- Les *méthodes de transcription* qui discrétisent le problème de commande optimale puis utilisent un solveur non-linéaire performant.

Regardons maintenant en détail chacune de ces méthodes afin de nous positionner.

Méthodes directes

Cette grande classe de méthodes a fait ses preuves et est actuellement toujours utilisée dans le domaine aérospatial. Le principe commun à cette classe de méthodes est le suivant :

- La commande est discrétisée et généralement considérée comme constante sur chaque intervalle de discrétisation,
- Cette commande étant donnée, un état associé peut alors être calculé par intégration de la dynamique.

- Vient ensuite l'étape de calcul de la direction de descente, dans le but d'améliorer le critère de performance tout en satisfaisant les contraintes, si cela n'est pas le cas.

Finalement les différences entre les méthodes viennent essentiellement du choix de cette direction de descente. Un atout majeur de la méthode, est la possibilité de prendre en compte un certain nombre de contraintes de natures variées.

Par contre, ces méthodes sont généralement lentes d'autant plus que l'horizon est grand ou que la discrétisation fine. En effet, elles possèdent généralement une certaine lenteur de convergence due au calcul d'une direction du premier ordre, et demandent un grand espace mémoire car la commande et/ou l'état sont stockés de façon discrétisée en mémoire.

Les implémentations du CNES et de l'ONERA, respectivement Orage et Flop, ont fait leurs preuves dans le domaine du calcul de trajectoire de rentrée et de montée pour Flop.

Méthodes indirectes

Contrairement aux méthodes de gradient, les méthodes indirectes, ne cherchent pas à faire décroître le coût à chaque itération, mais résolvent les conditions nécessaires d'optimalité dictées par le principe du maximum de Pontryaguine ([21], [20], [19]).

Dans le cas sans contraintes courantes il faut résoudre :

$$\begin{cases} \dot{y}(t) = H_p(y(t), u(t), p(t)), & t \in [0, T]; y(0) = y_0 \\ \dot{p}(t) = -H_y(y(t), u(t), p(t)), & t \in [0, T]; p(T) = \Phi'(y(T)) \\ u(t) = \operatorname{argmin}_w H(y(t), w, p(t)) & t \in [0, T], \end{cases} \quad (TPBVP)$$

où p est l'état adjoint et $H(y, u, p) := p \cdot f(y, u)$ le Hamiltonien du système. Le système $(TPBVP)$ est ce que l'on nomme un problème aux deux bouts (Two Points Boundary Value Problem). Les méthodes indirectes les plus connues sont les méthodes de tir simples et multiples. Les principales différences avec les méthodes directes, sont :

- L'intégration du système dynamique est abstraite, et est considérée comme une boîte noire.
- Les itérations conduisent à annuler la fonction de tir (définie par les conditions aux deux bouts du système) et non à faire décroître le coût.

Ces méthodes permettent d'obtenir des solutions avec une grande précision pour un coût raisonnable. En effet, tout en gardant en mémoire pour

chaque itération uniquement la valeur de l'état adjoint à l'instant initial, les inversions numériques mises en jeu ont une dimension très réduite (généralement il s'agit de la Hessienne de la fonction de tir). Au contraire les méthodes directes, pour lesquelles des fonctions discrétisées doivent rester stockées en mémoire, demandent des calculs numériques plus lourds car faisant intervenir un grand nombre de variables. Cette méthode demande cependant, une certaine régularité du Hamiltonien, et la connaissance de la structure des arcs de contraintes actives (ce qui est en général le cas pour les trajectoires de montée hors atmosphère). L'outil Optax du CNES utilise d'ailleurs ces propriétés.

Méthodes de transcription “directe”

Les méthodes de transcription “directe” sont des méthodes s'appuyant sur un solveur non-linéaire. Il s'agit en fait de discrétiser le problème de commande optimale afin d'obtenir un problème d'optimisation non-linéaire de grande taille. Cette nouvelle formulation du problème est ensuite résolue par un solveur NLP (Non Linear Programming). Cette méthode est utilisée en particulier dans l'outil SOCS ([9], [10], [8]). Cet outil de Boeing a récemment été intégré à l'outil ASTOS en temps que noyau de calcul [11].

Méthode de point intérieur

Celle-ci n'est pas à proprement parler une méthode de commande optimale, mais plutôt d'optimisation. D'un point de vue primal, elle revient à pénaliser la fonction de coût par une fonction barrière. C'est à dire que le problème (P) devient :

$$\begin{cases} \text{Min}_u \Phi(y(T)) + \int_0^T B_\varepsilon(g(u(t), y(t))); \\ \dot{y}(t) = f(u(t), y(t)), \quad t \in [0, T]; \quad y(0) = y^0. \end{cases} \quad (P_\varepsilon)$$

où le paramètre ε décroît vers 0, et où la fonction barrière B_ε peut être :

$$B_\varepsilon(y) = -\varepsilon \sum_i \log y_i$$

Il s'agit en fait d'approcher la solution du problème avec contrainte (P) par la solution des problèmes (P_ε) et de faire décroître progressivement ε . Les détails de cette méthode seront traités dans le chapitre 5.

Différentes pénalisations autres que logarithmiques peuvent être envisagées. La pénalisation exponentielle a été étudiée en particulier dans l'article [27].

1.1.2 Méthodes globales

Bien que satisfaisantes pour de nombreux problèmes, les méthodes locales ne garantissent pas l'extrémalité globale et les problèmes de trajectoire admettent fréquemment un grand nombre d'optima locaux. Si une méthode globale était disponible, elle permettrait de faire le tri parmi ces solutions. Nous allons brièvement présenter deux approches globales : le principe de programmation dynamique en commande optimale aussi nommé équation de Hamilton-Jacobi-Bellman, puis l'optimisation par intervalle.

Hamilton-Jacobi-Bellman (HJB)

La résolution du problème de commande optimale est traduite en l'intégration d'une équation aux dérivées partielles de la forme :

$$v_t(y, t) + \inf_u H(y, u, v_x(y, t)) = 0, \forall y \in \mathbb{R}^n, v(y, T) = \Phi'(y(T)). \quad (HJB)$$

La difficulté majeure est dans la discrétisation en variable d'espace, en effet une discrétisation en 100 pas pour un problème de dimension 6 donnerait un système de 10^{12} variables. C'est la "malédiction de la dimension". Par ailleurs, pour ne pas avoir à faire face à de telles tailles de matrices, des simplifications de modèle peuvent être envisagées et conduire à des dimensions d'état plus réduites, 3 par exemple. En effet cette dimension laisse envisager la possibilité d'utiliser la programmation dynamique. Finalement, à moins de trouver une discrétisation astucieuse ou d'avoir des calculateurs excessivement performants, cette approche risque de ne pas être utilisée pour l'aérospatiale.

Beaucoup de travaux sur cette méthode ont été et sont effectués, les travaux initiaux de Bellman [4] doivent être cités. En effet, c'est lui qui a introduit le principe de la programmation dynamique, puis ceux de P.-L Lions qui a introduit la notion de solution de viscosité dans [28]. Le livre de Barles [3] est une référence incontournable sur cette théorie; ainsi que les travaux numériques de Falcone, Giorgi et Loreti dans l'article [35] ou dans l'annexe du livre de Bardi Capuzzo-Dolcetta [2]. Voir aussi [18].

"Branch and Bound" et arithmétique d'intervalle

Encore au stade expérimental en optimisation classique, l'arithmétique d'intervalle cherche à résoudre les problèmes d'optimisation sur des intervalles et étend les opérations sur les nombres en des opérations sur les intervalles [60]. L'idée du "branch and bound" est de découper le domaine dans lequel on choisit d'optimiser et de supprimer certaines parties par des minoration et des majorations. L'arithmétique d'intervalle permet justement d'obtenir de

telles informations. Il reste cependant une difficulté : le traitement des contraintes.

1.1.3 Méthodes totalement analytiques

Dans le domaine des trajectoires de véhicules aérospatiaux, en particulier hors atmosphère, certains calculs peuvent être effectués analytiquement [14]. En effet, hors atmosphère, une expression explicite de la commande peut être obtenue en fonction de l'état adjoint par application du principe du maximum de Pontryaguine.

1.2 Méthodes de point intérieur

1.2.1 Optimisation linéaire

Sachant que la solution optimale d'un problème de programmation linéaire se situe sur un sommet du polyèdre des contraintes, l'algorithme du simplexe développé par Dantzig [29], qui a longtemps gardé la suprématie de l'optimisation linéaire, cherche une solution en longeant les arêtes du polyèdre, jusqu'à une solution. De nos jours, cet algorithme garde une place de choix pour des avantages de redémarrage après l'ajout de contraintes; cependant, les algorithmes intérieurs l'ont détrôné pour certains aspects, en particulier la complexité de résolution.

On sait que l'algorithme du simplexe a une complexité exponentielle (Klee et Minty [54]).

Le premier algorithme de complexité polynomiale pour la programmation linéaire dû à Hačijan [44] est à base d'ellipsoïdes. Bien que de complexité polynômiale cet algorithme n'est pas efficace en pratique. Par ailleurs Fiacco et McCormick [37] avaient développé des algorithmes basés sur la pénalisation du coût par une fonction barrière, qui étaient connus depuis les années 50 [38]. Finalement, Karmarkar [53] a décrit le premier algorithme de point intérieur efficace. Actuellement, les algorithmes de point intérieur sont très utilisés et font l'objet de nombreuses études.

1.2.2 Optimisation non linéaire

De nombreux travaux ont été effectués, nous citerons en particulier ceux de Wright [41] de Ulbrich [72] de Shanno et Vanderbei [68] ainsi que l'article de Bergounioux, Haddou, Hintermüller et Kunisch [6]. Une difficulté inhérente à cette méthode, reste l'initialisation. En effet, de même que l'algorithme de

Newton ou Quasi-Newton, cette méthode n'a une garantie de convergence que dans un voisinage de la solution. Des travaux sur cette initialisation ont été effectués, nous pourrions citer l'article de Sartenaer, Gretz et Nocedal [42].

D'un point de vue pratique, des logiciels ont été développés en s'appuyant sur des méthodes de points intérieurs tel que BARNLP et KNITRO.

1.2.3 Commande optimale

En commande optimale, cette méthode a été employée de différentes manières. Nous pouvons citer en premier lieu les travaux de Wright ([76]) et la thèse de Weiser ([75]) ainsi que l'article de Bonnans et Guilbaud [15]. Les autres utilisations d'algorithmes de point intérieur en commande optimale ont été effectuées dans le cadre de méthodes de transcription où le solveur NLP peut être de toute nature.

Chapitre 2

Définition du problème et modèles aéronautiques

Cette partie présente la modélisation physique du problème de trajectoire optimale que nous devrons traiter par la suite. Il s'agit d'une simple présentation des concepts physiques mis en jeu lors du lancement d'un véhicule spatial et de son retour sur Terre. Nous renvoyons pour des approfondissements à la référence suivante : [25].

Dans cette partie nous définirons le problème que nous souhaitons résoudre. Ensuite, nous rentrerons dans le détail de la modélisation des équations de la trajectoire, puis des contraintes à prendre en compte lors du calcul de trajectoire optimale.

Notations :

Définissons au préalable quelques notations que nous conserverons tout au long du manuscrit de thèse.

t	temps
r	rayon vecteur (centre Terre-engin)
λ	longitude
ϕ	latitude géocentrique
V	module de la vitesse dans le TGL (Trièdre Géographique Local)
γ	pente de la vitesse dans le TGL
ψ	azimut de la vitesse au sol (à partir du nord, positif vers l'est)
z	altitude géodésique/géométrique
H	altitude géopotentielle
M	nombre de Mach ($M = V/V_{son}$)
R_e	rayon équatorial terrestre (6 378 137 m)

μ_T	constante de gravitation de la Terre ($3.986\ 005\ 10^{13}\ m^3/s^2$)
J_2	terme du second ordre dans la décomposition en série du potentiel terrestre ($1.0826\ 10^{-3}$)
M_o	masse molaire moyenne de l'atmosphère terrestre
R	constante universelle des gaz
T	température
ω_T	vitesse angulaire de rotation de la Terre ($2\pi/86\ 164.09\ rad/s$)
D	traînée ($D = 1/2\rho V^2 S_{ref} C_X$)
L	portance ($L = 1/2\rho V^2 S_{ref} C_Z$)
ρ	masse volumique de l'atmosphère
S_{ref}	surface de référence aérodynamique du véhicule
C_X	coefficient de traînée
C_Z	coefficient de portance
f	finesse aérodynamique
α	incidence
β	dérapiage
μ	gîte aérodynamique
θ	assiette
p	pression statique
n	facteur de charge
Φ	flux thermique
Q	débit de masse

2.1 Position du problème

Depuis le premier vol spatial de Sputnik I en 1957, les motivations et la technologie pour accéder à l'espace puis à en revenir ont considérablement évolué, même si les principes de base sont restés les mêmes.

Hormis la navette spatiale américaine, les lanceurs actuels sont de type consommable, c'est à dire qu'aucune partie du lanceur n'est récupérée à l'issue de la mission de lancement. D'un point de vue de l'optimisation, les techniques actuelles pour ce type de trajectoire s'appuient sur une commande agissant essentiellement à travers l'orientation du vecteur de poussée. Par ailleurs, dans les études de concepts futurs, en cas de réutilisation d'un ou plusieurs étages, la mission complète du lanceur inclut une ou plusieurs phases de rentrée. Le calcul de trajectoires de rentrée atmosphérique peut s'avérer nécessaire aussi, pour les cas de sauvegarde de ces lanceurs réutilisables.

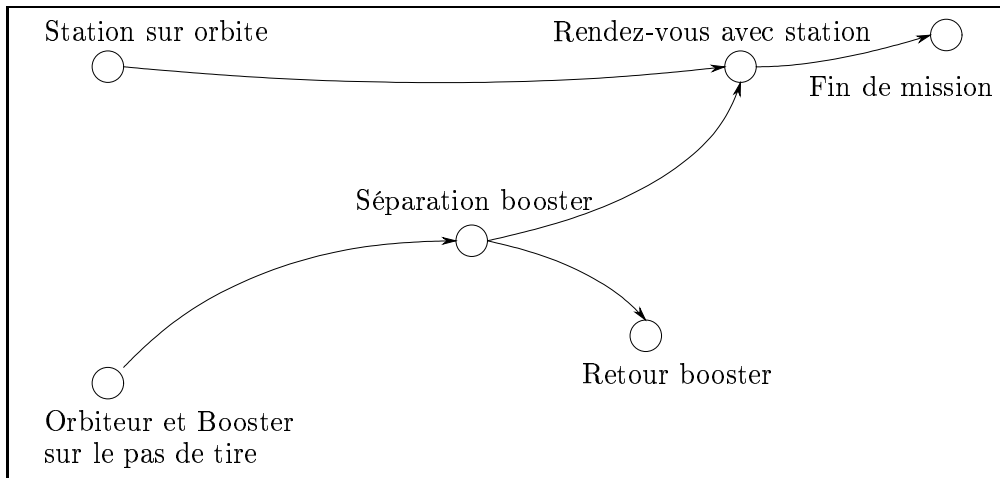
L'étude des lanceurs totalement ou partiellement réutilisables demande le calcul simultané de trajectoires optimales de montée et de rentrée; en par-

ticulier pour des lanceurs suborbitaux (SOH) ou avec récupération d'étages inférieurs. Ceci nous conduit à des *trajectoires multiarcs* arborescentes contrairement aux lanceurs consommables classiques qui eux ont des trajectoires multiarc séquentielles.

Finalement, ces types de concepts conduisent à une formulation très large du problème d'optimisation à résoudre. En effet, il faut résoudre pour une seule fonction de coût une série de problèmes de commande optimale combinés entre eux, et interdépendants, ce lien pouvant être séquentiel ou arborescent. Le problème à résoudre se pose comme la minimisation d'une fonction de coût dépendant des états du système et de leurs commandes, où les états suivent des lois dictées par leur dynamique, tout ceci sous contraintes courantes d'inégalité.

Cette thèse s'efforcera donc de mettre au point une méthode pour résoudre ce type de problème.

Exemple 1. Considérons par exemple la mission d'un lanceur bi-étage entièrement réutilisable avec rendez-vous du booster avec une station orbitale qui, dans cet exemple théorique pourrait être commandée. Le booster étant la partie du lanceur qui n'apporte une aide que lors de la phase de montée puis se sépare à mi-parcours pour retourner au sol, soit sur le site de lancement, soit sur un autre site et l'orbiteur étant le deuxième étage qui atteint l'orbite.



Pour chacune de ces trajectoires, il nous est nécessaire de faire un certain nombre de choix qui, par la suite, conditionneront la résolution. Ces choix sont les suivants :

- les coordonnées ainsi que les autres variables d'état, $(y_i, i$ étant l'indice de l'arc),
- les variables de commande, (u_i) ,

- les variables statiques (ne dépendant pas du temps), (π_i) ,
- les forces en présence qui nous dicteront la dynamique du système,

$$\dot{y}_i(t) = f_i(y_i(t), u_i(t), \pi_i, t),$$

- les contraintes courantes d'inégalité sur chaque arc,

$$a_i \leq g_i(y_i(t), u_i(t), \pi_i, t) \leq b_i.$$

Nous ne traiterons pas de contraintes d'égalité courantes qui n'ajoutent pas de difficultés essentielles.

- les conditions de jonction entre les arcs, (que nous expliciterons formellement dans le chapitre 7 à la page 127).

Pour chaque arc i , nous aurons un problème de commande de la forme :

$$\left\{ \begin{array}{l} \text{Min } \Phi_i^0(y_i(0)) + \Phi_i^T(y_i(T_i)) + \int_0^{T_i} \ell_i(y_i(t), u_i(t), \pi_i, t) dt; \\ \dot{y}_i(t) = f_i(u_i(t), y_i(t), \pi_i, t), \quad t \in [0, T_i]; \\ \Psi_i^0(y_i(0)) = 0, \quad \Psi_i^T(y_i(T_i)) = 0; \\ a_i \leq g_i(u_i(t), y_i(t), \pi_i, t) \leq b_i; \\ G(y_i(0), y_i(T_i), \pi_i, J) = 0 \end{array} \right. \quad (P_i)$$

où J représente d'autres variables $y_j(0), y_j(T_j)$ et π_j .

2.2 Vecteur état

Les points communs à tous ces types de trajectoire, sont :

- un état contenant les informations sur la position (3 données réelles dépendant du choix du repère),
- ainsi que les informations sur le vecteur vitesse (3 données supplémentaires elles aussi fonctions de choix).

D'autres informations telles que la masse de l'engin ou autre, seront ajoutées à l'état suivant qu'il s'agisse d'une ascension ou d'une rentrée. En ce qui concerne les coordonnées, définissons les repères possibles :

Les repères inertiels et géographiques locaux sont représentés dans les figures 2.1 et 2.2. Le *point vernal* est le point sur la sphère céleste formé par

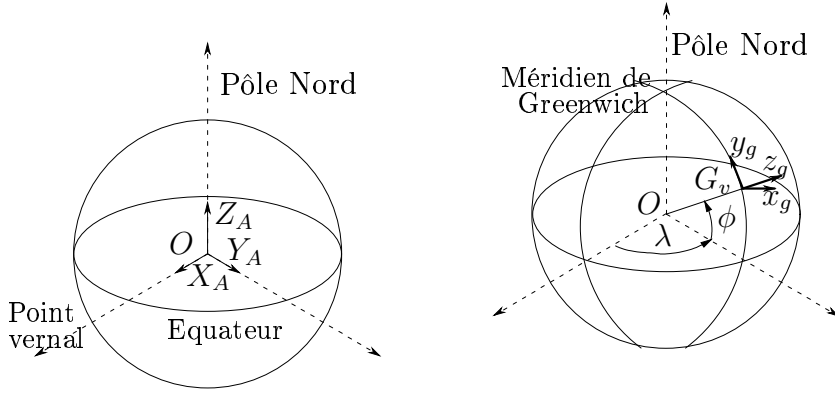


Figure 2.1: Référentiel inertiel

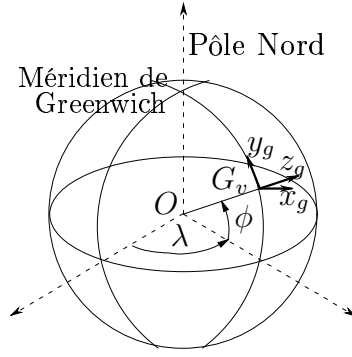


Figure 2.2: Référentiel géographique local

l'intersection entre l'écliptique et l'équateur céleste. À l'équinoxe de printemps, le soleil se trouve au point vernal. Dans le référentiel inertiel, les vecteurs de base du repère cartésien sont X_A, Y_A, Z_A .

Nous noterons le *trièdre géographique local* : TGL (x_g, y_g, z_g) . Celui-ci ne définit pas un référentiel galiléen car, il est lié à la rotation de la Terre. Dans un tel repère, la loi fondamentale de la dynamique doit tenir compte des forces d'entraînement et de Coriolis. Les conditions finales des engins qui doivent atteindre une orbite doivent cependant être écrites dans le référentiel inertiel, car les conditions portent généralement sur les paramètres orbitaux exprimés dans ce repère.

Dans le TGL, les coordonnées seront donc :

l'altitude	$h,$
la longitude	$\lambda,$
la latitude	$\phi,$
le module de la vitesse	$V,$
la pente	γ et
l'azimut	$\psi.$

Dans le TGL ou le référentiel inertiel, le vecteur position reste inchangé, mais le vecteur vitesse, lui, est composé par la vitesse d'entraînement.

$$\vec{V}_{\text{gal}} = \vec{V}_{\text{TGL}} + \vec{V}_{\text{rot}}$$

où

$$\begin{cases} \vec{V}_{\text{TGL}} &= V(\sin \gamma \vec{z}_g + \cos \gamma (\sin \psi \vec{x}_g + \cos \psi \vec{y}_g)) \\ \vec{V}_{\text{rot}} &= \omega_T (R_T + h) \cos \phi \vec{x}_g \end{cases}$$

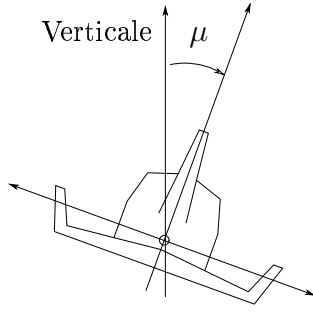


Figure 2.3: Gîte aérodynamique μ

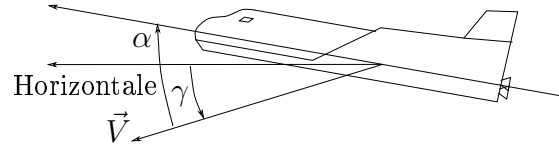


Figure 2.4: Pente γ et incidence α

Finalement les coordonnées du vecteur vitesse dans le repère galiléen sont données par les relations suivantes :

$$V_{\text{gal}}^2 = V^2 + 2V\omega_T(R_T + h) \cos \phi + \omega_T^2(R_T + h)^2 \cos^2 \phi \quad (2.1)$$

$$\sin \gamma_{\text{gal}} = \frac{V}{V_{\text{gal}}} \sin \gamma \quad (2.2)$$

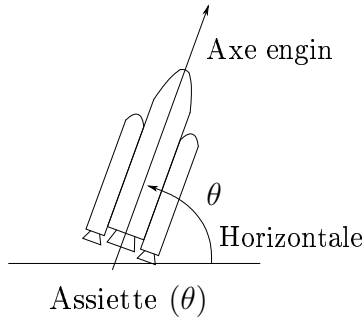
$$\sin \psi_{\text{gal}} = \frac{V \cos \gamma}{V_{\text{gal}} \cos \gamma_{\text{gal}}} \sin \psi \quad (2.3)$$

D'autre part, il nous sera utile d'exprimer la condition d'inclinaison en fonction de variables connues. Nous utiliserons la relation suivante :

$$\cos i = \sin \psi_{\text{gal}} \cos \phi_{\text{gal}} \quad (2.4)$$

2.3 Angles d'attitude du véhicule (commande)

Pour un véhicule contrôlé par les forces aérodynamiques, les angles généralement choisis pour représenter son attitude sont la gîte aérodynamique et l'incidence. Dans toute notre étude, nous supposons que l'incidence en lacet, ou dérapage, est nulle, ou tout du moins négligée. Dans ces conditions, la gîte se confond avec la gîte aérodynamique μ . Pour un véhicule contrôlé par l'orientation du vecteur poussée, l'attitude est plus généralement représentée par l'assiette :



L'assiette est en fait la différence entre l'incidence et la pente lorsque la gîte est nulle ($\theta = \gamma + \alpha$).

Du point de vue de l'optimisation le choix de l'assiette ou de la gîte comme commande du système peut influencer sur la résolution. En effet, la résolution est plus "simple" lorsque la commande agit directement sur le système, en pratique il est plus efficace de contrôler la direction du vecteur poussée (et donc l'assiette) qui est la force principale lors de la montée, plutôt que l'incidence.

2.4 Dynamique

La dynamique est le résultat du choix des variables d'état et de commande et provient principalement de l'application du principe fondamental de la dynamique. Ainsi pour construire cette dynamique il est nécessaire de préciser les forces en présence dans le système. Il s'agit principalement de la gravité, des forces aérodynamiques, de la poussée si elle existe, et des forces d'entraînement et de Coriolis si le référentiel choisi n'est pas galiléen. Nous parlerons dans la suite des modèles choisis qui définissent ces forces.

2.4.1 Modèles de gravité

La première force mise en jeu, est la gravité. Quelle que soit la trajectoire que l'on aura à calculer, cette force sera présente. Elle a pour particularité d'être induite par un champ de vecteur \vec{g} lui-même dérivant d'un potentiel U . Cette force de gravitation aura comme expression :

$$\vec{F}_{\text{grav}} = m\vec{g} \quad (2.5)$$

Le vecteur \vec{g} est par définition l'opposé du gradient du potentiel de gravitation. Dans notre cas, nous supposons qu'il est uniquement d'origine terrestre et nous négligerons l'attraction des autres astres. Dans le repère géographique local R_g :

$$\vec{g} = -\overrightarrow{\text{grad}} U = -\frac{\partial U}{\partial r} \vec{z}_g - \frac{1}{r} \frac{\partial U}{\partial \varphi} \vec{y}_g - \frac{1}{r \cos(\varphi)} \frac{\partial U}{\partial \lambda} \vec{x}_g \quad (2.6)$$

Par définition du gradient en coordonnées sphériques.

En première approximation

Si la Terre est considérée comme un corps parfaitement sphérique de densité homogène, le champ de gravité sera alors centripète et ne dépendra que de la distance au centre de la Terre. Nous parlerons du potentiel “Képlérien” $U = -\frac{\mu}{r}$, qui induit un champ de gravité : $\vec{g} = -\frac{\mu}{r^2}\vec{z}_G$.

En toute rigueur

La Terre n’est pas un corps à symétrie parfaitement sphérique; il est alors possible de développer U sous la forme d’une série trigonométrique.

$$U = \frac{\mu}{r} \left[1 + \sum_{n=2}^{\infty} J_n \left(\frac{R_e}{r} \right)^n P_n(\sin(\varphi)) + \sum_{n=2}^{\infty} \sum_{k=1}^{\infty} J_n^k P_n^k \sin(\varphi) \cos(k(\lambda - \lambda_n^k)) \right]$$

où les J_n représentent les harmoniques zonaux, P_n les harmoniques tesséraux et les λ_n sont des paramètres de phase.

En pratique

Un développement à l’ordre 2 constitue une bonne approximation en ce qui concerne notre application (ceci revient à considérer que la Terre est ellipsoïdale). Dans le cadre de travaux sur le calcul de trajectoire de satellites, le développement doit généralement être plus poussé.

$$U = -\frac{\mu}{r} \left[1 - J_2 \left(\frac{R_e}{r} \right)^2 \frac{3 \sin^2(\varphi) - 1}{2} \right]$$

d’où dans R_g :

$$\begin{aligned} \vec{g} = & -\frac{\mu}{r^2} \left[1 - \frac{3}{2} J_2 \left(\frac{R_e}{r} \right)^2 (3 \sin^2(\varphi) - 1) \right] \vec{z}_G \\ & - \frac{\mu}{r^2} J_2 \left(\frac{R_e}{r} \right)^2 3 \sin(\varphi) \cos(\varphi) \vec{y}_G \end{aligned}$$

2.4.2 Forces aérodynamiques

Le véhicule se déplaçant dans l’atmosphère interagit avec celle-ci par le biais de forces de frottement de nature complexe. Dans la modélisation classique,

la résultante des forces aérodynamiques dépend d'un certain nombre de facteurs en particulier la pression atmosphérique ρ , du véhicule (C_X, C_Z) et de l'attitude de celui-ci (incidence, gîte, ...). Elle est généralement décomposée suivant les axes aérodynamiques et a pour expression :

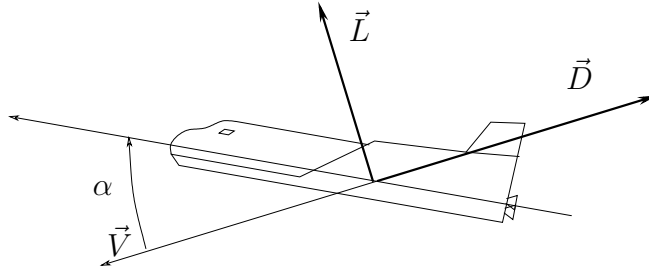
- La traînée *Drag* (opposée à la vitesse) :

$$D = \frac{1}{2} \rho v^2 S_{ref} C_X \quad (2.7)$$

- La portance *Lift* (perpendiculaire à la vitesse) :

$$L = \frac{1}{2} \rho v^2 S_{ref} C_Z \quad (2.8)$$

On fait l'hypothèse que le véhicule vole à dérapage nul, c'est à dire que la résultante aérodynamique est dans le plan de symétrie de l'engin.



Nous allons maintenant nous intéresser à la modélisation des coefficients aérodynamiques du véhicule. La modélisation de l'atmosphère sera traitée dans l'annexe A.1, à la page 153.

Modélisation aérodynamique

Dans le modèle le plus complet, à six degrés de liberté, les résultantes aérodynamiques en force et moment peuvent se décomposer de la façon suivante en axes aérodynamiques pour la force et en axes véhicules pour le moment. Les Figures 2.5 représentent le dérapage et l'incidence avec le repère véhicule (x_v, y_v, z_v) et le repère aérodynamique (x_a, y_a, z_a) .

Les forces :

$$\mathbf{F}_a = \begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} = \frac{1}{2} \rho V^2 S_{ref} \begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix}$$

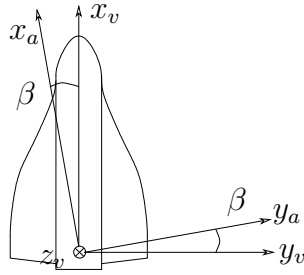


Figure 2.5: Dérapiage (β)

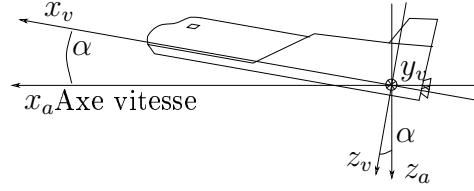


Figure 2.6: Incidence (α)

où

$$\begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix} = \begin{pmatrix} \Delta C_x(\alpha) + \Delta C_x(\alpha, \delta_m) + \Delta C_x(\alpha, \delta_{bf}) \\ \Delta C_y(\alpha, \beta) + \Delta C_y(\alpha, \delta_l) + \Delta C_y(\alpha, \delta_n) \\ \Delta C_z(\alpha) + \Delta C_z(\alpha, \delta_m) + \Delta C_z(\alpha, \delta_{bf}) \end{pmatrix}$$

Les moments :

$$\mathbf{M}_a = \begin{pmatrix} L \\ M \\ N \end{pmatrix} = \frac{1}{2} \rho V^2 S_{ref} l \begin{pmatrix} C_l \\ C_m \\ C_n \end{pmatrix} + \frac{1}{2} \rho V S_{ref} l^2 \begin{pmatrix} C_{lp}(\alpha)p + C_{lr}(\alpha)r \\ C_{mq}(\alpha)q \\ C_{np}(\alpha)p + C_{nr}(\alpha)r \end{pmatrix}$$

où

$$\begin{pmatrix} C_l \\ C_m \\ C_n \end{pmatrix} = \begin{pmatrix} \Delta C_l(\alpha, \beta) + \Delta C_l(\alpha, \delta_l) + \Delta C_l(\alpha, \delta_n) \\ \Delta C_m(\alpha) + \Delta C_m(\alpha, \delta_m) + \Delta C_m(\alpha, \delta_{bf}) \\ \Delta C_n(\alpha, \beta) + \Delta C_n(\alpha, \delta_l) + \Delta C_n(\alpha, \delta_n) \end{pmatrix}$$

avec

- p, q, r : composantes du vecteur vitesse de rotation en repère véhicule
- l : longueur de référence
- δ_l : angle de braquage de la gouverne de roulis
- δ_m : angle de braquage de la gouverne de tangage
- δ_n : angle de braquage de la gouverne de lacet
- δ_{bf} : angle de braquage de la gouverne de ventral (body-flap)

Tous ces coefficients dépendent du nombre de Mach et de l'altitude (effet de raréfaction à haute altitude, effet de sol au voisinage immédiat du sol). On a indiqué ici, les dépendances principales mais il peut se rajouter d'autres couplages (exemple : un terme $\delta C_z(\delta_n)$) et d'autres influences (exemple : aérofreins).

Remarque : lorsque l'on peut linéariser ces termes, en bas subsonique par exemple, on notera par exemple : $\Delta C_n(\delta_n) = C_{n,\delta_n} \delta_n$.

On définit également le coefficient $C_{n\beta}^*$ appelé coefficient de lacet dynamique par la relation :

$$C_{n\beta}^* = C_{n\beta} \cos(\alpha) - \frac{I_{zz}}{I_{xx}} C_{l\beta} \sin(\alpha)$$

Les gouvernes définies plus haut peuvent être virtuelles :

- δ_n peut résulter de la combinaison du braquage des gouvernes de winglet plutôt que du braquage du gouvernail de direction d'une dérive centrale
- δ_m et δ_l peuvent résulter sur une aile delta (utilisé dans de nombreux projets ailés de véhicules de rentrée) du braquage combiné ou différentiel respectivement des ailerons situés en bord de fuite de l'aile.

Coefficients aérodynamiques équilibrés

Dans les études de calcul de performance, trop de précision n'est pas utile et pose le problème de la trop grande dimension du vecteur de commande. La solution est donc de simplifier la modélisation du problème en faisant un certain nombre d'hypothèses. En particulier, nous choisissons un dérapage β nul, des braquages de gouvernes latérales δ_n et δ_l nuls, les vitesses de rotation nulles (équilibre) et tous les moments nuls aussi. Finalement il ne reste plus que l'incidence α et le gîte μ pour commander le véhicule. Finalement le braquage de gouverne de tangage δ_m est donné par l'équation de moment selon l'axe y de tangage. Les coefficients C_x et C_z tiennent alors compte de ces braquages.

Ce sont ces coefficients équilibrés qui doivent être utilisés pour les calculs de trajectoire. En effet, l'utilisation des coefficients en lisse ($\delta_m = 0$) peut conduire à surestimer la finesse ($f = \frac{C_z}{C_x}$) jusqu'à 20 %.

Si on utilise la théorie Newtonienne de l'impact pour le régime hypersonique, on peut obtenir l'approximation suivante pour les coefficients aérodynamiques en fonction de l'incidence α (en axes aérodynamiques) :

$$C_z = C_{z_0} \sin^2 \alpha$$

avec $C_{z_0} = 2$ en Newtonien strict, un peu moins en pratique.

Pour C_x , on peut utiliser la modélisation de polaire parabolique :

$$C_x = C_{x_0} + k C_z^2$$

avec C_{x_0} et k des constantes (variant cependant avec le nombre de Mach) à ajuster pour chaque véhicule.

On déduit de ces formules qu'il y a une incidence maximisant la finesse. Cependant, en régime hypersonique, pour protéger le cockpit s'il y en a un, et l'extrados du véhicule, on est conduit à adopter des incidences beaucoup plus grandes. De la sorte on a une finesse réduite et un écoulement complètement décollé sur l'extrados (en subsonique, on parlerait de "décrochage").

En pratique, les coefficients aérodynamiques sont des données tabulées et doivent être interpolés en fonction du Mach et de l'incidence. Mais nous utiliserons aussi une simplification supplémentaire supprimant la dépendance en Mach, et considérant C_x comme un polynôme du deuxième degré de α et C_z comme une fonction affine de ce même α .

2.4.3 Forces d'entraînement et de Coriolis

Dans la modélisation que nous devons considérer, il nous faut prendre en compte la rotation de la Terre. Du fait que celle-ci tourne autour d'un axe et que nous choisissons un repère tournant avec elle, tout se passe comme si des forces dites d'entraînement et de Coriolis agissaient. Elles dépendent de la vitesse de rotation de la Terre, de la vitesse relative (c'est la vitesse dans le repère tournant, ou repère géographique local), et de la distance à l'axe de rotation. On remarquera, que la force de Coriolis n'existe que dans le cas où la vitesse relative est non nulle, alors que la force d'entraînement existe dans tous les cas.

Force d'entraînement

Celle-ci est orthogonale à l'axe de rotation et centrifuge. Son expression est :

$$\vec{F}_e = \omega_T^2 \cos \phi (R_e + z) \vec{e} \quad (2.9)$$

avec $|\vec{\Omega}| = \omega_T$ et \vec{e} le vecteur unitaire centrifuge.

Force de Coriolis

Cette force, découverte par Gustave Gaspard Coriolis (1792-1843), est orthogonale à la vitesse relative et à l'axe de rotation de la Terre, et a pour expression :

$$\vec{F}_c = 2\vec{\Omega} \wedge \vec{V}_r. \quad (2.10)$$

$\vec{\Gamma}$: Accélération du véhicule
\vec{g}	: Gravité au point courant
m	: Masse du véhicule
$\vec{F}_{\text{aéro}}$: Résultante des forces aérodynamiques sur le véhicule.
\vec{F}_e	: Force d'entraînement.
\vec{F}_c	: Force de Coriolis.
$\vec{F}_{\text{poussée}}$: Force de poussée.

Table 2.2: Les forces mises en jeu

h	altitude	λ	longitude
ϕ	latitude	V	module de la vitesse
γ	pente	ψ	azimut

Table 2.3: Le jeu de coordonnées

2.4.4 Principe fondamental de la dynamique

En appliquant le principe fondamental de la dynamique, au système véhicule, dans le TGL nous trouverons :

$$\vec{\Gamma} = \vec{g} + \frac{\vec{F}_{\text{aéro}}}{m} + \frac{\vec{F}_e + \vec{F}_c}{m} + \frac{\vec{F}_{\text{poussée}}}{m}$$

Le Tableau 2.2 présente les différentes forces de cette formule.

2.4.5 Équation du mouvement en atmosphère

Nous cherchons ici à décrire le système, puis à projeter la dynamique sur une base pertinente.

Pour décrire le système, les paramètres $(z, \lambda, \phi, V, \gamma, \psi)$ du TGL (qui est tournant) ont été choisis. Le Tableau 2.3 rappelle la signification de ces variables.

En projetant ensuite le principe fondamental de la dynamique sur ces coordonnées, nous obtenons le système d'équations 2.11 à 2.15. Les équations 2.11 à 2.13 représentent la dynamique du vecteur position, alors que 2.14 à 2.15 représentent celle du vecteur vitesse.

$$\dot{h} = V \sin \gamma \quad (2.11)$$

$$\dot{\lambda} = \frac{V \cos \gamma \sin \psi}{(h + R_e) \cos \phi} \quad (2.12)$$

$$\dot{\phi} = \frac{V \cos \gamma \cos \psi}{h + R_e} \quad (2.13)$$

$$\begin{aligned} \dot{V} = & \frac{F \cos \alpha - D}{m} - g \sin \gamma \\ & + \omega_T^2 (h + R_e) \cos \phi (\sin \gamma \cos \phi - \cos \gamma \sin \phi \cos \psi) \\ \dot{\gamma} = & \frac{(L + F \sin \alpha) \cos \mu}{mV} + \left(\frac{V^2}{h + R_e} - g \right) \frac{\cos \gamma}{V} + 2\omega_T \sin \psi \cos \phi \\ & + \omega_T^2 \frac{h + R_T}{V} \cos \phi (\cos \gamma \cos \phi + \sin \gamma \sin \phi \cos \psi) \end{aligned} \quad (2.14)$$

$$\begin{aligned} \dot{\psi} = & \frac{(L + F \sin \alpha) \sin \mu}{mV \cos \gamma} + \frac{V}{h + R_e} \cos \gamma \sin \psi \tan \phi \\ & + 2\omega_T (\sin \phi - \cos \psi \cos \phi \tan \gamma) \\ & + \omega_T^2 \frac{h + R_e}{V} \frac{\sin \phi \cos \phi \sin \psi}{\cos \gamma} \end{aligned} \quad (2.15)$$

Ainsi les trajectoires que nous étudierons, devront vérifier cette dynamique. Lorsque la masse sera incluse dans le vecteur d'état sa dynamique sera la suivante :

$$\dot{m} = -Q. \quad (2.16)$$

Où Q est le débit dans la force de poussée

2.5 Contraintes sur le système

Notre recherche de commande optimale, doit, en plus d'être optimale, permettre au système de rester dans un domaine réaliste; il est impératif, par exemple, que le véhicule ne fonde pas sous l'effet de la chaleur de friction lors de la rentrée, qui est un phénomène dans lequel beaucoup d'énergie doit être dissipée. Les différentes contraintes sont ainsi décrites dans cette section.

Dans un premier temps, nous décrirons la limite thermique que possède un véhicule spatial, la limite d'équilibre et la limite en facteur de charge, la limite de pression dynamique puis la contrainte de couloir d'incidence. Nous

décrivons également une approche possible pour le guidage en rentrée, basé sur la notion de “*couloir de rentrée*”. Le dernier paragraphe donnera une représentation de ce domaine.

2.5.1 Limites thermiques

Comme il a été dit en introduction, il y a une limite de flux à ne pas dépasser.

Modélisation thermique simplifiée

Les échauffements supportés par le véhicule lors de son retour sont dûs à la forte augmentation de la température en arrière de l’onde de choc hypersonique.

En n’importe quel point de la paroi, on a la relation :

$$\Phi_{\text{conduction}} + \Phi_{\text{rayonnement}} = \Phi_{\text{convection}}$$

Elle exprime que le flux de convection dans la couche limite autour de la paroi est pour partie conduit à l’intérieur de la paroi et pour partie, dissipé sous forme de rayonnement vers l’espace.

Le flux de rayonnement est donné par la loi de Stefan :

$$\Phi_{\text{rayonnement}} = \epsilon \sigma_0 (T_p^4 - T_r^4)$$

où

- σ_0 est la constante de Stefan ($= 5.73 \cdot 10^{-8} [W m^{-2} K^{-1}]$)
- ϵ est l’émissivité (≈ 0.85 pour du composite carbone/carbone)
- T_p est la température de la paroi externe
- T_r est la température de rayonnement à l’infini ($\approx 20[K]$)

Le calcul de ces flux en 2 ou 3 dimensions requiert des programmes volumineux et complexes (différences finies, éléments finis ...) délicats de mise en oeuvre et coûteux en temps. On peut cependant obtenir une approximation du flux de convection en quelques points particuliers du véhicule et en négligeant la conduction dans la paroi (matériau très isolant). On obtient alors une surestimation de la température de paroi dite alors d’“équilibre”. Les points de la paroi qui reçoivent les flux les plus importants sont ceux exposés à l’écoulement : nez, bords d’attaque des ailes et éventuellement gouvernes fortement braquées.

Évaluation du flux au point d'arrêt du nez et des bords d'attaque

Au point d'arrêt, le flux de convection est approximé par la formule de Fay et Ridell :

$$\Phi_{\text{arrêt}} = C_{\Phi} \sqrt{\frac{\rho}{R_n}} V_{\text{aéro}}^3$$

avec $C_{\Phi} = 1,705 \cdot 10^{-4}$, ρ la densité de l'air, et R_n le rayon de courbure au point d'arrêt.

La formule suivante, semi-empirique, a été recalée sur les mesures de la navette américaine :

$$\Phi_{\text{arrêt}} = C'_{\Phi} \sqrt{\frac{\rho}{R_n}} V_{\text{aéro}}^{3,07} \left(1 - \frac{C_p T_p}{C_p T_{\infty} + \frac{1}{2} V_{\text{aéro}}^2} \right)$$

Une formulation classique du flux thermique, est celle de l'écoulement laminaire, et a pour expression :

$$\Phi = C_q \sqrt{\frac{\rho}{R_{\text{nez}}}} V_{\text{aéro}}^3 \quad (2.17)$$

avec $C_q = 1,705 \cdot 10^{-4}$, ρ densité de l'air, et R_{nez} rayon du "nez".

On pourra chercher :

$$\Phi \leq \Phi_{\text{max}} = \sigma \epsilon T_{\text{max}}^4$$

où ϵ est l'émissivité imposée apparente, et $\sigma = 5.67 \cdot 10^{-8} [W m^{-2} K^{-4}]$.

2.5.2 Limites d'équilibre

Cette contrainte décrit une condition de vol utilisée pour la rentrée. Si celle-ci n'est pas vérifiée, le véhicule décroche.

Cette contrainte est valable dans un domaine de vol où les approximations décrites par Sängner (voir [36] ou [25]) sont valables. Celles-ci sont :

- $\frac{V^2}{r} - g \ll L/m$ pour un véhicule à finesse non nulle.
- la pente γ est faible, de manière à ce que $mg \sin \gamma \ll D$
- $\dot{\gamma}$ reste faible, car γ variable rapide.
- $\rho = \rho_0 \exp(-\eta z)$

Avec toutes ces hypothèses, les équations de la dynamique pour un véhicule de rentrée non propulsé peuvent s'écrire :

$$\begin{cases} \dot{r} &= V\gamma \\ r\dot{\lambda} &= V\frac{\sin\psi}{\cos\phi} \\ r\dot{\phi} &= V\cos\psi \\ \dot{V} &= -\frac{D}{m} \\ 0 &= \frac{L}{m}\cos\mu + \left(\frac{V^2}{r} - g\right) \\ V\psi &= \frac{L}{m}\sin\mu \end{cases}$$

La cinquième équation définit une relation d'équilibre entre la portance verticale et l'accélération de la gravité diminuée de la force d'inertie centrifuge. La condition d'équilibre sera :

$$g \leq \frac{L}{m}\cos\mu + \frac{V^2}{r} \quad (2.18)$$

2.5.3 Limites en facteur de charge

Dans le vol à grande vitesse, les forces prépondérantes, sont les forces aérodynamiques qui agissent fortement car sont de la forme $K\rho V^2$ et les forces de poussée si elles existent. Or, le véhicule ne peut pas supporter une accélération trop violente d'autant plus si le vol est habité : les passagers devraient subir un poids apparent nettement supérieur à leur poids réel, ce qui n'est physiologiquement pas souhaitable. De plus même dans les vols non habités, la structure de l'engin ne peut pas supporter trop de "g". Ainsi pour toutes ces raisons une limite en facteur de charge est mise en place.

En l'absence de force de poussée, en particulier dans le cas de la rentrée atmosphérique, le facteur de charge total a pour formule :

$$n_{total} = \frac{1}{2} \frac{\rho S}{mg_0} (\sqrt{C_X^2 + C_Z^2}) V_{aéro}^2 \quad (2.19)$$

Il correspond à l'accélération induite par les forces aérodynamiques. On cherchera :

$$n_{total} \leq n_{total\ max},$$

où $n_{total\ max}$ est donné par les caractéristiques du véhicule. Dans certains cas, on souhaite borner le facteur de charge transverse uniquement, la composante dans l'axe engin étant considérée comme moins problématique :

$$n_{trans} = \frac{1}{2} \frac{\rho S}{mg_0} (C_Z \cos\alpha - C_X \sin\alpha) V_{aéro}^2 \quad (2.20)$$

Dans le cas de la montée lanceur, on choisit généralement une autre formulation pour exprimer les efforts aérodynamiques transverses. Le paragraphe suivant décrit cette autre contrainte que nous nommerons P_α .

2.5.4 Limites en pression dynamique

Tout comme la limite en facteur de charge, elle correspond à une autre formulation de la limite structurelle du véhicule. La formule de la pression dynamique est :

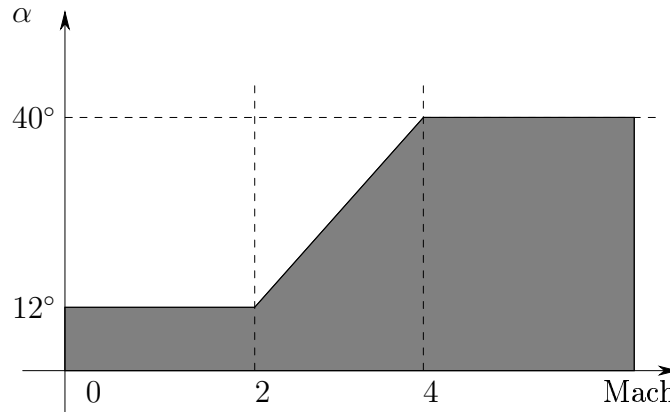
$$P_{dyn} = \frac{1}{2} \rho V_{aéro}^2. \quad (2.21)$$

Pour un lanceur “classique” de type consommable, on prend parfois la contrainte P_α , produit de la pression dynamique et de l’incidence. Cette contrainte représente bien les efforts transverses que subit un lanceur :

$$P_\alpha = P_{dyn} \alpha. \quad (2.22)$$

2.5.5 Couloir d’incidence

Pour certains véhicules, l’incidence ne peut pas prendre n’importe quelle valeur suivant la vitesse, ou plus précisément suivant le nombre de Mach ($\text{Mach} = \text{Vitesse} / \text{Vitesse du son}$). C’est en particulier le cas du booster du concept FSSC16 pour lequel le couloir d’incidence a la forme suivante :



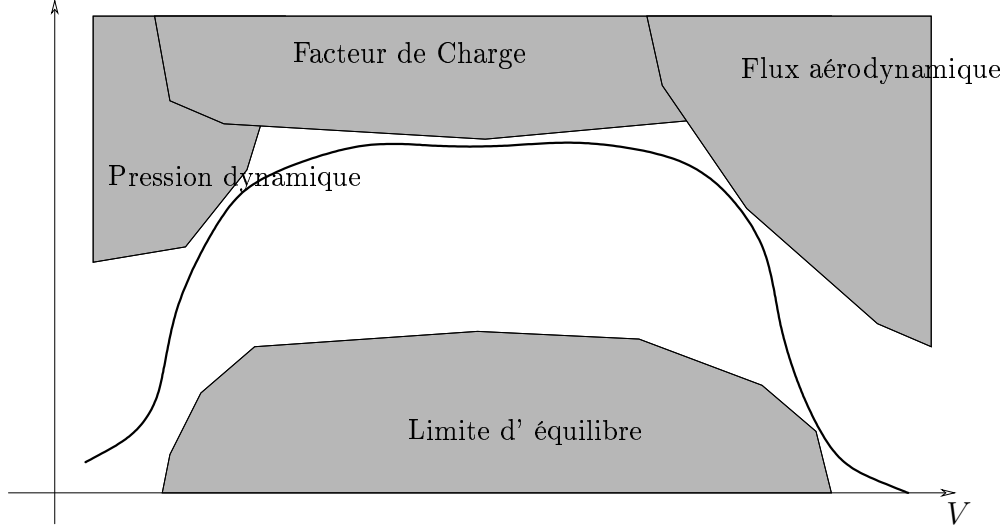
En fait, ce type de contrainte est lié à la pilotabilité du véhicule dans l’atmosphère.

2.5.6 Couloir de rentrée

Toutes ces contraintes créent un couloir appelé couloir de rentrée, dans lequel le véhicule doit évoluer. Ce type d’information est particulièrement utilisé en

guidage, on le représente généralement dans le repère traînée/vitesse pour un profil $\alpha = f(M)$ donné.

$$D = \frac{1}{2} \frac{S_{ref} C_x}{m} \rho V^2$$



Ce graphique est essentiellement utilisé pour le guidage en rentrée, et permet de positionner la trajectoire dans un certain domaine admissible qui matérialise la zone de satisfaction des contraintes.

2.6 Jonctions entre les arcs

Entre chaque arc, il y a des conditions dites de jonction. En particulier la continuité des vecteurs position et vitesse lors d'une séparation d'étage, ou le lien entre les masses à cet instant. En effet, à l'issue de la séparation, la masse de l'orbiteur vaut la masse du véhicule couplé avant l'événement moins la masse du booster. Ces ensembles de conditions seront détaillés lorsque nous présenterons un concept en particulier.

2.7 Exemple de concepts multiarcs

Nous étudions le cas de deux véhicules, l'un du type SOH et l'autre de type bi-étage en configuration siamoise. Nous nous inspirons des données *FESTIP* (Future European Space Transportation Investigations Program) pour les valeurs numériques.

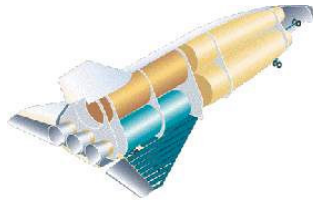


Figure 2.7: FSSC15

Dans cette partie, nous présenterons ces concepts et les caractéristiques de ces lanceurs futurs. (Pour plus de détails numériques voir : [34]). Les données que nous avons choisies sont des données simplifiées, en effet, nous cherchons dans cette thèse à démontrer l'efficacité de la méthode plus qu'à avoir une modélisation trop réaliste.

2.7.1 Monoétage de type SOH

Ce concept SOH (Sub Orbital Hopper) extrait du programme FESTIP consiste en un véhicule possédant une voilure (FSSC15). L'engin fera la majeure partie de la montée puis redescendra sur Terre pour atterrir sur l'île de l'Ascension. Il s'agit d'un *TSTO* (Two Stages To Orbit) qui a pour objectif de mettre une charge utile en orbite basse quasi équatoriale (250kmx250km pour une inclinaison de 5.2°). Le premier étage décolle horizontalement depuis Kourou, atteint une altitude et une vitesse sub-orbitale, puis déploie le second étage consommable qui atteindra l'orbite. Il y a donc une phase balistique où le véhicule n'est pas manoeuvrable.

2.7.2 Bi-étage en configuration siamoise

Nous nous intéressons en particulier au concept FSSC16 qui regroupe deux types de véhicules TSTO, un véhicule semi-réutilisable (SR) et un totalement réutilisable (FR). Ici, nous présenterons les caractéristiques simplifiées du FSSC16-FR. Le FSSC16-FR est donc un véhicule composé d'un booster et d'un orbiter, tous les deux réutilisables dans une configuration siamoise, ce qui signifie qu'ils possèdent la même forme et sont accolés l'un à l'autre (au lieu d'être disposés en ligne). Le booster se sépare de l'orbiter à basse altitude, et revient au site de lancement, au moyen

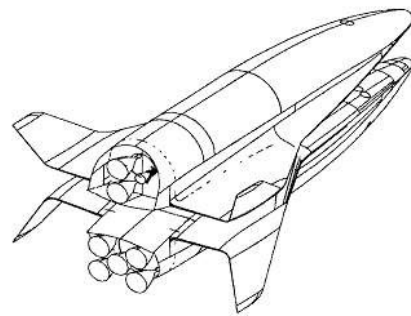


Figure 2.8: FSSC16-FR

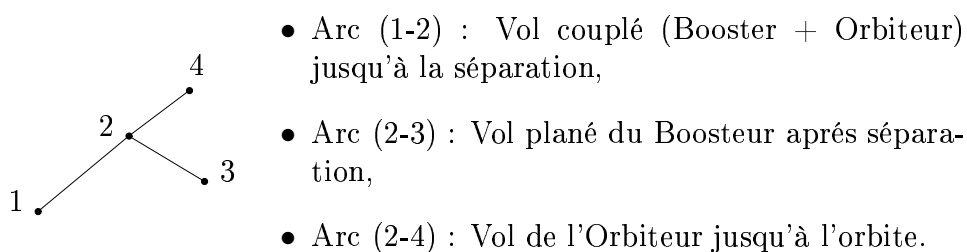


Figure 2.9: Graphe de scénario FSSC16

Caractéristique	Booster	Orbiteur
Masse sèche	56 869 kg	40 044 kg
Masse d'ergol	166 071,42 kg	247 928,57 kg
Moteurs Vulcain	5	2
	Booster + Orbiter arc (1-2)	Orbiteur arc (2-4)
Moteurs Vulcain	7	2
Poussée	9 210 707 N	2 631 631 N
Débit	2 110,469 kg/s	603,0049 kg/s
Temps de Poussée	78,68	411,15

Table 2.4: Caractéristiques propulsives

de moteurs turbo annexes. L'orbiter, lui, termine sa course jusqu'à l'orbite où il injecte la charge utile. Puis, après une manoeuvre de désorbitation, celui-ci revient se poser sur terre en planant. Lors de la phase de montée couplée, l'orbiteur alimente ses moteurs fusée avec les ergols du booster, ceci permet à l'orbiter de conserver ses réservoirs pleins jusqu'au moment de la séparation, c'est le "crossfeeding".

Caractéristiques propulsives

Nous supposons pour notre modèle que la poussée est constante, ce qui n'est en réalité pas le cas, car le régime de poussée du moteur Vulcain, utilisé sur ce concept, suit un profil particulier. De la même manière, nous avons choisi un débit constant, ce qui fixe les durées de chacun des arcs propulsés. Le Tableau 2.4 récapitule les données numériques choisies.

a_0	-0.05	b_0	0.05
a_1	1.67e-2	b_1	1.67e-4
		b_2	3.67e-4

Table 2.5: Donnée numérique C_D , C_L

Altitude	830 m
Latitude	5,232°
Longitude	-52,775°
Vitesse	83 m/s
Pente	89°
Azimut	libre
Masse	490912,99 kg

Table 2.6: Sommet 1 - FSSC16-FR

Caractéristique aérodynamiques

Pour la montée jumelée ainsi que le booster seul, les caractéristiques aérodynamiques seront les mêmes à l'exception de la surface de référence qui sera de $405,6m^2$ pour les véhicules indépendants, et de $811,2m^2$ pour le vol couplé. Les coefficients aérodynamiques eux seront de la forme :

$$C_L(\alpha) = a_0 + a_1\alpha \quad (2.23)$$

$$C_D(\alpha) = b_0 + b_1\alpha + b_2\alpha^2. \quad (2.24)$$

Les valeurs numériques choisies pour les coefficients sont résumées dans le Tableau 2.5 lorsque l'angle α est exprimé en degrés.

Conditions aux bornes

Nous allons ici décrire les conditions sur chacun des sommets du graphe de scénario (Figure 2.9).

Conditions initiales (sommets 1) Les caractéristiques sont décrites dans le Tableau 2.6. Pour des raisons de stabilité numérique, nous avons choisi des conditions initiales légèrement différentes de celles correspondant au départ sur le pas de tir. En effet, au vu de la dynamique, la vitesse ne peut être nulle ni la pente verticale, sinon la dynamique de l'azimut ne serait pas définie. En fait, nous prenons une approximation qui correspondrait à l'état après 10 secondes de poussée.

Altitude	100 km
Vitesse	7889.0 m/s
Pente	0°

Table 2.7: Sommet 1 - FSSC16-FR

Condition à la séparation (sommet 2) À ce sommet les conditions sont essentiellement un “raccordement” entre les variables d’état excepté pour les masses qui vérifient la relation suivante :

$$m_{12}(t_f) = m_{24}(0) + m_{23} \quad (2.25)$$

Condition finale du retour Booster plané (sommet 3) La condition finale sera la condition pour l’allumage des moteurs turbo, c’est à dire une altitude de 6 km et une vitesse de 170 m/s.

Condition finale Orbiter (sommet 4) Nous nous intéressons ici, à une mission polaire pour laquelle nous chercherons à atteindre l’orbite elliptique 100kxm250kxm98. Dans notre cas, la condition finale sera d’être au périégée de cette orbite, soit dans le référentiel inertiel les caractéristiques décrites dans le Tableau 2.7.

Critère d’optimisation

Nous choisissons comme critère à minimiser la distance restant à parcourir par le booster, entre la fin de la phase de retour planée et le site d’atterrissage (en l’occurrence Kourou).

Contraintes courantes sur chaque arc

Sur chacun des arcs de la mission, chaque trajectoire possède un certain nombre de contraintes courantes. Le Tableau 2.8 résume l’ensemble de ces contraintes. Les trajectoires de cette mission sont données dans le chapitre 7.

	Min	Max
Arc 1		
Incidence	-5°	5°
Gîte	-10°	10°
Pente	0°	90°
Arc 2		
Incidence	-30°	30°
Gîte	-30°	30°
Arc 3		
Incidence	-5°	5°
Gîte	-10°	10°
Facteur de Charge	-2.5 g	2.5 g

Table 2.8: Contraintes sur le système

Partie III

La discrétisation de Runge-Kutta

Chapitre 3

Méthode de Runge-Kutta en commande optimale

Les méthodes de Runge-Kutta sont des méthodes très performantes pour intégrer numériquement des systèmes différentiels. Or, dans un problème de commande optimale, les conditions d'optimalité demandent l'intégration d'un système qui s'interprète comme un système Hamiltonien. Il faut ainsi étudier ces méthodes dans le cadre du contrôle optimal.

Dans ce chapitre, les méthodes de Runge-Kutta seront présentées, puis les conditions d'ordre de ces méthodes en terme d'arbre seront abordées. L'extension au cas des systèmes Hamiltoniens sera évoqué.

3.1 Introduction

Nous traiterons deux types d'équations :

- les équations différentielles ordinaires (*EDO*) :

$$\dot{y}(x) = f(y(x)) \quad (3.1)$$

- les *systèmes partitionnés* :

$$\begin{cases} \dot{p}(t) = f(p(t), q(t)) \\ \dot{q}(t) = g(p(t), q(t)) \end{cases} \quad (3.2)$$

Les systèmes Hamiltoniens peuvent être vus comme des systèmes partitionnés particuliers dans lesquels $f = H_q$ et $g = -H_p$.

3.2 La résolution numérique d'EDO et de systèmes partitionnés

3.2.1 Définitions préliminaires

Définition 2. Une *méthode de Runge-Kutta (RK)* à s étages est définie par l'entier s , un vecteur $b \in \mathbb{R}^s$ et une matrice $a \in \mathbb{R}^{s \times s}$, et est donnée par :

$$\begin{cases} Y_i(x) = y(x_0) + (x - x_0) \sum_{j=1}^s a_{ij} f(Y_j), & i = 1 \dots s \\ Y(x) = y(x_0) + (x - x_0) \sum_{j=1}^s b_j f(Y_j) \end{cases} \quad (3.3)$$

Notation 3. Nous noterons c le vecteur défini par : $c_i = \sum_{j=1}^s a_{ij}$, $i = 1 \dots s$.

Notation 4. Les données d'une méthode RK sont généralement présentées sous la forme d'un tableau, nommé tableau de Butcher :

$$\begin{array}{c|ccc} c_1 & a_{11} & \dots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \dots & a_{ss} \\ \hline & b_1 & \dots & b_s \end{array}$$

Exemple 5. Les méthodes RK contiennent les méthodes d'Euler et de collocation. Le tableau ci-dessous présente quelques exemples :

Euler explicite	$\begin{array}{c c} 0 & 0 \\ \hline & 1 \end{array}$	Euler implicite	$\begin{array}{c c} 1 & 1 \\ \hline & 1 \end{array}$
Point milieu	$\begin{array}{c c} 1/2 & 1/2 \\ \hline & 1 \end{array}$	Trapèze	$\begin{array}{c cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$

Nous parlerons de méthodes explicites si a est triangulaire inférieure stricte, et implicite dans le cas contraire.

3.2.2 Méthodes de collocation

Notation 6. L'idée est de chercher un polynôme qui satisfasse l'égalité (3.1) en des points discrets dits points de collocation.

Définition 7. Soit $c \in \mathbb{R}^s$ un vecteur composé de valeurs distinctes. Le *polynôme de collocation* $u(t)$ est un polynôme de degré au plus s satisfaisant :

$$\begin{cases} u(x_0) &= y_0 \\ u'(x_0 + c_i(x - x_0)) &= f(u(x_0 + c_i(x - x_0))), \quad i = 1 \dots s \end{cases} \quad (3.4)$$

et la solution numérique de la *méthode de collocation* est définie par :

$$u(x)$$

Théorème 8. La méthode de collocation décrite dans la définition 7, est équivalente à une méthode de Runge-Kutta avec les coefficients :

$$a_{ij} = \int_0^{c_i} \ell_j(\tau) d\tau, \quad b_i = \int_0^1 \ell_i(\tau) d\tau \quad (3.5)$$

où $\ell_i(\tau)$ est le polynôme de Lagrange : $\ell_i(\tau) = \prod_{j \neq i} (\tau - c_j) / (c_i - c_j)$

Démonstration. Soit $u(x)$ le polynôme de collocation. Notons :

$$k_i = u'(x_0 + c_i(x - x_0)).$$

Ce polynôme de degré $s - 1$ est égal à son interpolation de Lagrange aux points c_i :

$$u'(x_0 + \tau(x - x_0)) = \sum_{j=1}^s k_j \ell_j(\tau)$$

Par intégration, il vient :

$$u(x_0 + c_i(x - x_0)) = y_0 + (x - x_0) \sum_{j=1}^s k_j \int_0^{c_i} \ell_j(\tau) d\tau, \quad \text{pour } i \text{ allant de } 1 \text{ à } s.$$

En remarquant finalement que $k_i = u'(x_0 + c_i(x - x_0)) = f(u(x_0 + c_i(x - x_0)))$ pour i allant de 1 à s , alors $u(x)$ apparaît comme une méthode de Runge-Kutta. ■

Réciproquement,

Théorème 9. Une méthode de Runge-Kutta peut s'interpréter comme une méthode de collocation si les c_i sont distincts et les conditions $C(s)$ et $B(s)$ sont satisfaites, avec :

$$\begin{aligned} C(q) : \quad \sum_{j=1}^s a_{ij} c_j^{k-1} &= \frac{c_i^k}{k}, \quad k = 1, \dots, q \quad i = 1, \dots, s \\ B(p) : \quad \sum_{i=1}^s b_i c_i^{k-1} &= \frac{1}{k}, \quad k = 1, \dots, p \end{aligned}$$

Démonstration. Comme les c_i sont distincts, les équations de $C(s)$ et $B(s)$ sont linéairement indépendantes (en effet, le déterminant du système à résoudre est de type Vandermonde), et définissent exactement les a_{ij} et b_i ; or la définition des coefficients de Runge-Kutta du théorème 8 vérifient $C(s)$ et $B(s)$ d'où le résultat. ■

3.2.3 Méthodes de Runge-Kutta partitionnées

Définition 10. Étant données deux méthodes de Runge-Kutta (a, b) et (\hat{a}, \hat{b}) , une *méthode de Runge-Kutta partitionnée* intégrant le système Hamiltonien (3.2) est donnée par :

$$\left\{ \begin{array}{l} P_i(t) = p(t_0) + (t - t_0) \sum_{j=1}^s a_{ij} f(P_j, Q_j), \quad i = 1 \dots s \\ Q_i(t) = q(t_0) + (t - t_0) \sum_{j=1}^s \hat{a}_{ij} g(P_j, Q_j), \quad i = 1 \dots s \\ P(t) = p(t_0) + (t - t_0) \sum_{j=1}^s b_j f(P_j, Q_j), \\ Q(t) = q(t_0) + (t - t_0) \sum_{j=1}^s \hat{b}_j f(P_j, Q_j). \end{array} \right. \quad (3.6)$$

3.3 Condition d'ordre pour les schémas de Runge-Kutta

3.3.1 Motivation

L'idée première est de développer la solution exacte ainsi que la solution numérique en séries de Taylor, pour ainsi établir les conditions d'ordre.

Développons naïvement la solution exacte :

$$\begin{aligned} y'(x_0) &= f(y(x_0)), \\ y''(x_0) &= f'(y(x_0))f(y(x_0)), \\ y^{(3)}(x_0) &= f''(y(x_0))(f(y(x_0)), f(y(x_0))) + f'(y(x_0))(f'(y(x_0))(f(y(x_0)))), \\ &\dots \end{aligned}$$

Nous pouvons la réécrire sans perte d'information sous la forme abrégée

Arbre	t	$\rho(t)$	$F(t)$
\bullet	τ	1	f
$\begin{array}{c} \bullet \\ \\ \bullet \end{array}$	$[\tau]$	2	$f'(f)$
$\begin{array}{c} \bullet \\ \\ \bullet \\ \\ \bullet \end{array}$	$[[\tau]]$	3	$f'(f'(f))$
$\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \end{array}$	$[\tau\tau]$	3	$f''(f, f)$
$\begin{array}{c} \bullet \\ \\ \bullet \\ \\ \bullet \\ \\ \bullet \end{array}$	$[[[\tau]]]$	4	$f'(f'(f'(f)))$
$\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \end{array}$	$[[\tau\tau]]$	4	$f'(f''(f, f))$
$\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \end{array}$	$[[\tau]\tau]$	4	$f''(f'(f), f)$
$\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \end{array}$	$[\tau\tau\tau]$	4	$f^{(3)}(f, f, f)$

Table 3.1: Arbre, Densité, Différentielle élémentaire

suivante où $f^{(i)} = f^{(i)}(y(x_0))$:

$$\begin{aligned}
y'(x_0) &= f, \\
y''(x_0) &= f'(f), \\
y^{(3)}(x_0) &= f''(f, f) + f'(f'(f)), \\
y^{(4)}(x_0) &= f^{(3)}(f, f, f) + 3f''(f'(f), f) + f'(f''(f, f)) + f'(f'(f'(f))), \\
&\dots
\end{aligned}$$

Ici, nous nous apercevons que l'on peut, à chaque terme du développement, associer un arbre qui représente une association de f et de ses dérivées. Le Tableau 3.1 décrit cette association. La quantité ρ sera décrite plus loin.

3.3.2 Définitions formelles à propos des arbres

Définition 11. L'ensemble des *arbres* T est défini inductivement par :

- L'arbre τ composé d'un seul sommet et dessiné : \bullet appartient à T
- Si (t_1, t_2, \dots, t_m) sont des arbres, alors l'arbre composé d'une racine se connectant à tous les arbres t_i par un arc, est un élément de T . Nous le noterons $[t_1 \dots t_m]$

Définition 12. Pour un arbre t , la *différentielle élémentaire* est une application $F(t) : E \rightarrow E$ définie inductivement par :

$$F(\tau)(y) = f(y), \quad (3.7)$$

$$F([t_1 \dots t_m])(y) = f^{(m)}(F(t_1)(y), \dots, F(t_m)(y)). \quad (3.8)$$

Définition 13. L'ordre $\rho(t)$ d'un arbre t est son nombre de sommets. On peut aussi le définir inductivement par les relations :

$$\begin{aligned} \rho(\tau) &= 1 \\ \rho([t_1 t_2 \dots t_s]) &= 1 + \rho(t_1) + \rho(t_2) + \dots + \rho(t_s) \end{aligned}$$

Définition 14. La *densité* γ est définie inductivement sur T par :

$$\begin{aligned} \gamma(\tau) &= 1 \\ \gamma([t_1 t_2 \dots t_s]) &= \rho([t_1 t_2 \dots t_s]) \gamma(t_1) \gamma(t_2) \dots \gamma(t_s) \end{aligned}$$

Définition 15. La *symétrie* d'un arbre t est l'ordre de son groupe symétrique. On le nomme $\sigma(t)$.

Remarque 16. Le groupe symétrique d'un arbre est le groupe des permutations des noeuds qui laissent l'arbre inchangé.

Théorème 17. [24, thm.301A p.127] Si $t = [t_1^{n_1} t_2^{n_2} \dots t_s^{n_s}]$, où les t_i sont distincts, alors

$$\begin{aligned} \sigma(\tau) &= 1 \\ \sigma(t) &= n_1! n_2! \dots n_s! \sigma(t_1)^{n_1} \sigma(t_2)^{n_2} \dots \sigma(t_s)^{n_s} \end{aligned}$$

3.3.3 Séries B

Définition 18. Étant donnée une application $a : T \cup \{\emptyset\} \rightarrow \mathbb{R}$, une série formelle de la forme :

$$B(a, y) = a(\emptyset)y + \sum_{t \in T} \frac{(x - x_0)^{\rho(t)}}{\sigma(t)!} a(t) F(t)(y)$$

sera appelée une *série B*.

Lemme 19. [46, Lemme 1.9 p.53] Soit une application $a : T \cup \{\emptyset\} \rightarrow \mathbb{R}$ vérifiant : $a(\emptyset) = 1$, alors nous avons :

$$hf(B(a, y)) = B(a', y)$$

avec $a'(\emptyset) = 0$, $a'(\tau) = 1$ et $a'([t_1 t_2 \dots t_s]) = a(t_1) a(t_2) \dots a(t_s)$

3.3.4 Série B de la solution exacte d'une EDO

Théorème 20. La solution exacte d'une EDO (3.1) est la série B associée à l'application e telle que :

$$e(\emptyset) = 1, \quad e(t) = \frac{1}{\gamma(t)} \quad \forall t \in T;$$

Démonstration. Conséquence de [46, Theorem 1.3 p.49] et [46, Theorem 1.4 p.51]. ■

3.3.5 Série B de la solution numérique de Runge-Kutta

Théorème 21. [46, Theorem 1.4 p.51] La solution numérique d'une méthode de Runge-Kutta (3.3) est la série B associée à l'application a , telle que :

$$a(\emptyset) = 1; \quad a(t) = \sum_{i=1}^s b_i \Phi_i(t), \quad \forall t \in T;$$

où Φ_i est défini comme suit :

- $\Phi_i(\tau) = 1$
- $\Phi_i([t_1 t_2 \dots t_s]) = \sum_{j_1, \dots, j_m=1}^s a_{ij_1} \dots a_{ij_m} \Phi_{j_1}(t_1) \dots \Phi_{j_m}(t_m)$

3.4 Conditions d'ordre

3.4.1 En terme d'arbre

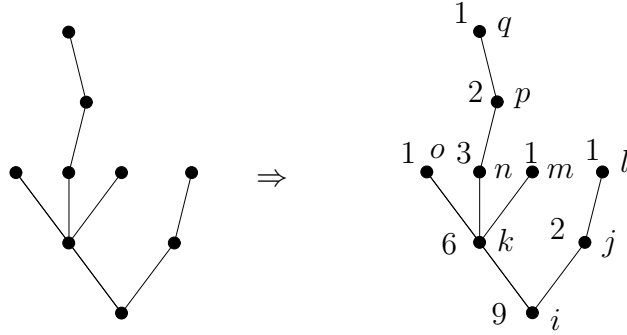
En comparant les séries B de la solution exacte et de la solution de Runge-Kutta, nous obtenons les conditions d'ordre p :

$$\sum_{i=1}^s b_i \Phi_i(t) = \frac{1}{\gamma(t)} \quad \forall t \in T, \quad \rho(t) \leq p \quad (3.9)$$

3.4.2 Méthode pratique pour le calcul des conditions d'ordre

Étant donné un arbre, on l'habille d'indices, avec l'indice i pour la racine, et de valeurs correspondant à l'ordre du sous-arbre ayant pour racine le sommet courant.

Exemple 22. Choisissons un arbre d'ordre 9 quelconque puis habillons le.



Ainsi décoré, il est simple de construire l'expression de Φ_i et de γ et ainsi la condition d'ordre associée :

$$\sum_{i,j,k,l,m,n,o,p,q} b_i a_{ij} a_{jl} a_{ik} a_{km} a_{kn} a_{np} a_{pq} a_{ko} = \frac{1}{9 \cdot 2 \cdot 6 \cdot 3 \cdot 2}$$

3.4.3 Méthode de collocation

Si une méthode de Runge-Kutta à s étages dérive d'une méthode de collocation, alors elle est au moins d'ordre s car elle est exacte si la solution est un polynôme de degré au plus s , puisque l'on choisit s points distincts de collocation.

Théorème 23. [46, Theorem 1.5 p.28] Pour une méthode de collocation, si la relation $B(p)$ est vérifiée pour $p > s$, alors la méthode est d'ordre p , p étant borné par $2s$.

Exemple 24. [46, p.30] Si l'on choisit les c_i racines du polynôme de Legendre :

$$\frac{d^s}{dx^s} (x^s (x-1)^s)$$

alors la méthode est d'ordre $2s$. Pour $s = 1$, on obtient la méthode du point milieu, et pour $s = 2$ et 3 , les méthodes de Gauss d'ordre 4 et 6, dont les coefficients de Runge-Kutta sous la forme de tableau de Butcher sont donnés

ci-dessous.

$\frac{1}{2} - \frac{\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{1}{4} - \frac{\sqrt{3}}{6}$	$\frac{1}{2} - \frac{\sqrt{15}}{10}$	$\frac{5}{36}$	$\frac{2}{9} - \frac{\sqrt{15}}{15}$	$\frac{5}{36} - \frac{\sqrt{15}}{30}$
$\frac{1}{2} + \frac{\sqrt{3}}{6}$	$\frac{1}{4} + \frac{\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{5}{36} + \frac{\sqrt{15}}{24}$	$\frac{2}{9}$	$\frac{5}{36} - \frac{\sqrt{15}}{24}$
			$\frac{1}{2} + \frac{\sqrt{15}}{10}$	$\frac{5}{36} + \frac{\sqrt{15}}{30}$	$\frac{2}{9} + \frac{\sqrt{15}}{15}$	$\frac{5}{36}$
	$\frac{1}{2}$	$\frac{1}{2}$		$\frac{5}{18}$	$\frac{4}{9}$	$\frac{5}{18}$

3.5 Précision de résolution et condition d'ordre pour les méthodes de Runge-Kutta partitionnées

3.5.1 Motivation

Développons la solution exacte du système partitionné :

$$\begin{aligned}
 p'(t_0) &= f(p(t_0), q(t_0)) \\
 q'(t_0) &= g(p(t_0), q(t_0)) \\
 p''(t_0) &= f_p(p(t_0), q(t_0))f(p(t_0), q(t_0)) + f_q(p(t_0), q(t_0))g(p(t_0), q(t_0)) \\
 q''(t_0) &= g_p(p(t_0), q(t_0))f(p(t_0), q(t_0)) + g_q(p(t_0), q(t_0))g(p(t_0), q(t_0))
 \end{aligned}$$

Nous pouvons le réécrire sans perte d'information sous la forme :

$$\begin{aligned}
 p'(t_0) &= f \\
 q'(t_0) &= g \\
 p''(t_0) &= f_p f + f_q g \\
 q''(t_0) &= g_p f + g_q g \\
 p^{(3)}(t_0) &= f_{pp}(f, f) + 2f_{pq}(f, g) + f_{qq}(g, g) \\
 &\quad f_p(f_p(f)) + f_p(f_q(g)) + f_q(g_p(f)) + f_q(g_q(g)) \\
 q^{(3)}(t_0) &= g_{pp}(f, f) + 2g_{pq}(f, g) + g_{qq}(g, g) \\
 &\quad g_p(f_p(f)) + g_p(f_q(g)) + g_q(g_p(f)) + g_q(g_q(g))
 \end{aligned}$$

Arbre	t	$\rho(t)$	$F(t)$	Arbre	t	$\rho(t)$	$F(t)$
\bullet	τ_p	1	f	\bullet	$[[\tau_p]_p]_p$	3	$f_p(f_p(f))$
\circ	τ_q	1	g	\circ	$[[\tau_q]_p]_p$	3	$f_p(f_q(f))$
\bullet	$[\tau_p]_p$	2	$f_p(f)$	\bullet	$[[\tau_p]_q]_p$	3	$f_q(g_p(f))$
\circ	$[\tau_q]_p$	2	$f_q(g)$	\bullet	$[[\tau_p]_p]_q$	3	$g_p(f_p(f))$
\bullet	$[\tau_p]_q$	2	$g_p(f)$	\circ	$[[\tau_q]_q]_p$	3	$f_q(g_q(g))$
\circ	$[\tau_q]_q$	2	$g_q(g)$	\bullet	$[[\tau_q]_p]_q$	3	$g_p(f_q(g))$
\bullet	$[\tau_p \tau_p]_p$	3	$f_{pp}(f, f)$	\bullet	$[[\tau_p]_q]_q$	3	$g_q(g_p(f))$
\circ	$[\tau_q \tau_p]_p$	3	$f_{qp}(g, f)$	\circ	$[[\tau_q]_q]_q$	3	$g_q(g_q(g))$
\bullet	$[\tau_q \tau_q]_p$	3	$f_{qq}(g, g)$				
\bullet	$[\tau_p \tau_p]_q$	3	$g_{pp}(f, f)$				
\circ	$[\tau_q \tau_p]_q$	3	$g_{qp}(g, f)$				
\circ	$[\tau_q \tau_q]_q$	3	$g_{qq}(g, g)$				

Table 3.2: Arbre bicolore, Densité, Différentielle élémentaire

3.5.2 Définition formelle des arbres bicolores

Définition 25. L'ensemble des *arbres bicolores* TP est défini inductivement par :

- L'arbre τ_p composé d'un seul sommet noir et dessiné : \bullet appartient à TP ,
- L'arbre τ_q composé d'un seul sommet blanc et dessiné : \circ appartient à TP ,
- Si (t_1, t_2, \dots, t_m) sont des arbres, alors l'arbre composé d'une racine noire se connectant à tous les arbres t_i par un arc, est un élément de TP . Nous le noterons $[t_1 \dots t_m]_p$
- Si (t_1, t_2, \dots, t_m) sont des arbres, alors l'arbre composé d'une racine blanche se connectant à tous les arbres t_i par un arc, est un élément de TP . Nous le noterons $[t_1 \dots t_m]_q$

Notation 26. Nous noterons l'ensemble des arbres bicolores de racine τ_p , TP_p , et ceux de racine τ_q , TP_q .

3.5.3 Séries P

Définition 27. Étant donnée une application $a : TP \cup \{\emptyset_p, \emptyset_q\} \rightarrow \mathbb{R}$, une série formelle de la forme :

$$P(a, (p, q)) = \begin{pmatrix} a(\emptyset_p)p + \sum_{t \in TP_p} \frac{(t-t_0)^{\rho(t)}}{\sigma(t)!} a(t)F(t)(p, q) \\ a(\emptyset_q)q + \sum_{t \in TP_q} \frac{(t-t_0)^{\rho(t)}}{\sigma(t)!} a(t)F(t)(p, q) \end{pmatrix}$$

sera appelée une *série P*.

Lemme 28. [46, Lemma 2.2 p.64] Soit une application $a : TP \cup \{\emptyset_p, \emptyset_q\} \rightarrow \mathbb{R}$ vérifiant : $a(\emptyset_p) = 1$ et $a(\emptyset_q) = 1$, alors nous avons :

$$h \begin{pmatrix} f(P(a, (p, q))) \\ g(P(a, (p, q))) \end{pmatrix} = P(a', (p, q))$$

avec $a'(\emptyset_p) = 0, a'(\emptyset_q) = 0, a'(\tau_p) = 1, a'(\tau_q) = 1$ et

$$a'([t_1 t_2 \dots t_s]_p) = a'([t_1 t_2 \dots t_s]_q) = a(t_1)a(t_2)\dots a(t_s)$$

3.5.4 Série P de la solution exacte d'un système partitionné

Théorème 29. La solution exacte d'un système partitionné (3.2) est la série P associée à l'application e telle que :

$$e(\emptyset_p) = e(\emptyset_q) = 1, \quad e(t) = \frac{1}{\gamma(t)} \quad \forall t \in T;$$

3.5.5 Série P de la solution numérique de Runge-Kutta partitionnée

Théorème 30. La solution numérique d'une méthode de Runge-Kutta partitionnée (3.6) est la série P associée à l'application a , telle que :

- $a(\emptyset_p) = a(\emptyset_q) = 1,$
- $a(t) = \begin{cases} \sum_{i=1}^s b_i \Phi_i(t) & \forall t \in TP_p, \\ \sum_{i=1}^s \hat{b}_i \Phi_i(t) & \forall t \in TP_q, \end{cases}$

où Φ_i est défini comme suit :

- $\Phi_i(\tau_p) = \Phi_i(\tau_q) = 1,$
- $\Phi_i(t) = \Psi_i(t_1) \dots \Psi_i(t_m),$
- $\Psi_i(t_k) = \begin{cases} \sum_{j_k=1}^s a_{ij_k} \Phi_{j_k}(t_k) & \text{si } t_k \in TP_p, \\ \sum_{j_k=1}^s \hat{a}_{ij_k} \Phi_{j_k}(t_k) & \text{si } t_k \in TP_q. \end{cases}$

3.6 Conditions d'ordre pour les schémas partitionnés

3.6.1 En terme d'arbre

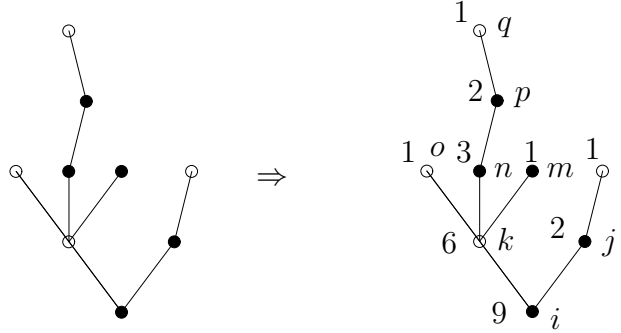
En comparant les séries P de la solution exacte et de la solution de Runge-Kutta, nous obtenons les conditions d'ordre n :

$$\begin{aligned} \sum_{i=1}^s b_i \Phi_i(t) &= \frac{1}{\gamma(t)} & \forall t \in TP_p, \quad \rho(t) \leq n \\ \sum_{i=1}^s \hat{b}_i \Phi_i(t) &= \frac{1}{\gamma(t)} & \forall t \in TP_q, \quad \rho(t) \leq n \end{aligned} \quad (3.10)$$

3.6.2 Méthode pratique pour le calcul des conditions d'ordre

Étant donné un arbre, on l'habille d'indices, avec l'indice i pour la racine, et de valeur correspondant à l'ordre du sous-arbre ayant pour racine le sommet courant.

Exemple 31. Choisissons un arbre d'ordre 9 quelconque puis habillons le.



Ainsi habillé il est simple de construire l'expression de Φ_i et de γ et ainsi la condition d'ordre associée :

$$\sum_{i,j,k,l,m,n,o,p,q} b_i a_{ij} \hat{a}_{jl} \hat{a}_{ik} a_{km} a_{kn} a_{np} \hat{a}_{pq} \hat{a}_{ko} = \frac{1}{9 \cdot 2 \cdot 6 \cdot 3 \cdot 2}$$

3.7 Les méthodes de Runge-Kutta symplectique

Les schémas d'intégration symplectique sont des méthodes d'intégration adaptées aux systèmes Hamiltonien. En effet, elles possèdent de nombreux avantages remarquables dont la propriété de concerver les intégrales premières du système.

Les méthodes de Runge-Kutta simples ou partitionnées peuvent être symplectiques; si cela est le cas, les coefficients vérifient des égalités supplémentaires. Pour les schémas partitionnés celles-ci sont :

$$\hat{b}_i = b_i, \quad \hat{a}_{ij} = b_j - \frac{b_j}{b_i} a_{ji}, \quad \text{pour tout } i = 1, \dots, s \text{ et } j = 1, \dots, s. \quad (3.11)$$

Et dans le cas non-partitionné :

$$b_i = b_i, \quad a_{ij} = b_j - \frac{b_j}{b_i} a_{ji}, \quad \text{pour tout } i = 1, \dots, s \text{ et } j = 1, \dots, s. \quad (3.12)$$

Dans ces circonstances certaines conditions d'ordre obtenues précédemment, deviennent redondantes entre elles. Pour le cas non partitionné les conditions indépendantes sont décrites dans l'article de J.M.Sanz-Serna [67] et une méthode de calcul des conditions d'ordre dans le cas partitionné fait l'objet du chapitre suivant.

Chapitre 4

Computation of order conditions for symplectic partitioned Runge-Kutta schemes with application to optimal control

We discuss the derivation of order conditions for the discretization of (unconstrained) optimal control problems, when the scheme for the state equation is of Runge-Kutta type.

This problem appears to be essentially the one of checking order conditions for symplectic partitioned Runge-Kutta schemes. We show that the computations using bi-coloured trees are naturally expressed in this case in terms of *oriented free tree*. This gives a way to compute them by an appropriate computer program.

We have implemented an algorithm which is able to compute conditions up to order 7 (we display them up to order 6). The results are in accordance with those of Hager (where they were computed for order up to 4) as well as those of Murua where the number of conditions up to order 7 is stated.

4.1 Introduction

The motivation of this work comes from an analysis by Hager [45] of order conditions for optimal control problems (of an ordinary differential equation). The idea is to discretize the state equation by a Runge-Kutta scheme, with a different value of control associated with each “inner state”. Hager observes that the resulting optimality system, after some change of variable, is a partitioned Runge-Kutta scheme. He computes then (by hand, i.e., without

computer code) the order conditions up to the 4th order. See also the results of [30] and [31] on constrained optimal control problems (a first order state constrained problem, discretized by Euler's scheme, and a control constrained problem with a Runge-Kutta discretization).

There are essentially two hypotheses in the analysis of Hager in [45], one on the original problem and the other is a restriction on the scheme. One has to assume that the Hamiltonian is strongly convex w.r.t. the control, or more generally that the second derivative of Hamiltonian w.r.t. the control is invertible. This allows to eliminate the control thanks to the implicit function theorem, so that we have an equivalent scheme for the reduced (state, adjoint state) system. The second hypothesis is that none of the coefficients b_i 's (of the particular Runge-Kutta scheme) is zero.

The main result of Hager [45] is that, if the original Runge-Kutta scheme is of (global) order p (i.e., when applied to an uncontrolled differential equation) then the resulting scheme has order $q \leq p$, with equality if $p \leq 2$ but strict inequality in some cases whenever $p \geq 3$. In addition, $q = p$ if the scheme is explicit of order at most 4.

For order greater than four, one cannot do computations by hand. It is then useful to rely on the theory of order conditions for partitioned Runge-Kutta scheme. This theory, based on bicolour rooted trees with which are associated certain numbers, is an extension of the original work by Butcher for (non partitioned) Runge-Kutta schemes, see Butcher [24, p. 88].

It appears that the class of partitioned Runge-Kutta schemes coming from the discretization of optimal control problems are in fact partitioned symplectic Runge-Kutta schemes, characterized by relation (4.4) below. So the question boils down to the one of expressing order conditions for this class. The main result of this chapter is that we can obtain the desired expressions using a "calculus" on oriented free trees. To be specific, with each oriented free tree are associated some weights, and the main operation is to "split" any rooted tree into a sum (with coefficients ± 1) of such oriented free trees.

As Sofroniou and Oevel in [69], the idea of the paper, is to give another definition of elementary weight on H-tree (oriented free tree), but unlike the paper of Sofroniou we keep natural free parameters $(a_{ij}, b_i, \hat{a}_{ij}, \hat{b}_i)$ of our PRK method. Moreover, our study deals with PRK and not symplectic schemes where $a_{ij} = \hat{a}_{ij}$ and $b_i = \hat{b}_i$.

The chapter is organized as follows. In the next section we detail the discretization of optimal control problems by Runge-Kutta schemes, and show the relation with partitioned symplectic Runge-Kutta schemes satisfying (4.4). Then in section 4.3 we review the theory of order conditions for partitioned Runge-Kutta schemes. Section 4.4 introduces oriented free trees, and shows how the order conditions can be expressed in terms of the latter.

Finally section 4.5 discusses the implementation, and displays the results for order up to 5 and the number of conditions for order up to 7.

4.2 Discretization of unconstrained optimal control problems

Let f and Φ be C^∞ functions $\mathbb{R}^m \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $\mathbb{R}^n \rightarrow \mathbb{R}$, respectively. Let us consider the following unconstrained optimal control problem:

$$\begin{cases} \text{Min } \Phi(y(T)); \\ \dot{y}(t) = f(u(t), y(t)), & t \in [0, T]; \\ y(0) = y^0. \end{cases} \quad (P)$$

We restrict the analysis to continuous control functions. Let us denote by $H(u, y, p) := p \cdot f(u, y)$ the *pseudo-Hamiltonian* of the problem. The first order necessary optimality conditions of this problem may be written in the following form:

$$\left\{ \begin{array}{l} \dot{y}(t) = f(u(t), y(t)), \\ \dot{p}(t) = -H_y(u(t), y(t), p(t)), \\ 0 = H_u(u(t), y(t), p(t)), \\ p(T) = \Phi'(y(T)), \quad y(0) = y^0. \end{array} \right\} \quad t \in [0, T], \quad (OC)$$

We say that $(\bar{u}, \bar{y}, \bar{p})$ is an extremum if it satisfies (OC) (\bar{u} being a continuous function). Let $(\bar{u}, \bar{y}, \bar{p})$ be an extremum. If

$$u \mapsto H_{uu}(u, y, p) \quad \text{is invertible along the trajectory,} \quad (4.1)$$

then by the implicit functions theorem, in a small L^∞ neighborhood of this trajectory, we have that $H_u(u(t), y(t), p(t)) = 0$ if and only if $u = \phi(y(t), p(t))$, where ϕ is a C^∞ mapping. Let us define the *true Hamiltonian* as $\mathcal{H}(y, p) := H(\phi(y, p), y, p)$. Using $H_u(\phi(y(t), p(t)), y(t), p(t)) = 0$, we obtain

$$\mathcal{H}_y(y, p) = H_y(y, p); \quad \mathcal{H}_p(y, p) = H_p(y, p). \quad (4.2)$$

Consequently, under hypothesis (4.1), (OC) is locally equivalent to the *reduced Hamiltonian system*

$$\begin{cases} \dot{y}(t) = \mathcal{H}_p(\phi(y, p), y, p), & t \in [0, T], \\ -\dot{p}(t) = \mathcal{H}_y(\phi(y, p), y, p), & t \in [0, T], \\ p(T) = \Phi'(y(T)), \quad y(0) = y^0. \end{cases} \quad (4.3)$$

We now turn to the discussion of the discretization of the optimal control problem (P) . The Runge-Kutta discretization considered in [45] is

$$\begin{cases} \text{Min } \Phi(y_N); \\ y_{k+1} = y_k + h_k \sum_{i=1}^s b_i f(u_{ki}, y_{ki}), \\ y_{ki} = y_k + h_k \sum_{j=1}^s a_{ij} f(u_{kj}, y_{kj}), \\ y_0 = y^0, \end{cases} \quad (DP_1)$$

for all $k = 0$ to $N - 1$ and $i = 1$ to s . Here $h_k > 0$ is the size of the k th time step, and (a, b) is the set of Runge-Kutta coefficients. Let us underline the choice of using different values of controls u_{kj} associated with inner states y_{kj} ; this contrasts with other approaches in which the discretization of controls is coarser than the one of the state, i.e., for control taken constant, or only affine on each time step (e.g. [7, 8, 16]).

We may rewrite (DP_1) under the equivalent form

$$\begin{cases} \text{Min } \Phi(y_N); \\ 0 = h_k \sum_{i=1}^s b_i K_{ki} + y_k - y_{k+1}, \\ 0 = f(u_{ki}, y_k + h_k \sum_{j=1}^s a_{ij} K_{kj}) - K_{ki}, \\ 0 = y^0 - y_0. \end{cases} \quad (DP_2)$$

In the expression below we contract $y_k + h_k \sum_{j=1}^s a_{ij} K_{kj}$ into y_{ki} . The Lagrangian function associated with (DP_2) is:

$$\Phi(y_N) + p^0 \cdot (y^0 - y_0) + \sum_{k=0}^{N-1} \left\{ p_{k+1} \cdot \left(h_k \sum_{i=1}^s b_i K_{ki} + y_k - y_{k+1} \right) + \sum_{i=1}^s \xi_{ki} \cdot (f(u_{ki}, y_{ki}) - K_{ki}) \right\}.$$

Here p_{k+1} , ξ_{ki} , and p^0 are the Lagrange multipliers associated with the constraints of (DP_2) . Variables p_k will be interpreted as the discretization of co-state of continuous formulation. Setting to zero the derivative of this Lagrangian function w.r.t. the primal variables y_N , y_0 , y_k for $k = 1$ to $N - 1$, K_{ki} and u_{ki} for $k = 0 \dots N - 1$, $i = 1 \dots s$, we obtain

$$\begin{aligned} p_N &= \Phi'(y_N), \\ p_1 &= p^0, \\ p_k - p_{k+1} &= \sum_{i=1}^s f_y(u_{ki}, y_{ki})^\top \xi_{ki}, \\ 0 &= h_k b_i p_{k+1} + h_k \sum_{j=1}^s a_{ji} f_y(u_{kj}, y_{kj})^\top \xi_{kj} - \xi_{ki}, \\ 0 &= f_u(u_{ki}, y_{kj})^\top \xi_{ki}, \quad k = 0 \dots N - 1, \quad i = 1 \dots s. \end{aligned}$$

Using now the hypothesis that $b_i \neq 0$, set $p_{ki} := \xi_{ki}/(h_k b_i)$ for all $k = 0$ to $N - 1$, and $i = 1$ to s . Eliminating ξ_{ki} from the above relations, we obtain the equivalent optimality conditions

$$\begin{cases} y_{k+1} = y_k + h_k \sum_{i=1}^s b_i f(u_{ki}, y_{ki}), \\ y_{ki} = y_k + h_k \sum_{j=1}^s a_{ij} f(u_{kj}, y_{kj}), \\ p_{k+1} = p_k - h_k \sum_{i=1}^s \hat{b}_i H_y(u_{ki}, y_{ki}, p_{ki}), \\ p_{ki} = p_k - h_k \sum_{j=1}^s \hat{a}_{ij} H_y(u_{kj}, y_{kj}, p_{kj}), \\ 0 = H_u(u_{ki}, y_{ki}, p_{ki}), \\ y_0 = y^0, \quad p_N = \Phi'(y_N), \end{cases} \quad (DOC)$$

where coefficients \hat{b} and \hat{a} are defined by the following relations:

$$\hat{b}_i := b_i, \quad \hat{a}_{ij} := b_j - \frac{b_j}{b_i} a_{ji}, \quad \text{for all } i = 1, \dots, s \text{ and } j = 1, \dots, s. \quad (4.4)$$

If the algebraic constraints $H_u(u_{ki}, y_{ki}, p_{ki}) = 0$ are locally equivalent to $u_{ki} = \phi(y_{ki}, p_{ki})$, then (DOC) is equivalent to the same partitioned Runge-Kutta scheme applied to the reduced system (4.3). This approach based on formulation (DP_2) is slightly simpler, but equivalent to the one of Hager [45].

It is said that a partitioned Runge-Kutta scheme (or more generally any one step scheme) is symplectic if the corresponding flow is symplectic. It is known that partitioned Runge-Kutta schemes satisfying (4.4) are symplectic, see [49, Theorem 4.6]. We obtain that the scheme obtained by discretization of problem (P) is symplectic. In particular the following diagram commutes, when we use the above discretization:

$$\begin{array}{ccc} (P) & \xrightarrow{\text{discretization}} & (DP) \\ \text{optimality} \downarrow & & \text{optimality} \downarrow \\ \text{conditions} & & \text{conditions} \end{array} \quad (D)$$

$$(OC) \xrightarrow{\text{discretization}} (DOC)$$

For a detailed presentation of partitioned Runge-Kutta and symplectic methods we refer to the books of Hairer [46, 47].

4.3 Order conditions for partitioned Runge-Kutta schemes

A basic tool in the study of order conditions for Runge-Kutta schemes is the theory of B-series and associated calculus on rooted trees, due to Butcher

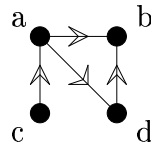
(see [24]). For partitioned Runge-Kutta schemes an extension of this theory is the one of expansion in P-series, and the associated calculus on bicoloured rooted trees, see [46]. The latter allows to state the order conditions in terms of coefficients a , b , \hat{a} , and \hat{b} , of the following type: certain polynomials (which are in fact sum of monomials with unit coefficients) in these variables have to be equal to certain fractional numbers. The substitution of \hat{a} , and \hat{b} (using (4.4)) would give complicated expressions, since instead of each monomial we would have a sum of rational fractions. We will show that among all these terms it is sufficient to express a condition on a “principal term” since the other terms of the sum are already determined by previous conditions. This allows a tremendous simplification that permits us to display these conditions up to order 5, in terms of a and b alone.

Remark 1. To follow the track of Murua in [63], we could build H-serie, by defining an elementary differential of an H-tree, then build the serie of the exact solution for the hamiltonian system and the one of the numerical solution from PRK, and finally compare term to term the coefficients, and thanks to lineary independency we get the order condition. But to do this we need to solve the difficult problem to give a good definition of the elementary differential of an H-tree.

In order to fix notation we will need some definitions. A *graph* G is a pair (V, E) where V has a finite cardinal and $E \subset \mathcal{P}(V)$. Elements of V (resp. E) are called vertices (resp. edges). Here $\mathcal{P}(V)$ denotes the set of subsets of V . The set of *graphs* will be denoted $\tilde{\mathcal{G}}$.

An *oriented graph* G is a pair (V, E) where V is as before and $E \subset V \times V$; elements of E are called oriented edges. The set of *oriented graph* will be denoted \mathcal{G} .

Example 2. $V = \{a, b, c, d\}$ and $E = \{(a, b), (c, a), (d, b), (a, d)\}$



A path of a graph is a finite sequence of vertices (at least two) such that there exists an edge between two successive vertices. A graph is *connected* if any pair of vertices are member of at least one path. A *circuit* is a path whose first and last elements coincide. A *free tree* t is a connected graph without circuits. The set of free trees will be denoted $\tilde{\mathcal{T}}$. A *rooted tree* t is a pair of a tree and of one of its vertices, called the root, and is denoted $r(t)$. The

leaves are the vertices with only one adjacent edge, other than the root. We may identify a rooted tree with an oriented graph, where all edges point in direction of leaves.

The set of *rooted trees* will be denoted \mathcal{T}^* . We sometimes speak of free trees, as opposed to rooted trees. We define in a similar manner oriented trees, also called H-trees in [63]. The set of *oriented free trees* is denoted \mathcal{T} .

Bicoloured graphs are graphs associated with a mapping which to each vertex v associates a colour $c(v)$, of value B or W (black or white). The set of *bi-coloured oriented graphs* (resp. *bi-coloured rooted trees*) will be denoted \mathcal{BG} (resp. \mathcal{BT}^*).

We denote $V_B = c^{-1}(\{B\})$ (resp. $V_W = c^{-1}(\{W\})$) the set of black (resp. white) vertices and E_B (resp. E_W) the set of edges ending on black (resp. white) vertex.

For a given rooted tree, whose vertices are associated with letters i, j, \dots it is convenient to denote

$$\begin{aligned} \tilde{b}_{i_k} &= b_{i_k} \text{ if vertex } k \text{ is white, } \hat{b}_{i_k} \text{ otherwise.} \\ \tilde{a}_{i_k i_\ell} &= a_{ij} \text{ if vertex } \ell \text{ is white, } \hat{a}_{ij} \text{ otherwise.} \end{aligned} \quad (4.5)$$

Here i_k associates with each vertex $k \in \{1, \dots, \#t\}$ a variable i_k (varying from 1 to s in the subsequent expressions).

With each rooted tree t is associated a value $\gamma(t)$ in an inductive way: $\gamma(\bullet) = 1$, and

$$\gamma(t) = \#t \times \gamma(t_1) \times \dots \times \gamma(t_m), \quad (4.6)$$

where by t_1, \dots, t_m we denote the branches of t (the connected graphs obtained by removing the root from t , viewed as rooted trees, the root being the vertex whose single adjacent edge has been removed).

Next, we need to recall the definition of the elementary weight of a tree t , where t is a bicoloured rooted graph, for given Runge-Kutta coefficients a and b :

$$\phi(t) = \sum_{i=1}^s \tilde{b}_i \phi_i(t). \quad (4.7)$$

The above functions $\phi_i(t)$ are themselves defined in an inductive way, by $\phi_i(\bullet) = \phi_i(\circ) = 1$ and

$$\begin{aligned} \phi_i(t) &= \psi_i(t_1) \times \dots \times \psi_i(t_m), \\ \psi_i(t_k) &= \sum_{j_k=1}^s \tilde{a}_{ij_k} \phi_{j_k} \end{aligned} \quad (4.8)$$

We have the following result (see [46, Thm III.2.5]):

Theorem 3. A partitioned Runge-Kutta scheme has (global) order q if and only if

$$\phi(t) = \frac{1}{\gamma(t)}, \quad \text{for all } t \in \mathcal{BT}^*, \quad \#t \leq q. \quad (4.9)$$

Substituting the expressions of \hat{a} and \hat{b} in (4.4) in the formula of $\phi(t)$ would give complicated expressions. We show in the next section how to state equivalent, but simpler conditions.

4.4 Calculus on oriented free trees

In this section we choose as convention that the set of vertices V is of the form : $\{1, \dots, \#V\}$ we do not lose generality. Then we start by stating alternative expressions of the elementary weight $\phi(t)$, where t is a rooted bicoloured tree, defined in 4.3, for a general partitioned scheme. Taking into account the definition of functions $\psi_i(t_k)$, we obtain the more explicit expression

$$\phi(t) = \sum_{i_1=1}^s \tilde{b}_{i_1} \sum_{i_2, \dots, i_{\#t}=1}^s \prod_{(x,y) \in E} \tilde{a}_{i_x i_y}. \quad (4.10)$$

Here again i_k associates with each vertex $k \in \{1, \dots, \#t\}$ (vertex 1 is the root) a variable i_k (varying from 1 to s). Formula (4.10) may be written as

$$\phi(t) = \sum_{v \in V} \sum_{i_v=1}^s \prod_{k \in V} \tilde{b}_{i_k} \prod_{(x,y) \in E} \frac{\tilde{a}_{i_x i_y}}{\tilde{b}_{i_y}}. \quad (4.11)$$

Indeed, all \tilde{b}_{i_ℓ} in the above expression vanish except \tilde{b}_{i_1} , so that (4.10) and (4.11) coincide. Observe, however, that the above formula makes sense for (non necessarily connected) bi-coloured oriented graphs. Any such graph t is a finite union of connected graphs with empty intersection of vertices, called *connected components* of t , and denoted $\{t^q, q \in Q\}$ (not to be confused with branches of a rooted tree, denoted t_i).

Lemma 4. The elementary weight of a bi-coloured oriented graph $t = (V, E, c)$, with connected components $\{t^q, q \in Q\}$, is the product of the elementary weights of its connected components, i.e.,

$$\phi(t) = \prod_{q \in Q} \phi(t^q). \quad (4.12)$$

Proof. We have :

$$\begin{aligned}
\phi(t) &= \sum_{v \in V} \sum_{i_v=1}^s \prod_{k \in V} \tilde{b}_{i_k} \prod_{(x,y) \in E} \frac{\tilde{a}_{i_x i_y}}{\tilde{b}_{i_y}} \\
&= \sum_{v \in V} \sum_{i_v=1}^s \prod_{q \in Q} \left(\prod_{k \in V_q} \tilde{b}_{i_k} \prod_{(x,y) \in E_q} \frac{\tilde{a}_{i_x i_y}}{\tilde{b}_{i_y}} \right) \\
&= \prod_{q \in Q} \left(\sum_{v \in V_q} \sum_{i_v=1}^s \left(\prod_{k \in V_q} \tilde{b}_{i_k} \prod_{(x,y) \in E_q} \frac{\tilde{a}_{i_x i_y}}{\tilde{b}_{i_y}} \right) \right),
\end{aligned}$$

where the last equality uses the identity (valid for arbitrary families of sets I_q and functions A_q)

$$\prod_{q \in Q} \left(\sum_{i \in I_q} A_q(i) \right) = \sum_{r \in Q} \sum_{i \in I_r} \left(\prod_{q \in Q} A_q(i) \right). \quad (4.13)$$

The conclusion follows. \square

Given an oriented graph $t = (V, E)$, and $F \subset E$, the set of arcs in opposite direction to those of F is denoted as

$$F^\top := \{(x, y) \in V \times V; (y, x) \in F\}. \quad (4.14)$$

Theorem 5. The elementary weight of a bi-coloured oriented graph $t = (V, E, c)$, when (4.4) holds, satisfies

$$\phi(t) = \sum_{\hat{E}_B \in \mathcal{P}(E_B)} (-1)^{\#\hat{E}_B} \phi(V, E_W \cup \hat{E}_B^\top). \quad (4.15)$$

Proof. Substituting the expressions of \hat{a} and \hat{b} in (4.4), we may write elementary weights as follows:

$$\phi(t) = \sum_{v \in V} \sum_{i_v=1}^s \prod_{k \in V} b_{i_k} \prod_{(x,y) \in E_W} \frac{a_{i_x i_y}}{b_{i_y}} \prod_{(x,y) \in E_B} \left(1 - \frac{a_{i_y i_x}}{b_{i_x}} \right). \quad (4.16)$$

Expanding the last term, get

$$\phi(t) = \sum_{\hat{E}_B \in \mathcal{P}(E_B)} (-1)^{\#\hat{E}_B} \sum_{v \in V} \sum_{i_v=1}^s \prod_{k \in V} b_{i_k} \prod_{(x,y) \in E_W} \frac{a_{i_x i_y}}{b_{i_y}} \prod_{(x,y) \in \hat{E}_B} \frac{a_{i_y i_x}}{b_{i_x}}. \quad (4.17)$$

The conclusion follows. \square

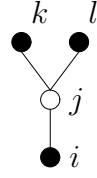


Figure 4.1: Bi-coloured tree t

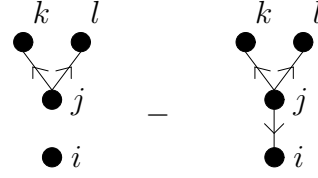


Figure 4.2: $t_1 - t_2$

Our procedure computes order conditions in a recursive way. When dealing with order p , all conditions of order smaller than p have already been obtained. We claim that elementary weights of all terms in the sum of (4.15) have already been computed, except (perhaps) for the one in which $\hat{E}_B = E_B$.

Indeed, for an arc in $E_B \setminus \hat{E}_B$ then there is no contribution from this arc: we can interpret this as a deletion of the arc, whereas for an arc in E_B , we obtain the usual term (for a non partitioned Runge-Kutta scheme) except that we have $a_{i_\ell i_k}$ instead of $a_{i_k i_\ell}$, which would be the usual term if the direction of arc had been changed, and an associated minus sign.

Now whenever an arc is deleted, it follows from lemma 4 that the term is a product of elementary weights for trees of smaller size. Since black nodes have all possible locations, when conditions of order less than p were computed, the elementary weights of all trees of weight at most $p - 1$ have been computed.

The idea now is to focus on this graph interpretation, and compute order conditions. With theorem 4.15 we can associate to a bi-coloured tree a set of oriented graph for which only one is connected. This means that we can obtain order conditions for each graph. Let us now take a tutorial sample to show how the method works.

Example 6. This example illustrates how we can compute order conditions associated with a given graph. In this example t_1 from Figure 4.2 has two connected components, and hence, $\phi(t_1) = \sum_{i,j,k,l=1}^s b_i b_j a_{jk} a_{il}$ is the product of two elementary weights that have been already evaluated (when dealing with order 4 conditions). Therefore the new condition can be expressed as $\phi(t_2) = \sum_{i,j,k,l=1}^s b_j a_{ji} a_{jk} a_{il} = \phi(t_1) - \phi(t) = \phi(t_1) - 1/\gamma(t) = \phi(t_1) - 1/12$, where the right-hand-side reduces to a rational number.

Note that we get one order condition for each bi-coloured graph. Since there are less oriented free trees than bi-coloured trees, it may happen that all terms of the sum in (4.15) are already evaluated.

Remark 7. The above discussion is coherent with the result of Murua [63]: there exist as many order conditions for order n for an integration method as there exist oriented free tree with n nodes.

4.5 Implementation and computational results

4.5.1 Implementation

Given a rooted bicoloured tree t , denote by $(\sigma_i, g_i)_i$ the associated family of oriented graphs obtained by the operation of either cutting or reversing black edges; denote by g^* the connected element of this family (obtained by reversing all black edges). Let $v(g)$ be the value of the elementary weight, stored in a data base called **Bank**. Since elementary weights are rational numbers, we store them as a pair of irreducible integers. The procedure for obtaining order conditions at a given order n may be written as

```

OrderGen(n)
For  $t \in \mathcal{BT}^*$  ordered by increasing degree, with  $|t| \leq n$ 
  Compute the family of oriented graphs  $(\sigma_i, g_i)_i$ 
  Evaluate elementary weights of all no-connected  $g_i$ 
  If  $g^* \in \mathbf{Bank}$ 
    Then check if  $\sum_i \sigma_i v(g_i) + \sigma^* v(g^*) = 1/\gamma(t)$ 
    Else  $\mathbf{Bank} \leftarrow \mathbf{Bank} \cup \{v(g^*) = \sigma^*(1/\gamma(t) - \sum_i \sigma_i v(g_i))\}$ 
End For
Return Bank

```

Remark 8. The Gauss-Legendre Runge-Kutta method (in which a and b are equal to \hat{a} and \hat{b}) is known to be symplectic, of order $2s$ where s is the number of stages. Therefore the check of compatibility conditions that occurs when g^* already belongs to **Bank** has to be satisfied. We use it only as a debugging tool.

Our implementation uses the program for trees generation of Li and Ruskey [57]. Then we build all possible colorations for a given tree, and apply procedure **OrderGen**. The code is written in the C language. The coloration procedure is as follows:

```

ColorGen(t)
If  $t = \tau$  return [W,B]
Else Express  $t$  as links from root to branches  $t = [t_1^{n_1} \dots t_s^{n_s}]$ 
  For  $i=1$  to  $s$  ColorGen( $t_i$ )
  Combine all coloration into Tab
Return(Tab)

```

Order	1	2	3	4	5	6	7
Simple	1	1	2	4	9	20	48
Symplectic	1	1	3	8	27	91	350
Partitioned	2	4	14	52	214	916	4116

Table 4.1: Number of order conditions

Remark 9. Let $n_c(t_i)$ denote the number of coloration of branch t_i . Then the number of colorations of t is

$$\prod_{i=1}^s \frac{n_c(t_i)!}{n_i!(n_c(t_i) - n_i)!}$$

This explains the combinatorial explosion and the limitation of our result to $n = 7$.

Remark 10. Most of the computing time is spent in the search of a given graph (whenever it exists) in the bank. Our implementation checks bijection with all graphs of same number of vertices and edges. Obviously this procedure could be improved. For order greater than 6, conditions are too numerous to be displayed. However, it can be useful to formulate them in order to compute the order for specific values of a and b .

4.5.2 Computational result

We first display the number of order conditions in table 4.1. Observe the rapid increase of these numbers with p for symplectic schemes, and even more with general partitioned schemes.

In the next tables, we use the usual notations $d_j = \sum_i b_i a_{ij}$ and $c_i = \sum_j a_{ij}$. All indexes in the sum vary from 1 to s , where s is the number of stages in the Runge-Kutta method. All tables are generated automatically by the computer code. Conditions for order 1 to 4 were already obtained by Hager [45].

Table 4.2: Order 1

Graphe	Condition
•	$\sum b_i = 1$

Table 4.3: Order 2

Graphe	Condition
• ↔ •	$\sum d_j = \frac{1}{2}$

Table 4.4: Order 3




Graphe	Conditions	Graphe	Conditions
	$\sum c_j d_j = \frac{1}{6}$		$\sum b_i c_i^2 = \frac{1}{3}$
	$\sum \frac{1}{b_k} d_k^2 = \frac{1}{3}$		

Table 4.5: Order 4





















Graphe	Conditions	Graphe	Conditions
	$\sum \frac{1}{b_k} a_{lk} d_k d_l = \frac{1}{8}$		$\sum a_{jk} d_j c_k = \frac{1}{24}$
	$\sum \frac{b_i}{b_k} a_{ik} c_i d_k = \frac{5}{24}$		$\sum b_i a_{ij} c_i c_j = \frac{1}{8}$
	$\sum c_j^2 d_j = \frac{1}{12}$		$\sum b_i c_i^3 = \frac{1}{4}$
	$\sum \frac{1}{b_k} c_k d_k^2 = \frac{1}{12}$		$\sum \frac{1}{b_l^2} d_l^3 = \frac{1}{4}$

Table 4.6: Order 5

Graphe	Conditions	Graphe	Conditions
	$\sum \frac{b_i}{b_k} a_{ik} a_{il} d_k c_l = \frac{3}{40}$		$\sum b_i a_{ik} a_{ij} c_j c_k = \frac{1}{20}$
	$\sum a_{lk} a_{kj} c_j d_l = \frac{1}{120}$		$\sum b_i a_{ik} a_{kj} c_i c_j = \frac{1}{30}$
	$\sum \frac{b_i}{b_k} a_{lk} a_{il} c_i d_k = \frac{11}{120}$		$\sum \frac{b_i}{b_k} a_{lk} a_{ik} c_i d_l = \frac{3}{40}$
	$\sum \frac{b_i b_j}{b_k} a_{jk} a_{ik} c_i c_j = \frac{2}{15}$		$\sum \frac{b_i}{b_l b_m} a_{im} a_{il} d_l d_m = \frac{2}{15}$
	$\sum \frac{1}{b_k} a_{ml} a_{lk} d_k d_m = \frac{1}{30}$		$\sum \frac{1}{b_k} a_{mk} a_{lk} d_l d_m = \frac{1}{20}$
	$\sum \frac{1}{b_l b_m} a_{lm} d_l^2 d_m = \frac{1}{15}$		$\sum \frac{1}{b_k} a_{kl} d_k^2 c_l = \frac{1}{60}$





































Order 5

Graphe	Conditions	Graphe	Conditions
	$\sum \frac{1}{b_l^2} a_{ml} d_l^2 d_m = \frac{1}{10}$		$\sum \frac{b_i}{b_l^2} a_{il} c_i d_l^2 = \frac{3}{20}$
	$\sum \frac{1}{b_k} a_{lk} d_k c_l d_l = \frac{7}{120}$		$\sum a_{jk} c_j d_j c_k = \frac{1}{40}$
	$\sum \frac{1}{b_k} a_{lk} c_k d_k d_l = \frac{1}{40}$		$\sum \frac{b_i}{b_k} a_{ik} c_i c_k d_k = \frac{7}{120}$
	$\sum \frac{b_i}{b_k} a_{ik} c_i^2 d_k = \frac{3}{20}$		$\sum b_i a_{ij} c_i^2 c_j = \frac{1}{10}$
	$\sum a_{kj} c_j^2 d_k = \frac{1}{60}$		$\sum b_i a_{ij} c_i c_j^2 = \frac{1}{15}$
	$\sum c_j^3 d_j = \frac{1}{20}$		$\sum b_i c_i^4 = \frac{1}{5}$
	$\sum \frac{1}{b_k} c_k^2 d_k^2 = \frac{1}{30}$		$\sum \frac{1}{b_l^2} c_l d_l^3 = \frac{1}{20}$
	$\sum \frac{1}{b_m^3} d_m^4 = \frac{1}{5}$		





































Table 4.7: Order 6

Graphe	Conditions	Graphe	Conditions
	$\sum \frac{b_i}{b_l b_m} a_{im} a_{il} a_{nl} d_m d_n = \frac{7}{144}$		$\sum \frac{b_i}{b_k} a_{im} a_{ik} a_{lk} d_l c_m = \frac{19}{720}$
	$\sum \frac{1}{b_k} a_{nm} a_{mk} a_{lk} d_l d_n = \frac{1}{72}$		$\sum \frac{b_i}{b_k} a_{im} a_{mk} a_{lk} c_i d_l = \frac{13}{360}$
	$\sum \frac{1}{b_k} a_{nk} a_{mn} a_{lm} d_k d_l = \frac{1}{144}$		$\sum a_{lm} a_{kl} a_{jk} d_j c_m = \frac{1}{720}$
	$\sum \frac{b_i}{b_k} a_{ik} a_{mk} a_{lm} c_i d_l = \frac{7}{360}$		$\sum \frac{b_i}{b_l b_m} a_{im} a_{in} a_{nl} d_l d_m = \frac{1}{18}$
	$\sum \frac{b_i}{b_k} a_{im} a_{il} a_{lk} d_k c_m = \frac{13}{360}$		$\sum \frac{b_i}{b_k} a_{im} a_{ml} a_{lk} c_i d_k = \frac{19}{720}$
	$\sum \frac{b_i}{b_k} a_{lm} a_{il} a_{ik} d_k c_m = \frac{7}{360}$		$\sum \frac{b_i b_j}{b_l b_m} a_{jm} a_{im} a_{il} c_j d_l = \frac{61}{720}$
	$\sum \frac{b_i b_j}{b_k} a_{jl} a_{jk} a_{ik} c_i c_l = \frac{7}{144}$		$\sum \frac{b_i b_j}{b_k} a_{jl} a_{lk} a_{ik} c_i c_j = \frac{1}{18}$

Order 6

Graphe	Conditions	Graphe	Conditions
	$\sum b_i a_{kl} a_{jk} a_{ij} c_i c_l = \frac{1}{144}$		$\sum b_i a_{il} a_{ik} a_{kj} c_j c_l = \frac{1}{72}$
	$\sum \frac{b_i}{b_k} a_{ik} a_{il} d_k c_l^2 = \frac{7}{180}$		$\sum b_i a_{ik} a_{ij} c_j^2 c_k = \frac{1}{36}$
	$\sum a_{lk} a_{kj} c_j^2 d_l = \frac{1}{360}$		$\sum b_i a_{ik} a_{kj} c_i c_j^2 = \frac{1}{72}$
	$\sum \frac{b_i}{b_k} a_{lk} a_{il} c_i^2 d_k = \frac{13}{180}$		$\sum b_i a_{jk} a_{ij} c_i^2 c_k = \frac{1}{36}$
	$\sum \frac{b_i}{b_k} a_{lk} a_{ik} c_i^2 d_l = \frac{19}{360}$		$\sum \frac{b_i b_j}{b_k} a_{jk} a_{ik} c_i^2 c_j = \frac{7}{72}$
	$\sum \frac{b_i}{b_l b_m} a_{im} a_{il} c_l d_l d_m = \frac{13}{360}$		$\sum \frac{b_i}{b_k} a_{il} a_{ik} c_k d_k c_l = \frac{1}{45}$
	$\sum \frac{1}{b_k} a_{ml} a_{lk} c_k d_k d_m = \frac{1}{180}$		$\sum \frac{b_i}{b_k} a_{il} a_{lk} c_i c_k d_k = \frac{7}{360}$
	$\sum \frac{1}{b_k} a_{mk} a_{lm} d_k c_l d_l = \frac{7}{360}$		$\sum a_{kl} a_{jk} c_j d_j c_l = \frac{1}{180}$
	$\sum \frac{1}{b_k} a_{mk} a_{lk} c_l d_l d_m = \frac{1}{45}$		$\sum \frac{b_i}{b_k} a_{ik} a_{lk} c_i c_l d_l = \frac{13}{360}$
	$\sum \frac{b_i}{b_m^2 b_n} a_{in} a_{im} d_m^2 d_n = \frac{7}{72}$		$\sum \frac{b_i}{b_l^2} a_{im} a_{il} d_l^2 c_m = \frac{19}{360}$
	$\sum \frac{1}{b_l^2} a_{nm} a_{ml} d_l^2 d_n = \frac{1}{36}$		$\sum \frac{b_i}{b_l^2} a_{im} a_{ml} c_i d_l^2 = \frac{13}{180}$
	$\sum \frac{1}{b_l b_m} a_{nm} a_{ln} d_l^2 d_m = \frac{1}{72}$		$\sum \frac{1}{b_k} a_{lm} a_{kl} d_k^2 c_m = \frac{1}{360}$
	$\sum \frac{1}{b_l b_m} a_{nm} a_{lm} d_l^2 d_n = \frac{1}{36}$		$\sum \frac{b_i}{b_l b_m} a_{im} a_{lm} c_i d_l^2 = \frac{7}{180}$
	$\sum \frac{1}{b_k} a_{lk} a_{lm} d_k d_l c_m = \frac{1}{60}$		$\sum a_{jl} a_{jk} d_j c_k c_l = \frac{1}{120}$
	$\sum \frac{1}{b_k} a_{mk} a_{kl} d_k c_l d_m = \frac{1}{240}$		$\sum \frac{b_i}{b_k} a_{ik} a_{kl} c_i d_k c_l = \frac{1}{80}$
	$\sum \frac{b_i}{b_k} a_{ik} a_{il} c_i d_k c_l = \frac{7}{120}$		$\sum b_i a_{ik} a_{ij} c_i c_j c_k = \frac{1}{24}$
	$\sum a_{lj} a_{jk} c_j c_k d_l = \frac{1}{240}$		$\sum b_i a_{ij} a_{jk} c_i c_j c_k = \frac{1}{48}$
	$\sum \frac{b_i}{b_l b_m} a_{lm} a_{il} c_i d_l d_m = \frac{11}{240}$		$\sum \frac{b_i}{b_l^2} a_{ml} a_{il} c_i d_l d_m = \frac{7}{120}$

Order 6

Graphe	Conditions	Graphe	Conditions
	$\sum \frac{b_i b_j}{b_l^2} a_{jl} a_{il} c_i c_j d_l = \frac{11}{120}$		$\sum \frac{b_i}{b_k} a_{lk} a_{il} c_i d_k c_l = \frac{11}{240}$
	$\sum \frac{b_i}{b_k} a_{lk} a_{ik} c_i c_k d_l = \frac{1}{60}$		$\sum \frac{b_i b_j}{b_k} a_{jk} a_{ik} c_i c_j c_k = \frac{1}{24}$
	$\sum \frac{1}{b_l b_m} a_{nm} a_{nl} d_l d_m d_n = \frac{1}{24}$		$\sum \frac{1}{b_l b_m} a_{nl} a_{lm} d_l d_m d_n = \frac{1}{48}$
	$\sum \frac{b_i}{b_l b_m} a_{im} a_{il} c_i d_l d_m = \frac{11}{120}$		$\sum \frac{1}{b_k} a_{ml} a_{lk} d_k c_l d_m = \frac{1}{80}$
	$\sum \frac{1}{b_l^2} a_{nl} a_{ml} d_l d_m d_n = \frac{1}{24}$		$\sum \frac{1}{b_k} a_{mk} a_{lk} c_k d_l d_m = \frac{1}{120}$
	$\sum \frac{1}{b_m^2 b_n} a_{mn} d_m^3 d_n = \frac{1}{24}$		$\sum \frac{1}{b_l^2} a_{lm} d_l^3 c_m = \frac{1}{120}$
	$\sum \frac{1}{b_m^3} a_{nm} d_m^3 d_n = \frac{1}{12}$		$\sum \frac{b_i}{b_m^3} a_{im} c_i d_m^3 = \frac{7}{60}$
	$\sum \frac{1}{b_l b_m} a_{lm} c_l d_l^2 d_m = \frac{1}{40}$		$\sum \frac{1}{b_k} a_{kl} c_k d_k^2 c_l = \frac{1}{120}$
	$\sum \frac{1}{b_l^2} a_{ml} c_l d_l^2 d_m = \frac{1}{60}$		$\sum \frac{b_i}{b_l^2} a_{il} c_i c_l d_l^2 = \frac{1}{30}$
	$\sum \frac{1}{b_k} a_{lk} d_k c_l^2 d_l = \frac{1}{30}$		$\sum a_{jk} c_j^2 d_j c_k = \frac{1}{60}$
	$\sum \frac{1}{b_k} a_{lk} c_k^2 d_k d_l = \frac{1}{120}$		$\sum \frac{b_i}{b_k} a_{ik} c_i c_k^2 d_k = \frac{1}{40}$
	$\sum \frac{b_i}{b_k} a_{ik} c_i^3 d_k = \frac{7}{60}$		$\sum b_i a_{ij} c_i^3 c_j = \frac{1}{12}$
	$\sum a_{kj} c_j^3 d_k = \frac{1}{120}$		$\sum b_i a_{ij} c_i c_j^3 = \frac{1}{24}$
	$\sum \frac{1}{b_l b_m} a_{lm} d_l^2 c_m d_m = \frac{1}{90}$		$\sum \frac{1}{b_k} a_{kl} d_k^2 c_l^2 = \frac{1}{180}$
	$\sum \frac{1}{b_m^2 b_n} a_{nm} d_m^2 d_n^2 = \frac{1}{18}$		$\sum \frac{1}{b_l^2} a_{ml} d_l^2 c_m d_m = \frac{2}{45}$
	$\sum \frac{b_i}{b_l^2} a_{il} c_i^2 d_l^2 = \frac{19}{180}$		$\sum \frac{1}{b_k} a_{lk} c_k d_k c_l d_l = \frac{1}{72}$
	$\sum a_{jk} c_j d_j c_k^2 = \frac{1}{90}$		$\sum \frac{b_i}{b_k} a_{ik} c_i^2 c_k d_k = \frac{2}{45}$
	$\sum b_i a_{ij} c_i^2 c_j^2 = \frac{1}{18}$		$\sum c_j^4 d_j = \frac{1}{30}$

Order 6






























Graphe	Conditions	Graphe	Conditions
	$\sum b_i c_i^5 = \frac{1}{6}$		$\sum \frac{1}{b_k} c_k^3 d_k^2 = \frac{1}{60}$
	$\sum \frac{1}{b_l^2} c_l^2 d_l^3 = \frac{1}{60}$		$\sum \frac{1}{b_m^3} c_m d_m^4 = \frac{1}{30}$
	$\sum \frac{1}{b_n^4} d_n^5 = \frac{1}{6}$		





































Table 4.8: Order 7

Graphe	Conditions	Graphe	Conditions
	$\sum \frac{b_i b_j}{b_l b_m} a_{jm} a_{jl} a_{il} a_{in} d_m c_n = \frac{31}{1008}$		$\sum \frac{b_i b_j}{b_k} a_{jm} a_{jk} a_{ik} a_{il} c_l c_m = \frac{1}{56}$
	$\sum \frac{b_i}{b_k} a_{nm} a_{mk} a_{ik} a_{il} c_l d_n = \frac{17}{2520}$		$\sum \frac{b_i b_j}{b_k} a_{jm} a_{mk} a_{ik} a_{il} c_j c_l = \frac{11}{560}$
	$\sum \frac{b_i}{b_k} a_{nk} a_{mn} a_{im} a_{il} d_k c_l = \frac{11}{1008}$		$\sum b_i a_{lm} a_{kl} a_{ik} a_{ij} c_j c_m = \frac{1}{336}$
	$\sum \frac{b_i}{b_k} a_{nk} a_{mk} a_{im} a_{il} c_l d_n = \frac{71}{5040}$		$\sum \frac{b_i b_j}{b_k} a_{jk} a_{mk} a_{im} a_{il} c_j c_l = \frac{37}{1680}$
	$\sum \frac{b_i}{b_k} a_{ik} a_{in} a_{nm} a_{ml} d_k c_l = \frac{1}{252}$		$\sum a_{nm} a_{ml} a_{lk} a_{kj} c_j d_n = \frac{1}{5040}$
	$\sum b_i a_{im} a_{ml} a_{lk} a_{kj} c_i c_j = \frac{1}{840}$		$\sum \frac{b_i}{b_k} a_{nk} a_{in} a_{im} a_{ml} d_k c_l = \frac{5}{504}$
	$\sum b_i a_{lm} a_{il} a_{ik} a_{kj} c_j c_m = \frac{1}{252}$		$\sum \frac{b_i}{b_k} a_{nk} a_{ik} a_{im} a_{ml} c_l d_n = \frac{17}{2520}$
	$\sum \frac{b_i b_j}{b_k} a_{jk} a_{ik} a_{im} a_{ml} c_j c_l = \frac{4}{315}$		$\sum \frac{b_i b_j}{b_l b_m} a_{jm} a_{jl} a_{nl} a_{in} c_i d_m = \frac{181}{5040}$
	$\sum \frac{b_i}{b_k} a_{nm} a_{mk} a_{lk} a_{il} c_i d_n = \frac{5}{504}$		$\sum \frac{b_i b_j}{b_k} a_{jm} a_{mk} a_{lk} a_{il} c_i c_j = \frac{11}{420}$
	$\sum \frac{b_i}{b_k} a_{nk} a_{mn} a_{lm} a_{il} c_i d_k = \frac{29}{5040}$		$\sum \frac{b_i}{b_k} a_{nk} a_{mk} a_{lm} a_{il} c_i d_n = \frac{11}{1008}$
	$\sum \frac{b_i b_j}{b_k} a_{jk} a_{mk} a_{lm} a_{il} c_i c_j = \frac{13}{840}$		$\sum \frac{b_i b_j}{b_l b_m} a_{jm} a_{jn} a_{nl} a_{il} c_i d_m = \frac{169}{5040}$
	$\sum \frac{b_i}{b_k} a_{nm} a_{ml} a_{lk} a_{ik} c_i d_n = \frac{1}{252}$		$\sum \frac{b_i b_j}{b_l b_m} a_{nm} a_{jn} a_{jl} a_{il} c_i d_m = \frac{181}{5040}$

Order 7

Graphe	Conditions	Graphe	Conditions
	$\sum \frac{b_i b_j}{b_l b_m} a_{nm} a_{jm} a_{jl} a_{il} c_i d_n = \frac{31}{1008}$		$\sum \frac{b_i b_j b_k}{b_l b_m} a_{km} a_{jm} a_{jl} a_{il} c_i c_k = \frac{17}{315}$
	$\sum \frac{b_i b_j}{b_m b_n b_o} a_{jo} a_{jn} a_{in} a_{im} d_m d_o = \frac{17}{315}$		$\sum \frac{b_i}{b_l b_m} a_{on} a_{nm} a_{im} a_{il} d_l d_o = \frac{4}{315}$
	$\sum \frac{b_i}{b_l b_m} a_{om} a_{no} a_{in} a_{il} d_l d_m = \frac{13}{840}$		$\sum \frac{b_i}{b_l b_m} a_{om} a_{nm} a_{in} a_{il} d_l d_o = \frac{37}{1680}$
	$\sum \frac{1}{b_k} a_{on} a_{nm} a_{ml} a_{lk} d_k d_o = \frac{1}{840}$		$\sum \frac{b_i}{b_l b_m} a_{om} a_{io} a_{in} a_{nl} d_l d_m = \frac{11}{420}$
	$\sum \frac{b_i}{b_l b_m} a_{om} a_{im} a_{in} a_{nl} d_l d_o = \frac{11}{560}$		$\sum \frac{1}{b_k} a_{on} a_{nk} a_{mk} a_{lm} d_l d_o = \frac{1}{252}$
	$\sum \frac{1}{b_k} a_{ok} a_{nk} a_{mn} a_{lm} d_l d_o = \frac{1}{336}$		$\sum \frac{b_i}{b_l b_m} a_{om} a_{im} a_{il} a_{nl} d_n d_o = \frac{1}{56}$
	$\sum \frac{b_i}{b_m b_n b_o} a_{io} a_{in} a_{mn} d_m^2 d_o = \frac{8}{315}$		$\sum \frac{b_i}{b_l b_m} a_{in} a_{im} a_{lm} d_l^2 c_n = \frac{17}{1260}$
	$\sum \frac{1}{b_l b_m} a_{on} a_{nm} a_{lm} d_l^2 d_o = \frac{1}{126}$		$\sum \frac{b_i}{b_l b_m} a_{in} a_{nm} a_{lm} c_i d_l^2 = \frac{5}{252}$
	$\sum \frac{1}{b_l b_m} a_{om} a_{no} a_{ln} d_l^2 d_m = \frac{1}{420}$		$\sum \frac{1}{b_k} a_{mn} a_{lm} a_{kl} d_k^2 c_n = \frac{1}{2520}$
	$\sum \frac{1}{b_l b_m} a_{om} a_{nm} a_{ln} d_l^2 d_o = \frac{1}{168}$		$\sum \frac{b_i}{b_l b_m} a_{im} a_{nm} a_{ln} c_i d_l^2 = \frac{1}{126}$
	$\sum \frac{b_i}{b_m^2 b_n} a_{in} a_{io} a_{om} d_m^2 d_n = \frac{37}{840}$		$\sum \frac{b_i}{b_l^2} a_{in} a_{im} a_{ml} d_l^2 c_n = \frac{71}{2520}$
	$\sum \frac{1}{b_l^2} a_{on} a_{nm} a_{ml} d_l^2 d_o = \frac{1}{168}$		$\sum \frac{b_i}{b_l^2} a_{in} a_{nm} a_{ml} c_i d_l^2 = \frac{11}{504}$
	$\sum \frac{b_i}{b_m^2 b_n} a_{on} a_{io} a_{im} d_m^2 d_n = \frac{11}{280}$		$\sum \frac{b_i}{b_l^2} a_{mn} a_{im} a_{il} d_l^2 c_n = \frac{17}{1260}$
	$\sum \frac{b_i}{b_m^2 b_n} a_{on} a_{in} a_{im} d_m^2 d_o = \frac{1}{28}$		$\sum \frac{b_i b_j}{b_m^2 b_n} a_{jn} a_{in} a_{im} c_j d_m^2 = \frac{31}{504}$
	$\sum \frac{b_i}{b_l b_m} a_{im} a_{il} a_{nl} d_m c_n d_n = \frac{13}{560}$		$\sum \frac{b_i}{b_k} a_{im} a_{ik} a_{lk} c_l d_l c_m = \frac{13}{1008}$
	$\sum \frac{1}{b_k} a_{nm} a_{mk} a_{lk} c_l d_l d_n = \frac{1}{168}$		$\sum \frac{b_i}{b_k} a_{im} a_{mk} a_{lk} c_i c_l d_l = \frac{41}{2520}$
	$\sum \frac{1}{b_k} a_{nk} a_{mn} a_{lm} d_k c_l d_l = \frac{23}{5040}$		$\sum a_{lm} a_{kl} a_{jk} c_j d_j c_m = \frac{1}{1008}$
	$\sum \frac{1}{b_k} a_{nk} a_{mk} a_{lm} c_l d_l d_n = \frac{1}{126}$		$\sum \frac{b_i}{b_k} a_{ik} a_{mk} a_{lm} c_i c_l d_l = \frac{29}{2520}$





































Order 7

Graphe	Conditions	Graphe	Conditions
	$\sum \frac{b_i}{b_l b_m} a_{im} a_{in} a_{nl} c_l d_l d_m = \frac{29}{2520}$		$\sum \frac{b_i}{b_k} a_{im} a_{il} a_{lk} c_k d_k c_m = \frac{1}{126}$
	$\sum \frac{1}{b_k} a_{nm} a_{ml} a_{lk} c_k d_k d_n = \frac{1}{1008}$		$\sum \frac{b_i}{b_k} a_{im} a_{ml} a_{lk} c_i c_k d_k = \frac{23}{5040}$
	$\sum \frac{b_i}{b_l b_m} a_{nm} a_{in} a_{il} c_l d_l d_m = \frac{41}{2520}$		$\sum \frac{b_i}{b_k} a_{lm} a_{il} a_{ik} c_k d_k c_m = \frac{1}{168}$
	$\sum \frac{b_i}{b_l b_m} a_{nm} a_{im} a_{il} c_l d_l d_n = \frac{13}{1008}$		$\sum \frac{b_i b_j}{b_l b_m} a_{jm} a_{im} a_{il} c_j c_l d_l = \frac{13}{560}$
	$\sum \frac{b_i b_j}{b_l b_m} a_{jm} a_{jl} a_{il} c_i^2 d_m = \frac{31}{504}$		$\sum \frac{b_i b_j}{b_k} a_{jl} a_{jk} a_{ik} c_i^2 c_l = \frac{1}{28}$
	$\sum \frac{b_i}{b_k} a_{ml} a_{lk} a_{ik} c_i^2 d_m = \frac{17}{1260}$		$\sum \frac{b_i b_j}{b_k} a_{jl} a_{lk} a_{ik} c_i^2 c_j = \frac{11}{280}$
	$\sum \frac{b_i}{b_k} a_{mk} a_{lm} a_{il} c_i^2 d_k = \frac{11}{504}$		$\sum b_i a_{kl} a_{jk} a_{ij} c_i^2 c_l = \frac{1}{168}$
	$\sum \frac{b_i}{b_k} a_{mk} a_{lk} a_{il} c_i^2 d_m = \frac{71}{2520}$		$\sum \frac{b_i b_j}{b_k} a_{jk} a_{lk} a_{il} c_i^2 c_j = \frac{37}{840}$
	$\sum \frac{b_i}{b_k} a_{ik} a_{im} a_{ml} d_k c_l^2 = \frac{1}{126}$		$\sum b_i a_{il} a_{ik} a_{kj} c_j^2 c_l = \frac{1}{168}$
	$\sum a_{ml} a_{lk} a_{kj} c_j^2 d_m = \frac{1}{2520}$		$\sum b_i a_{il} a_{lk} a_{kj} c_i c_j^2 = \frac{1}{420}$
	$\sum \frac{b_i}{b_k} a_{mk} a_{im} a_{il} d_k c_l^2 = \frac{5}{252}$		$\sum b_i a_{kl} a_{ik} a_{ij} c_j^2 c_l = \frac{1}{126}$
	$\sum \frac{b_i}{b_k} a_{mk} a_{ik} a_{il} c_l^2 d_m = \frac{17}{1260}$		$\sum \frac{b_i b_j}{b_k} a_{jk} a_{ik} a_{il} c_j c_l^2 = \frac{8}{315}$
	$\sum \frac{b_i}{b_l b_m} a_{im} a_{il} a_{nl} c_l d_m d_n = \frac{53}{5040}$		$\sum \frac{b_i}{b_k} a_{im} a_{ik} a_{lk} c_k d_l c_m = \frac{31}{5040}$
	$\sum \frac{1}{b_k} a_{nm} a_{mk} a_{lk} c_k d_l d_n = \frac{1}{504}$		$\sum \frac{b_i}{b_k} a_{im} a_{mk} a_{lk} c_i c_k d_l = \frac{2}{315}$
	$\sum \frac{1}{b_k} a_{nk} a_{ln} a_{ml} d_k c_l d_m = \frac{17}{5040}$		$\sum a_{lm} a_{jl} a_{kj} c_j d_k c_m = \frac{1}{1260}$
	$\sum \frac{1}{b_k} a_{nk} a_{lk} a_{ml} c_l d_m d_n = \frac{5}{1008}$		$\sum \frac{b_i}{b_k} a_{ik} a_{lk} a_{ml} c_i c_l d_m = \frac{19}{2520}$
	$\sum \frac{b_i}{b_m^2 b_n} a_{in} a_{im} a_{om} d_m d_n d_o = \frac{4}{105}$		$\sum \frac{b_i}{b_l^2} a_{in} a_{il} a_{ml} d_l d_m c_n = \frac{17}{840}$
	$\sum \frac{1}{b_l^2} a_{on} a_{nl} a_{ml} d_l d_m d_o = \frac{1}{84}$		$\sum \frac{b_i}{b_l^2} a_{in} a_{nl} a_{ml} c_i d_l d_m = \frac{5}{168}$





































Order 7

Graphe	Conditions	Graphe	Conditions
	$\sum \frac{1}{b_l b_m} a_{om} a_{lo} a_{nl} d_l d_m d_n = \frac{1}{280}$		$\sum \frac{1}{b_k} a_{mn} a_{km} a_{lk} d_k d_l c_n = \frac{1}{1680}$
	$\sum \frac{1}{b_l b_m} a_{om} a_{lm} a_{nl} d_l d_n d_o = \frac{1}{112}$		$\sum \frac{b_i}{b_l b_m} a_{im} a_{lm} a_{nl} c_i d_l d_n = \frac{1}{84}$
	$\sum \frac{b_i}{b_l b_m} a_{im} a_{in} a_{nl} d_l d_m c_n = \frac{17}{630}$		$\sum \frac{b_i}{b_k} a_{im} a_{il} a_{lk} d_k c_l c_m = \frac{19}{1008}$
	$\sum \frac{1}{b_k} a_{nm} a_{ml} a_{lk} d_k c_l d_n = \frac{11}{5040}$		$\sum \frac{b_i}{b_k} a_{im} a_{ml} a_{lk} c_i d_k c_l = \frac{13}{1260}$
	$\sum \frac{b_i}{b_l b_m} a_{nm} a_{in} a_{il} c_i d_l d_m = \frac{107}{2520}$		$\sum \frac{b_i}{b_k} a_{lm} a_{il} a_{ik} c_i d_k c_m = \frac{1}{63}$
	$\sum \frac{b_i}{b_l b_m} a_{nm} a_{im} a_{il} c_i d_l d_n = \frac{41}{1260}$		$\sum \frac{b_i b_j}{b_l b_m} a_{jm} a_{im} a_{il} c_i c_j d_l = \frac{149}{2520}$
	$\sum \frac{b_i}{b_m b_n b_o} a_{io} a_{im} a_{mn} d_m d_n d_o = \frac{1}{35}$		$\sum \frac{b_i}{b_l b_m} a_{in} a_{il} a_{lm} d_l d_m c_n = \frac{29}{1680}$
	$\sum \frac{1}{b_l b_m} a_{on} a_{nl} a_{lm} d_l d_m d_o = \frac{1}{210}$		$\sum \frac{b_i}{b_l b_m} a_{in} a_{nl} a_{lm} c_i d_l d_m = \frac{9}{560}$
	$\sum \frac{1}{b_l b_m} a_{om} a_{no} a_{nl} d_l d_m d_n = \frac{11}{840}$		$\sum \frac{1}{b_k} a_{mn} a_{lm} a_{lk} d_k d_l c_n = \frac{1}{280}$
	$\sum \frac{1}{b_l b_m} a_{om} a_{nm} a_{nl} d_l d_n d_o = \frac{9}{560}$		$\sum \frac{b_i}{b_l b_m} a_{im} a_{nm} a_{nl} c_i d_l d_n = \frac{43}{1680}$
	$\sum \frac{b_i b_j}{b_l b_m} a_{jm} a_{jl} a_{il} c_i c_l d_m = \frac{43}{1680}$		$\sum \frac{b_i b_j}{b_k} a_{jl} a_{jk} a_{ik} c_i c_k c_l = \frac{9}{560}$
	$\sum \frac{b_i}{b_k} a_{ml} a_{lk} a_{ik} c_i c_k d_m = \frac{1}{280}$		$\sum \frac{b_i b_j}{b_k} a_{jl} a_{lk} a_{ik} c_i c_j c_k = \frac{11}{840}$
	$\sum \frac{b_i}{b_k} a_{mk} a_{lm} a_{il} c_i d_k c_l = \frac{9}{560}$		$\sum b_i a_{kl} a_{jk} a_{ij} c_i c_j c_l = \frac{1}{210}$
	$\sum \frac{b_i}{b_k} a_{mk} a_{lk} a_{il} c_i c_l d_m = \frac{29}{1680}$		$\sum \frac{b_i b_j}{b_k} a_{jk} a_{lk} a_{il} c_i c_j c_l = \frac{1}{35}$
	$\sum \frac{b_i b_j}{b_m^2 b_n} a_{jn} a_{jm} a_{im} c_i d_m d_n = \frac{149}{2520}$		$\sum \frac{b_i b_j}{b_l^2} a_{jm} a_{jl} a_{il} c_i d_l c_m = \frac{41}{1260}$
	$\sum \frac{b_i}{b_l^2} a_{nm} a_{ml} a_{il} c_i d_l d_n = \frac{1}{63}$		$\sum \frac{b_i b_j}{b_l^2} a_{jm} a_{ml} a_{il} c_i c_j d_l = \frac{107}{2520}$
	$\sum \frac{b_i}{b_l b_m} a_{nm} a_{ln} a_{il} c_i d_l d_m = \frac{13}{1260}$		$\sum \frac{b_i}{b_k} a_{lm} a_{kl} a_{ik} c_i d_k c_m = \frac{11}{5040}$
	$\sum \frac{b_i}{b_l b_m} a_{nm} a_{lm} a_{il} c_i d_l d_n = \frac{19}{1008}$		$\sum \frac{b_i b_j}{b_l b_m} a_{jm} a_{lm} a_{il} c_i c_j d_l = \frac{17}{630}$





































Order 7

Graphe	Conditions	Graphe	Conditions
	$\sum \frac{b_i}{b_k} a_{ik} a_{il} a_{lm} d_k c_l c_m = \frac{1}{84}$		$\sum b_i a_{il} a_{ij} a_{jk} c_j c_k c_l = \frac{1}{112}$
	$\sum a_{ml} a_{lj} a_{jk} c_j c_k d_m = \frac{1}{1680}$		$\sum b_i a_{il} a_{lj} a_{jk} c_i c_j c_k = \frac{1}{280}$
	$\sum \frac{b_i}{b_k} a_{mk} a_{im} a_{il} d_k c_l = \frac{5}{168}$		$\sum b_i a_{kl} a_{ik} a_{ij} c_i c_j c_l = \frac{1}{84}$
	$\sum \frac{b_i}{b_k} a_{mk} a_{ik} a_{il} c_i c_l d_m = \frac{17}{840}$		$\sum \frac{b_i b_j}{b_k} a_{jk} a_{ik} a_{il} c_i c_j c_l = \frac{4}{105}$
	$\sum \frac{b_i}{b_l b_m} a_{im} a_{il} a_{ln} d_l d_m c_n = \frac{19}{2520}$		$\sum \frac{b_i}{b_k} a_{im} a_{ik} a_{kl} d_k c_l c_m = \frac{5}{1008}$
	$\sum \frac{1}{b_k} a_{nm} a_{mk} a_{kl} d_k c_l d_n = \frac{1}{1260}$		$\sum \frac{b_i}{b_k} a_{im} a_{mk} a_{kl} c_i d_k c_l = \frac{17}{5040}$
	$\sum \frac{1}{b_k} a_{nk} a_{ln} a_{lm} d_k d_l c_m = \frac{2}{315}$		$\sum a_{lm} a_{jl} a_{jk} d_j c_k c_m = \frac{1}{504}$
	$\sum \frac{1}{b_k} a_{nk} a_{lk} a_{lm} d_l c_m d_n = \frac{31}{5040}$		$\sum \frac{b_i}{b_k} a_{ik} a_{lk} a_{lm} c_i d_l c_m = \frac{53}{5040}$
	$\sum \frac{b_i}{b_k} a_{ik} a_{il} d_k c_l^3 = \frac{1}{42}$		$\sum b_i a_{ik} a_{ij} c_j^3 c_k = \frac{1}{56}$
	$\sum a_{lk} a_{kj} c_j^3 d_l = \frac{1}{840}$		$\sum b_i a_{ik} a_{kj} c_i c_j^3 = \frac{1}{140}$
	$\sum \frac{b_i}{b_k} a_{lk} a_{il} c_i^3 d_k = \frac{5}{84}$		$\sum b_i a_{jk} a_{ij} c_i^3 c_k = \frac{1}{42}$
	$\sum \frac{b_i}{b_k} a_{lk} a_{ik} c_i^3 d_l = \frac{17}{420}$		$\sum \frac{b_i b_j}{b_k} a_{jk} a_{ik} c_i^3 c_j = \frac{8}{105}$
	$\sum \frac{b_i}{b_l b_m} a_{im} a_{il} c_l^2 d_l d_m = \frac{19}{1260}$		$\sum \frac{b_i}{b_k} a_{il} a_{ik} c_k^2 d_k c_l = \frac{5}{504}$
	$\sum \frac{1}{b_k} a_{ml} a_{lk} c_k^2 d_k d_m = \frac{1}{630}$		$\sum \frac{b_i}{b_k} a_{il} a_{lk} c_i c_k^2 d_k = \frac{17}{2520}$
	$\sum \frac{1}{b_k} a_{mk} a_{lm} d_k c_l^2 d_l = \frac{4}{315}$		$\sum a_{kl} a_{jk} c_j^2 d_j c_l = \frac{1}{252}$
	$\sum \frac{1}{b_k} a_{mk} a_{lk} c_l^2 d_l d_m = \frac{31}{2520}$		$\sum \frac{b_i}{b_k} a_{ik} a_{lk} c_i c_l^2 d_l = \frac{53}{2520}$
	$\sum \frac{b_i}{b_m^2 b_n} a_{in} a_{im} c_m d_m^2 d_n = \frac{53}{2520}$		$\sum \frac{b_i}{b_l^2} a_{im} a_{il} c_l d_l^2 c_m = \frac{31}{2520}$
	$\sum \frac{1}{b_l^2} a_{nm} a_{ml} c_l d_l^2 d_n = \frac{1}{252}$		$\sum \frac{b_i}{b_l^2} a_{im} a_{ml} c_i c_l d_l^2 = \frac{4}{315}$

Order 7

Graphe	Conditions	Graphe	Conditions
	$\sum \frac{1}{b_l b_m} a_{nm} a_{ln} c_l d_l^2 d_m = \frac{17}{2520}$		$\sum \frac{1}{b_k} a_{lm} a_{kl} c_k d_k^2 c_m = \frac{1}{630}$
	$\sum \frac{1}{b_l b_m} a_{nm} a_{lm} c_l d_l^2 d_n = \frac{5}{504}$		$\sum \frac{b_i}{b_l b_m} a_{im} a_{lm} c_l d_l^2 = \frac{19}{1260}$
	$\sum \frac{b_i}{b_n^3 b_o} a_{io} a_{in} d_n^3 d_o = \frac{8}{105}$		$\sum \frac{b_i}{b_m^3} a_{in} a_{im} d_m^3 c_n = \frac{17}{420}$
	$\sum \frac{1}{b_m^3} a_{on} a_{nm} d_m^3 d_o = \frac{1}{42}$		$\sum \frac{b_i}{b_m^3} a_{in} a_{nm} c_i d_m^3 = \frac{5}{84}$
	$\sum \frac{1}{b_m^2 b_n} a_{on} a_{mo} d_m^3 d_n = \frac{1}{140}$		$\sum \frac{1}{b_l^2} a_{mn} a_{lm} d_l^3 c_n = \frac{1}{840}$
	$\sum \frac{1}{b_m^2 b_n} a_{on} a_{mn} d_m^3 d_o = \frac{1}{56}$		$\sum \frac{b_i}{b_m^2 b_n} a_{in} a_{mn} c_i d_m^3 = \frac{1}{42}$
	$\sum \frac{1}{b_k} a_{mk} a_{ml} d_k c_l^2 d_m = \frac{1}{140}$		$\sum a_{kl} a_{kj} c_j^2 d_k c_l = \frac{1}{252}$
	$\sum \frac{1}{b_k} a_{mk} a_{kl} d_k c_l^2 d_m = \frac{1}{840}$		$\sum \frac{b_i}{b_k} a_{ik} a_{kl} c_i d_k c_l^2 = \frac{11}{2520}$
	$\sum \frac{b_i}{b_k} a_{ik} a_{il} c_i d_k c_l^2 = \frac{2}{63}$		$\sum b_i a_{ik} a_{ij} c_i c_j^2 c_k = \frac{1}{42}$
	$\sum a_{lk} a_{kj} c_j^2 c_k d_l = \frac{1}{630}$		$\sum b_i a_{ik} a_{kj} c_i c_j^2 c_k = \frac{1}{105}$
	$\sum \frac{b_i}{b_l b_m} a_{lm} a_{il} c_i^2 d_l d_m = \frac{29}{840}$		$\sum \frac{b_i}{b_k} a_{kl} a_{ik} c_i^2 d_k c_l = \frac{5}{504}$
	$\sum \frac{b_i}{b_l^2} a_{ml} a_{il} c_i^2 d_l d_m = \frac{17}{420}$		$\sum \frac{b_i b_j}{b_l^2} a_{jl} a_{il} c_i^2 c_j d_l = \frac{41}{630}$
	$\sum \frac{b_i}{b_k} a_{lk} a_{il} c_i^2 d_k c_l = \frac{19}{504}$		$\sum b_i a_{jk} a_{ij} c_i^2 c_j c_k = \frac{1}{56}$
	$\sum \frac{b_i}{b_k} a_{lk} a_{ik} c_i^2 c_k d_l = \frac{31}{2520}$		$\sum \frac{b_i b_j}{b_k} a_{jk} a_{ik} c_i^2 c_j c_k = \frac{9}{280}$
	$\sum \frac{1}{b_l b_m} a_{nm} a_{nl} c_l d_l d_m d_n = \frac{1}{105}$		$\sum \frac{1}{b_k} a_{lm} a_{lk} c_k d_l c_m = \frac{11}{2520}$
	$\sum \frac{1}{b_l b_m} a_{nm} a_{ml} c_l d_l d_m d_n = \frac{1}{336}$		$\sum \frac{b_i}{b_l b_m} a_{im} a_{ml} c_l d_l d_m = \frac{41}{5040}$
	$\sum \frac{b_i}{b_l b_m} a_{im} a_{il} c_i c_l d_l d_m = \frac{67}{2520}$		$\sum \frac{b_i}{b_k} a_{il} a_{ik} c_i c_k d_k c_l = \frac{1}{56}$
	$\sum \frac{1}{b_k} a_{ml} a_{lk} c_k d_k c_l d_m = \frac{13}{5040}$		$\sum \frac{b_i}{b_k} a_{il} a_{lk} c_i c_k d_k c_l = \frac{19}{1680}$





































Order 7

Graphe	Conditions	Graphe	Conditions
	$\sum \frac{1}{b_l b_m} a_{lm} a_{nl} d_l d_m c_n d_n = \frac{19}{1680}$		$\sum \frac{1}{b_k} a_{km} a_{lk} d_k c_l d_l c_m = \frac{13}{5040}$
	$\sum \frac{1}{b_l^2} a_{nl} a_{ml} d_l c_m d_m d_n = \frac{1}{56}$		$\sum \frac{b_i}{b_l^2} a_{il} a_{ml} c_i d_l c_m d_m = \frac{67}{2520}$
	$\sum \frac{1}{b_k} a_{mk} a_{lm} d_k c_l d_l c_m = \frac{41}{5040}$		$\sum a_{kl} a_{jk} c_j d_j c_k c_l = \frac{1}{336}$
	$\sum \frac{1}{b_k} a_{mk} a_{lk} c_k c_l d_l d_m = \frac{11}{2520}$		$\sum \frac{b_i}{b_k} a_{ik} a_{lk} c_i c_k c_l d_l = \frac{1}{105}$
	$\sum \frac{1}{b_m^2 b_n} a_{on} a_{om} d_m^2 d_n d_o = \frac{9}{280}$		$\sum \frac{1}{b_l^2} a_{mn} a_{ml} d_l^2 d_m c_n = \frac{31}{2520}$
	$\sum \frac{1}{b_m^2 b_n} a_{on} a_{nm} d_m^2 d_n d_o = \frac{1}{56}$		$\sum \frac{b_i}{b_m^2 b_n} a_{in} a_{nm} c_i d_m^2 d_n = \frac{19}{504}$
	$\sum \frac{b_i}{b_m^2 b_n} a_{in} a_{im} c_i d_m^2 d_n = \frac{41}{630}$		$\sum \frac{b_i}{b_l^2} a_{im} a_{il} c_i d_l^2 c_m = \frac{17}{420}$
	$\sum \frac{1}{b_l^2} a_{nm} a_{ml} d_l^2 c_m d_n = \frac{5}{504}$		$\sum \frac{b_i}{b_l^2} a_{im} a_{ml} c_i d_l^2 c_m = \frac{29}{840}$
	$\sum \frac{1}{b_m b_n b_o} a_{no} a_{mn} d_m^2 d_n d_o = \frac{1}{105}$		$\sum \frac{1}{b_l b_m} a_{mn} a_{lm} d_l^2 d_m c_n = \frac{1}{630}$
	$\sum \frac{1}{b_m^2 b_n} a_{om} a_{nm} d_m d_n^2 d_o = \frac{1}{42}$		$\sum \frac{b_i}{b_m^2 b_n} a_{im} a_{nm} c_i d_m d_n^2 = \frac{2}{63}$
	$\sum \frac{1}{b_l b_m} a_{nm} a_{ln} d_l^2 d_m c_n = \frac{11}{2520}$		$\sum \frac{1}{b_k} a_{lm} a_{kl} d_k^2 c_l c_m = \frac{1}{840}$
	$\sum \frac{1}{b_l b_m} a_{nm} a_{lm} d_l^2 c_m d_n = \frac{1}{252}$		$\sum \frac{b_i}{b_l b_m} a_{im} a_{lm} c_i d_l^2 c_m = \frac{1}{140}$
	$\sum \frac{b_i}{b_k} a_{ik} a_{il} c_k d_k c_l^2 = \frac{1}{84}$		$\sum b_i a_{ik} a_{ij} c_j^2 c_k^2 = \frac{1}{63}$
	$\sum \frac{b_i}{b_l^2} a_{il} a_{im} d_l^2 c_m^2 = \frac{17}{630}$		$\sum a_{kl} a_{lj} c_j^2 c_k d_k = \frac{1}{504}$
	$\sum b_i a_{ik} a_{kj} c_i^2 c_j^2 = \frac{1}{84}$		$\sum \frac{1}{b_k} a_{km} a_{ml} d_k^2 c_l^2 = \frac{1}{1260}$
	$\sum \frac{b_i}{b_k} a_{lk} a_{il} c_i^2 c_k d_k = \frac{1}{63}$		$\sum \frac{b_i}{b_l^2} a_{ml} a_{im} c_i^2 d_l^2 = \frac{71}{1260}$
	$\sum \frac{b_i}{b_k} a_{lk} a_{ik} c_i^2 c_l d_l = \frac{13}{504}$		$\sum \frac{b_i b_j}{b_k} a_{jk} a_{ik} c_i^2 c_j^2 = \frac{1}{14}$
	$\sum \frac{b_i}{b_l b_m} a_{lm} a_{im} c_i^2 d_l^2 = \frac{17}{630}$		$\sum \frac{b_i}{b_l b_m} a_{im} a_{il} c_l d_l c_m d_m = \frac{13}{1260}$





































Order 7

Graphe	Conditions	Graphe	Conditions
	$\sum \frac{b_i}{b_m^2 b_n} a_{im} a_{in} d_m^2 c_n d_n = \frac{13}{504}$		$\sum \frac{1}{b_k} a_{lm} a_{mk} c_k d_k c_l d_l = \frac{1}{280}$
	$\sum \frac{1}{b_l b_m} a_{mn} a_{nl} c_l d_l d_m^2 = \frac{1}{504}$		$\sum \frac{1}{b_l^2} a_{nl} a_{mn} d_l^2 c_m d_m = \frac{1}{63}$
	$\sum \frac{1}{b_k} a_{mk} a_{lk} c_l d_l c_m d_m = \frac{13}{1260}$		$\sum \frac{1}{b_l b_m} a_{lm} a_{nm} d_l^2 c_n d_n = \frac{1}{84}$
	$\sum \frac{b_i}{b_n^2 b_o} a_{io} a_{in} d_n^2 d_o^2 = \frac{1}{14}$		$\sum \frac{1}{b_m^2 b_n} a_{no} a_{om} d_m^2 d_n^2 = \frac{1}{84}$
	$\sum \frac{1}{b_m b_n b_o} a_{no} a_{mo} d_m^2 d_n^2 = \frac{1}{63}$		$\sum \frac{b_i}{b_k} a_{ik} a_{im} a_{il} d_k c_l c_m = \frac{1}{42}$
	$\sum b_i a_{il} a_{ik} a_{ij} c_j c_k c_l = \frac{1}{56}$		$\sum a_{mj} a_{jl} a_{jk} c_k c_l d_m = \frac{1}{840}$
	$\sum b_i a_{ij} a_{jl} a_{jk} c_i c_k c_l = \frac{1}{140}$		$\sum \frac{b_i}{b_k} a_{lk} a_{lm} a_{il} d_k c_m = \frac{23}{1680}$
	$\sum \frac{b_i}{b_k} a_{mk} a_{kl} a_{ik} c_i c_l d_m = \frac{1}{336}$		$\sum \frac{b_i b_j}{b_k} a_{jk} a_{kl} a_{ik} c_i c_j c_l = \frac{1}{105}$
	$\sum \frac{b_i}{b_l b_m} a_{im} a_{in} a_{il} d_l d_m c_n = \frac{29}{840}$		$\sum \frac{1}{b_k} a_{nl} a_{lm} a_{lk} d_k c_m d_n = \frac{1}{336}$
	$\sum \frac{1}{b_k} a_{nk} a_{km} a_{lk} d_l c_m d_n = \frac{1}{840}$		$\sum \frac{b_i b_j}{b_l b_m} a_{lm} a_{jl} a_{il} c_i c_j d_m = \frac{9}{280}$
	$\sum \frac{b_i b_j}{b_l^2} a_{ml} a_{jl} a_{il} c_i c_j d_m = \frac{29}{840}$		$\sum \frac{b_i b_j b_k}{b_l^2} a_{kl} a_{jl} a_{il} c_i c_j c_k = \frac{2}{35}$
	$\sum \frac{b_i}{b_l b_m} a_{nm} a_{in} a_{nl} c_i d_l d_m = \frac{9}{280}$		$\sum \frac{b_i}{b_l b_m} a_{nl} a_{il} a_{lm} c_i d_m d_n = \frac{23}{1680}$
	$\sum \frac{b_i}{b_l^2} a_{nl} a_{il} a_{ml} c_i d_m d_n = \frac{1}{42}$		$\sum \frac{b_i}{b_m b_n b_o} a_{io} a_{in} a_{im} d_m d_n d_o = \frac{2}{35}$
	$\sum \frac{1}{b_l b_m} a_{on} a_{nm} a_{nl} d_l d_m d_o = \frac{1}{105}$		$\sum \frac{1}{b_l b_m} a_{ol} a_{lm} a_{nl} d_m d_n d_o = \frac{1}{140}$
	$\sum \frac{1}{b_l^2} a_{ol} a_{nl} a_{ml} d_m d_n d_o = \frac{1}{56}$		$\sum \frac{1}{b_l b_m} a_{lm} a_{ln} d_l^2 d_m c_n = \frac{1}{168}$
	$\sum \frac{1}{b_k} a_{km} a_{kl} d_k^2 c_l c_m = \frac{1}{420}$		$\sum \frac{1}{b_l^2} a_{nl} a_{lm} d_l^2 c_m d_n = \frac{1}{420}$
	$\sum \frac{b_i}{b_l^2} a_{il} a_{lm} c_i d_l^2 c_m = \frac{1}{168}$		$\sum \frac{1}{b_k} a_{lk} a_{lm} d_k c_l d_l c_m = \frac{3}{280}$
	$\sum a_{jl} a_{jk} c_j d_j c_k c_l = \frac{1}{168}$		$\sum \frac{1}{b_k} a_{mk} a_{kl} c_k d_k c_l d_m = \frac{1}{560}$



Order 7

Graphe	Conditions	Graphe	Conditions
	$\sum \frac{b_i}{b_k} a_{ik} a_{kl} c_i c_k d_k c_l = \frac{11}{1680}$		$\sum \frac{b_i}{b_k} a_{ik} a_{il} c_i^2 d_k c_l = \frac{1}{21}$
	$\sum b_i a_{ik} a_{ij} c_i^2 c_j c_k = \frac{1}{28}$		$\sum a_{lj} a_{jk} c_j^2 c_k d_l = \frac{1}{420}$
	$\sum b_i a_{ij} a_{jk} c_i c_j^2 c_k = \frac{1}{70}$		$\sum \frac{b_i}{b_m^2 b_n} a_{mn} a_{im} c_i d_m^2 d_n = \frac{23}{840}$
	$\sum \frac{b_i}{b_m^3} a_{nm} a_{im} c_i d_m^2 d_n = \frac{1}{21}$		$\sum \frac{b_i b_j}{b_m^3} a_{jm} a_{im} c_i c_j d_m^2 = \frac{29}{420}$
	$\sum \frac{b_i}{b_l b_m} a_{lm} a_{il} c_i c_l d_l d_m = \frac{31}{1680}$		$\sum \frac{b_i}{b_l^2} a_{ml} a_{il} c_i c_l d_l d_m = \frac{3}{280}$
	$\sum \frac{b_i b_j}{b_l^2} a_{jl} a_{il} c_i c_j c_l d_l = \frac{19}{840}$		$\sum \frac{b_i}{b_k} a_{lk} a_{il} c_i d_k c_l^2 = \frac{23}{840}$
	$\sum \frac{b_i}{b_k} a_{lk} a_{ik} c_i c_k^2 d_l = \frac{1}{168}$		$\sum \frac{b_i b_j}{b_k} a_{jk} a_{ik} c_i c_j c_k^2 = \frac{2}{105}$
	$\sum \frac{1}{b_m b_n b_o} a_{mo} a_{mn} d_m^2 d_n d_o = \frac{2}{105}$		$\sum \frac{1}{b_m^2 b_n} a_{om} a_{mn} d_m^2 d_n d_o = \frac{1}{70}$
	$\sum \frac{1}{b_l b_m} a_{nm} a_{nl} d_l d_m c_n d_n = \frac{19}{840}$		$\sum \frac{1}{b_l b_m} a_{nl} a_{lm} c_l d_l d_m d_n = \frac{11}{1680}$
	$\sum \frac{b_i}{b_l b_m} a_{im} a_{il} c_i^2 d_l d_m = \frac{29}{420}$		$\sum \frac{1}{b_k} a_{ml} a_{lk} d_k c_l^2 d_m = \frac{1}{168}$
	$\sum \frac{1}{b_m^3} a_{om} a_{nm} d_m^2 d_n d_o = \frac{1}{28}$		$\sum \frac{1}{b_l^2} a_{nl} a_{ml} c_l d_l d_m d_n = \frac{1}{168}$
	$\sum \frac{1}{b_k} a_{mk} a_{lk} c_k^2 d_l d_m = \frac{1}{420}$		$\sum \frac{1}{b_n^3 b_o} a_{no} d_n^4 d_o = \frac{1}{35}$
	$\sum \frac{1}{b_m^3} a_{mn} d_m^4 c_n = \frac{1}{210}$		$\sum \frac{1}{b_n^4} a_{on} d_n^4 d_o = \frac{1}{14}$
	$\sum \frac{b_i}{b_n^4} a_{in} c_i d_n^4 = \frac{2}{21}$		$\sum \frac{1}{b_m^2 b_n} a_{mn} c_m d_m^3 d_n = \frac{11}{840}$
	$\sum \frac{1}{b_l^2} a_{lm} c_l d_l^3 c_m = \frac{1}{280}$		$\sum \frac{1}{b_m^3} a_{nm} c_m d_m^3 d_n = \frac{1}{84}$
	$\sum \frac{b_i}{b_m^3} a_{im} c_i c_m d_m^3 = \frac{3}{140}$		$\sum \frac{1}{b_l b_m} a_{lm} c_l^2 d_l^2 d_m = \frac{1}{84}$
	$\sum \frac{1}{b_k} a_{kl} c_k^2 d_k^2 c_l = \frac{1}{210}$		$\sum \frac{1}{b_l^2} a_{ml} c_l^2 d_l^2 d_m = \frac{1}{210}$
	$\sum \frac{b_i}{b_l^2} a_{il} c_i c_l^2 d_l^2 = \frac{1}{84}$		$\sum \frac{1}{b_k} a_{lk} d_k c_l^3 d_l = \frac{3}{140}$

Order 7

Graphe	Conditions	Graphe	Conditions
	$\sum a_{jk} c_j^3 d_j c_k = \frac{1}{84}$		$\sum \frac{1}{b_k} a_{lk} c_k^3 d_k d_l = \frac{1}{280}$
	$\sum \frac{b_i}{b_k} a_{ik} c_i^3 d_k = \frac{11}{840}$		$\sum \frac{b_i}{b_k} a_{ik} c_i^4 d_k = \frac{2}{21}$
	$\sum b_i a_{ij} c_i^4 c_j = \frac{1}{14}$		$\sum a_{kj} c_j^4 d_k = \frac{1}{210}$
	$\sum b_i a_{ij} c_i c_j^4 = \frac{1}{35}$		$\sum \frac{1}{b_m^2 b_n} a_{mn} d_m^3 c_n d_n = \frac{1}{168}$
	$\sum \frac{1}{b_l^2} a_{lm} d_l^3 c_m^2 = \frac{1}{420}$		$\sum \frac{1}{b_n^2 b_o^2} a_{no} d_n^3 d_o^2 = \frac{1}{28}$
	$\sum \frac{1}{b_m^3} a_{nm} d_m^3 c_n d_n = \frac{1}{28}$		$\sum \frac{b_i}{b_m^3} a_{im} c_i^2 d_m^3 = \frac{17}{210}$
	$\sum \frac{1}{b_n^3 b_o} a_{on} d_n^3 d_o^2 = \frac{1}{21}$		$\sum \frac{1}{b_l b_m} a_{lm} c_l d_l^2 c_m d_m = \frac{13}{2520}$
	$\sum \frac{1}{b_k} a_{kl} c_k d_k^2 c_l^2 = \frac{1}{315}$		$\sum \frac{1}{b_m^2 b_n} a_{nm} d_m^2 c_n d_n^2 = \frac{5}{252}$
	$\sum \frac{1}{b_l^2} a_{ml} c_l d_l^2 c_m d_m = \frac{11}{1260}$		$\sum \frac{b_i}{b_l^2} a_{il} c_i^2 c_l d_l^2 = \frac{31}{1260}$
	$\sum \frac{1}{b_m^2 b_n} a_{nm} c_m d_m^2 d_n^2 = \frac{1}{126}$		$\sum \frac{1}{b_k} a_{lk} c_k d_k^2 c_l^2 d_l = \frac{11}{1260}$
	$\sum a_{jk} c_j^2 d_j c_k^2 = \frac{1}{126}$		$\sum \frac{1}{b_l^2} a_{ml} d_l^2 c_m^2 d_m = \frac{31}{1260}$
	$\sum \frac{1}{b_k} a_{lk} c_k^2 d_k c_l d_l = \frac{13}{2520}$		$\sum \frac{b_i}{b_k} a_{ik} c_i^2 c_k^2 d_k = \frac{5}{252}$
	$\sum \frac{1}{b_l b_m} a_{ml} c_l^2 d_l d_m^2 = \frac{1}{315}$		$\sum \frac{b_i}{b_k} a_{ik} c_i^3 c_k d_k = \frac{1}{28}$
	$\sum b_i a_{ij} c_i^3 c_j^2 = \frac{1}{21}$		$\sum \frac{b_i}{b_l^2} a_{il} c_i^3 d_l^2 = \frac{17}{210}$
	$\sum a_{kj} c_j^3 c_k d_k = \frac{1}{168}$		$\sum b_i a_{ij} c_i^2 c_j^3 = \frac{1}{28}$
	$\sum \frac{1}{b_k} a_{kl} d_k^2 c_l^3 = \frac{1}{420}$		$\sum c_j^5 d_j = \frac{1}{42}$
	$\sum b_i c_i^6 = \frac{1}{7}$		$\sum \frac{1}{b_k} c_k^4 d_k^2 = \frac{1}{105}$
	$\sum \frac{1}{b_l^3} c_l^3 d_l^3 = \frac{1}{140}$		$\sum \frac{1}{b_m^3} c_m^2 d_m^4 = \frac{1}{105}$

Order 7

Graphe	Conditions	Graphe	Conditions
	$\sum \frac{1}{b_n^4} c_n d_n^5 = \frac{1}{42}$		$\sum \frac{1}{b_o^5} d_o^6 = \frac{1}{7}$

Partie IV

Méthode de point intérieur

Chapitre 5

An Interior-Point Approach to Trajectory Optimization

This chapter presents the interior-point approach for solving optimal control problems. Combining a flexible refinement scheme with dedicated linear algebra solvers, we obtain an efficient algorithm. Numerical results are displayed for various problems, among them several variants of atmospheric reentry.

5.1 Introduction

This part discusses the numerical resolution of optimal control problems of systems governed by ordinary differential equations. We deal with trajectory optimization methods. In other words, the object manipulated by the algorithm is a certain function of time (as opposed to Hamilton-Jacobi-Bellman approaches which solve a certain partial differential equation). Trajectory optimization algorithms start from an initial, in general nonfeasible trajectory, and find a trajectory that approximately satisfies some necessary conditions for optimality. In this chapter we assume all data to be at least C^2 (twice continuously differentiable), which allows to use second-order Taylor expansions. Our format includes running mixed control and state constraints (which means that some constraints involving both the control and state, or only one of them, are active for all time).

Although there is no full agreement on this terminology, one generally distinguishes direct and indirect methods. Direct methods approximately solve a time discretized version of the optimal control problem by some nonlinear programming algorithm, and then possibly refine the discretization mesh and loop. In indirect methods, the control variable is eliminated thanks to Pontryaguin's principle, and the resulting two point boundary value problem

(TPBVP), with only state and costate variables (plus the Lagrange multipliers associated with constraints) is solved without discretizing functions of time; the time integration is performed thanks to a (somewhat hidden) ODE solver. The simplest indirect method is the shooting algorithm, which with the pair (state, costate) at initial time associates its value at the final time. In this way necessary optimality conditions appearing in Pontryaguin's principle are reduced to the final dimensional problem of matching the end point conditions. Since the integration of the reduced (state, costate) differential equation over a long time may be unstable, one often prefers to take as variables the values of state and costate at sampled points, so that integration occurs over small intervals of time. See Stoer and Bulirsch [70].

The common feeling is that indirect methods are more efficient when the intervals of time during which a set of active constraints is active is easy to predict, up to a parametrization of junction times (the endpoints of these intervals). Also, these junction times have to satisfy some regularity conditions, see Maurer [59]. In that case, by using high order integration schemes, one obtains very precise and cheap solvers.

We note that the pseudospectral knotting methods of Ross and Fahroo [66] are direct methods, in the sense that they apply a nonlinear programming solver to a parametrization of the control and state, but at the same time make hypotheses on the structure of interior and boundary arcs, similar to those used in indirect methods.

Note that (especially in the case of state constraints) checking if various regularity conditions needed by the indirect approach

Note that (especially in the case of state constraints) checking if various regularity conditions needed by the indirect approach hold, and writing the optimality conditions correctly, may be a delicate task [59]. It may also happen that little is known on the structure of the solution of the problem. In that case it may be preferable to use direct methods, that are as easy to use as any nonlinear programming solver: in other words, finding a correct initial point requires often some intuition from the user, while some scaling and tuning of algorithmic parameters may help. We refer to Betts [7, 23] for an overview of numerical methods for optimal control problems and comparison of various approaches.

Direct methods generally solve the optimality system of the discretized problem by using Newton or quasi-Newton algorithms. An early reference on Newton's method for (unconstrained) optimal control problems is Mitter [61]. Following the discover of sequential quadratic programming (SQP) algorithms by Han [50] and Powell [65], a first generation of direct methods in which only control variables appear in the optimization solver were proposed by Kraft [55]. Implementations of that time used full matrices and full quasi-Newton

approximations of Hessian of Lagrangian (based, for instance, on the BFGS update). This induced a severe limitation of dimension. Next appeared sparse implementations of SQP algorithms and their application to optimal control problems, using sparse finite differences for derivatives (Betts and Huffman [13]), or exact values of derivatives (Bonnans and Launay [16]). In these codes the optimization variables are both control and states, and the state equation is an equality constraint of the optimization solver. In both cases the quadratic programming (QP) algorithm consists in an active set strategy combined with a reduction of variables similar to reduced gradient methods, see e.g. Section 6.3 in Gill, Murray and Wright [43]. Therefore if many constraints are active, the basis matrix may need a large memory. Also, as for all active set strategies for large scale problems, the number of inner iterations (of the QP) may become extremely large (since typically the active set changes by one component at most at each iteration; it is true, however, that the cost of an active-set iteration is usually smaller than the one of an interior-point algorithm).

These drawbacks pushed some authors to propose interior-point algorithms for optimal control problems. Wright [76] analyses some path-following algorithms for solving convex quadratic problems, and applies them to the resolution of QPs arising in SQP algorithms for general optimal control problems. Vicente [73] discusses interior-point methods for nonlinear programs but (despite the title) optimal control problems are only a motivation for the paper and not really discussed. Leibfritz and Sachs [56] analyse SQP algorithms with inexact resolution of the QP, and apply this analysis to the resolution of the QP by interior-point algorithms. Bergounioux et al. [6] give numerical comparisons of two interior-point solvers to an active-set strategy algorithm. Wächter and Biegler [74] developed the interior-point nonlinear programming solver IPOPT; the latter is applied to an optimal control problem (optimization of a chemical process) in Jockenhövel, Biegler and Wächter [52]. In the context of optimal control of partial differential equations, let us quote Refs. [6, 12].

All the above papers put the emphasis on the optimization solver, and do not take into account the influence of discretization errors. An exception is Betts [8] where a refinement strategy is proposed. This strategy is heuristic and takes into account only discretization errors for the state equation.

There is also a literature specifically devoted to the analysis of discretization errors, mainly the analysis of distance between the solution of the continuous problem and the one of the discretized problem; we review some of its results at the end of section 5.2.

Our contribution is to present a direct type algorithm based on three main features. First, it is an interior-point algorithm based on the logarithmic

penalty of inequality constraints. An advantage of these algorithms is their ability to find a solution in a relatively small number of iterations. Second, we use a QR factorization for band matrices that takes full advantage of the dynamic structure of the data: the cost of factorization is proportional to the number of time steps. Third, we have a flexible discretization refinement procedure, that is activated at each major iterations of the algorithm, i.e., when after solving the optimality system for a given value of the optimization parameter. This is possible for two reasons: (i) since the penalized problem is unconstrained, we are able to compute (discretization) error estimates, and (ii) the structure of the interior-point algorithm allows to add easily new discretization points (this is not so easy with active set strategies). We apply the method in particular to the atmospheric reentry problem.

The chapter is organized as follows. We review the theory of error estimates for unconstrained optimal control problems, and some extensions to constrained problems, in Section 5.2. Then we introduce an “optimal refinement” procedure in Section 5.3. Fast dedicated numerical algebra is presented in Section 5.4. The interior-point approach for constrained problems is discussed in Section 5.5. Then we turn to numerical results. Section 5.6 is devoted to Goddard’s problem. Section 5.7 presents an application to Fuller’s problem. Finally reentry problems are dealt with in section 5.8.

5.2 Unconstrained problems: error estimates

In this section we briefly review the recent analysis of error estimates for unconstrained optimal control problems done in [17]. It combines the formulation of the optimality system of the discretized problem due to Hager [45], with the theory of error orders for one step methods. Adding an additional state variable if necessary, we may assume that the cost is a function of the final state:

$$\text{Min } \Phi(y(T)); \quad \dot{y}(t) = f(y(t), u(t)), \quad t \in [0, T]; \quad y(0) = y^0. \quad (5.1)$$

Here f and Φ are C^∞ functions $\mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $\mathbb{R}^n \rightarrow \mathbb{R}$, respectively. Use now a Runge-Kutta type discretization, where the control variable is discretized similarly to the state variable, i.e., with each “inner state” y_{ki} is associated an inner control u_{ki} :

$$\begin{cases} \text{Min } & \Phi(y_N); \\ y_{k+1} &= y_k + h_k \sum_{i=1}^s b_i f(y_{ki}, u_{ki}), \quad k = 0, \dots, N-1, \\ y_{ki} &= y_k + h_k \sum_{j=1}^s a_{ij} f(y_{kj}, u_{kj}), \quad i = 1, \dots, s, \\ y_0 &= y^0. \end{cases} \quad (5.2)$$

The positive time steps h_k are such that $\sum_{k=0}^{N-1} h_k = T$. The Runge-Kutta scheme is parameterized by its coefficients (a, b) , an $s \times s$ matrix and a vector of \mathbb{R}^s respectively. Assuming all coefficients b_i to be nonzero, it was shown by Hager [45] that (after some change of variables in the Lagrange multipliers) the optimality system can be written in the following form:

$$\left\{ \begin{array}{ll} y_{k+1} = y_k + h_k \sum_{i=1}^s b_i f(y_{ki}, u_{ki}), & k = 0, \dots, N-1, \\ y_{ki} = y_k + h_k \sum_{j=1}^s a_{ij} f(y_{kj}, u_{kj}), & i = 1, \dots, s, \\ p_{k+1} = p_k - h_k \sum_{i=1}^s \hat{b}_i f_y(y_{ki}, u_{ki})^\top p_{ki}, & k = 0, \dots, N-1, \\ p_{ki} = p_k - h_k \sum_{j=1}^s \hat{a}_{ij} f_y(y_{kj}, u_{kj})^\top p_{kj}, & i = 1, \dots, s, \\ 0 = f_u(y_k, u_k)^\top p_k, & k = 0, \dots, N-1, \\ 0 = f_u(y_{ki}, u_{ki})^\top p_{ki}, & i = 1, \dots, s, \\ y_0 = y^0, \quad p_N = \Phi'(y_N), \end{array} \right. \quad (5.3)$$

where

$$\hat{b}_i := b_i \quad \text{and} \quad \hat{a}_{ij} := (b_i b_j - b_j a_{ij}) / b_i, \quad \text{for all } i \text{ and } j. \quad (5.4)$$

The above system may be interpreted as a *partitioned Runge-Kutta discretization scheme* (i.e., a Runge-Kutta discretization with coefficients that depend on the index of the differential variable) for the optimality conditions, stated below, of problem (5.1):

$$\left\{ \begin{array}{ll} \dot{y}(t) = f(y(t), u(t)), & t \in [0, T], \\ \dot{p}(t) = -f_y(y(t), u(t))^\top p(t), & t \in [0, T], \\ p(T) = \Phi'(y(T)), \quad y(0) = y^0, \\ 0 = f_u(y(t), u(t))^\top p(t), & t \in [0, T]. \end{array} \right. \quad (5.5)$$

If the Hamiltonian function $H(y, u, p) := p \cdot f(y, u)$ is, in the neighborhood of the optimal trajectory, a strongly convex function of u , then we can eliminate the control from the above algebraic constraint (thanks to the implicit function theorem) so that (5.5) reduces to a two points boundary value problem. In that case, if the solution of the discretized problem is, as we may hope, close to the one of the continuous problem, we can as well eliminate controls from the algebraic equations in (5.3). We obtain so-called continuous and discrete *reduced* (i.e., without control variables) formulations, and the discrete reduced formulation appears as a partitioned Runge-Kutta discretization of (5.5).

The theory of error orders for partitioned Runge-Kutta discretization schemes is now well-understood, see [46, 48, 49]. When (5.4) holds it can be proved that the scheme is symplectic (see e.g. [46] for a detailed study of symplectic schemes). Conditions for error order up to 4 were obtained “by hands” in [45]. One may find in [17] a simplification of order conditions derived from [46, 48, 49] for symplectic schemes, that allows to state them for order up to 6.

The above mentioned results apply only for unconstrained problems with strongly convex Hamiltonians. Dontchev et al. [33] show that, for problems with control constraints, high order Runge-Kutta schemes typically will give (because of the constraint) an error of order two. For mixed control and state constraints, Malanowski et al. obtain an error estimate of the order of the stepsize. A similar result is obtained by Dontchev and Hager [32] for first-order state constraints. Extending these results to other classes of optimal control problem (as for Goddard's problem of section 5.6) is a significant challenge.

5.3 Mesh Refinement

Consider an ordinary differential equation, whose end point conditions will be discussed later:

$$\dot{z}(t) = F(z(t)), \quad t \in [0, T], \quad (5.6)$$

where F is a C^∞ mapping $\mathbb{R}^n \rightarrow \mathbb{R}^n$. The corresponding one step discretization is

$$z_{k+1} = z_k + h_k \Phi(z_k, h_k), \quad (5.7)$$

where $\Phi : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n$. Again $\{h_k\}$ are positive time steps, such that $\sum_{k=0}^{N-1} h_k = T$. For a Cauchy problem, i.e., if the initial condition $z(0)$ is given, the value of time steps is obtained by induction: given all h_i for $i < k$, as well as z_k , define the local error at step k as $e_k := \|\hat{z}_k(h_k) - z_{k+1}\|$, where $\hat{z}_k(t)$ satisfies $\hat{z}_k(0) = z_k$ and $\dot{\hat{z}}_k(t) = F(\hat{z}_k(t))$. The theory of error estimates tells us that the local error on step k is (at least for the Runge-Kutta schemes that we have in mind) of the form $e_k = C_k h_k^{p+1} + o(h_k^{p+1})$, where the order p is, in principle, known (or can be identified numerically). The error estimate, for a first trial of the value of h_k , may therefore be estimated by comparing the local variation of the differential variable with the one computed by a scheme of higher order. This gives, if the time step is small enough, an estimate \hat{e}_k of the "true" local error e_k , satisfying

$$\hat{e}_k = C_k h_k^{p+1} + o(h_k^{p+1}); \quad (5.8)$$

the latter can be used to set h_k to a value that ensures that the local error is below a given threshold.

For a two points boundary value problem, the situation is different. Typically the discretized problem is solved first using a rough estimate of the size of time steps, and then one tries to refine the mesh, i.e., to add other discretization points. Consider the *optimal refinement problem*, which is defined as problem of having the sum of local errors below a given threshold

E . Since $e_k = C_k h_k^{p+1} + o(h_k^{p+1})$, dividing the step h_k into q_k smaller steps of size h_k/q_k , where q_k is a positive integer, reduces the principal term of the local error to $C_k (h_k/q_k)^{p+1}$ on each smaller step, i.e. a total of e_k/q_k^p on the q_k steps. Neglecting the contribution of $o(h_k^{p+1})$ to the local error, we see that the optimal refinement problem can be formulated as follows:

$$\text{Min}_{q \in \mathbb{N}_*^N} \sum_{k=1}^N q_k; \quad \sum_{k=1}^N \frac{\hat{e}_k}{q_k^p} \leq E. \quad (5.9)$$

Here \mathbb{N}_* denotes the set of positive integers, and \hat{e}_k is again the estimate of local error obtained with a higher order scheme. We should notice here, that we define the optimal refinement problem with a constraint over the sum of local error, which is not the global integration error. Indeed, global error should take into account the propagation of local error through the integration of the ODE. In our case we have a Two Points Boundary Value Problem (TPBVP) unless Cauchy problem where the state is fully defined at the first time, here error propagates not only from the initial value but also from the final value for specific indexes. This means that the global error is this sum of local error with a ponderation.

Let us now prove that this nonlinear integer programming problem can be solved at low price by the algorithm below:

Algorithm 11. (Optimal refinement)

For $k = 1, \dots, N$ **do** $q_k := 1$. **End for**
While $\sum_{k=1}^N e_k/q_k^p > E$ **do**
 Compute $k_g \in \text{argmax}_k \{e_k (1/q_k^p - 1/(q_k + 1)^p)\}$
 $q_{k_g} := q_{k_g} + 1$.
End While

Call local gain the amount $g_k := \hat{e}_k (1/q_k^p - 1/(q_k + 1)^p)$. This is the amount by which the estimate of sum of local errors is decreased by increasing q_k by one. The algorithm consists simply in adding points where the local gain is maximal.

Lemma 12. Algorithm 11 stops after a finite number of steps, and its output is the solution of problem (5.9).

Proof. We leave to the reader the proof of finiteness of Algorithm 11, and prove its optimality. Consider the related problem of minimizing the sum of estimates of local errors, subject to a given number of added points:

$$(D_m) \quad \text{Min}_{q \in \mathbb{N}_*^N} \sum_{k=1}^N \frac{\hat{e}_k}{q_k^p}; \quad \sum_{k=1}^N q_k = N + m.$$

The value function of this problem denoted by E_m , is strictly decreasing (we may assume all \hat{e}_k to be positive) and has limit 0 when $m \uparrow +\infty$. If $E_0 < E$ then there is no need to add points and the conclusion holds. Otherwise, there exists a unique $r \in \mathbb{N}_*$ such that

$$E_r \leq E < E_{r-1}. \quad (5.10)$$

Any solution of (D_r) is solution of (5.9), since by the definition of r it is not possible to satisfy the constraint of (5.9) by adding less than E_r points. We end the proof by showing that the output of Algorithm 11, denoted by $\{\bar{q}_k\}$, is solution of (D_r) .

The proof is by induction over the values of $r \in \mathbb{N}$ such that (5.10) holds. If $r = 0$ the conclusion holds (no point is added). Assume that it holds for all integer up to $r - 1$, and take a value of E such that (5.9) has value $N + r$. Let \bar{q} be the output of Algorithm 11, and let k_0 be the index for which a time step has been added at the last step of algorithm 11. Set $\hat{q}_k = \bar{q}_k$ for all $k \neq k_0$, and $\hat{q}_{k_0} = \bar{q}_{k_0} - 1$.

Let $\{q_k\}$ satisfy the constraint of (D_r) . We have to prove that the amount $\Delta := \sum_{k=1}^N \hat{e}_k (1/\bar{q}_k^p - 1/q_k^p)$ is nonpositive. There exists i such that $q_i > 1$. Set $\tilde{q}_k = q_k$ for all $k \neq i$, and $\tilde{q}_i = q_i - 1$. We may write $\Delta = \Delta_1 + \Delta_2$, where

$$\Delta_1 := \sum_{k=1}^N \hat{e}_k \left(\frac{1}{\hat{q}_k^p} - \frac{1}{\tilde{q}_k^p} \right); \quad (5.11)$$

$$\Delta_2 := \hat{e}_{k_0} \left(\frac{1}{\bar{q}_{k_0}^p} - \frac{1}{(\bar{q}_{k_0} - 1)^p} \right) - \hat{e}_i \left(\frac{1}{q_i^p} - \frac{1}{(q_i - 1)^p} \right). \quad (5.12)$$

Since by our induction $\{\hat{q}_k\}$ is solution of (D_{r-1}) , and \tilde{q} is feasible for (D_{r-1}) , we have that $\Delta_1 \leq 0$. Let us prove that $\Delta_2 \leq 0$ for a certain i . If $\bar{q}_{k_0} \leq q_{k_0}$, then we may take $i = k_0$. Otherwise, in view of the constraint of (D_r) , we may take $i \neq k_0$ such that $\bar{q}_i < q_i$. Since k_0 is the index of maximal local gain we have that

$$\Delta_2 \leq \hat{e}_i \left(\frac{1}{\bar{q}_i^p} - \frac{1}{(\bar{q}_i - 1)^p} \right) - \hat{e}_i \left(\frac{1}{q_i^p} - \frac{1}{(q_i - 1)^p} \right) < 0. \quad (5.13)$$

The conclusion follows. \square

Remark 13. A fast implementation of algorithm 11 is obtained by sorting the time steps along the values of the local gain, and that needs at most $O(N \log N)$ operations. Each step of the algorithm needs to update this ordering, which needs at most $O(\log N)$ operations. We obtain that the algorithm needs at most $O((N + m) \log N)$ operations. This is negligible with respect to the number of operations needed for computing Newton steps, as we will see in the next section.

5.4 Linear algebra

Solving the discrete optimality system (5.3) by a Newton step needs to solve linear systems whose Jacobian matrix is, when variables are ordered by increasing values of time, a band matrix. We recall some classical results on band matrices, referring e.g. to [5] for details. Given a square matrix A of size n_A , we denote q the band size, i.e., the smallest integer such that $A_{ij} = 0$ if $|i - j| > q$ (for instance, a diagonal matrix has band size 0).

Lemma 14. The computation of the QR factorization, based on Givens rotations, of a matrix A of band size q is such that the (upper triangular) factor R has band size $2q$. The number of operations for factorization is of order $q^2 n_A$, and the number of operations for solving the linear system (after factorization) is of order $q n_A$.

We apply this result to the Jacobian of (5.3). We may assume that $m = O(n)$. Then the band size satisfies $q = O(sn)$, where s is the number of inner steps in the Runge-Kutta scheme, and $n_A = O(snN)$. We obtain the following.

Lemma 15. The computation of the QR factorization, based on Givens rotations, of the Jacobian of (5.3) needs $O(s^3 n^3 N)$ operations for factorization, and $O(s^2 n^2 N)$ operations for solving the linear system (after factorization).

Remark 16. (i) The cost of factorization of the Jacobian is of the same order than the integration of the state equation (the control being given) by a fully implicit scheme, and the ratio is of order s^2 for a semi implicit scheme (with $O(s)$ implicit steps). This means that the computation of the Newton step is not much more expensive than the simulation of the system by an implicit scheme.

(ii) The ratio of number of operations between factorization and solving is of order sn . This means that, generally, factorization will be by far the most expensive part of resolution.

(iv) For constrained problems we must add rows (constraints) and variables (Lagrange multipliers). However, provided the number of constraints is of order n , the conclusion of Lemma 15 also holds in that case.

5.5 Interior-point algorithms, constrained problems

5.5.1 Interior point

Consider now a constrained optimal control problem of the following form:

$$\text{Min } \int_0^T \ell(y(t), u(t))dt + \Phi(y(T)); \begin{cases} \dot{y}(t) = f(y(t), u(t)), & t \in [0, T]; \\ 0 \leq g(y(t), u(t)), & t \in [0, T]; \\ y(0) = y^0. \end{cases} \quad (5.14)$$

Here $g : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^r$ is a mapping of class C^∞ . The idea of logarithmic penalty is, for a (small) parameter ε , to introduce the penalizer running cost

$$\ell_\varepsilon(y, u) := \begin{cases} \ell(y, u) - \varepsilon \sum_{i=1}^r \log[g_i(y, u)] & \text{if } g_i(y, u) > 0, i = 1, \dots, r, \\ +\infty, & \text{otherwise,} \end{cases} \quad (5.15)$$

and the related penalized optimal control problem:

$$\text{Min } \int_0^T \ell_\varepsilon(y(t), u(t))dt + \Phi(y(T)); \begin{cases} \dot{y}(t) = f(y(t), u(t)), & t \in [0, T], \\ y(0) = y^0. \end{cases} \quad (5.16)$$

A feasible point of this problem is such that the state equation holds, the inequality $g_i(y(t), u(t)) > 0$ a.e., and in addition, $\log[g_i(y(t), u(t))]$ is Lebesgue integrable, for all $i = 1$ to r .

Little is known on the theoretical properties of logarithmic penalty for optimal control problems. The only reference we know is Bonnans and Guillebaud [15]. Under the hypothesis that running constraints are in fact bound constraints on controls, they prove convergence of optimal control and states (and also costates) for either (strongly) convex linear quadratic problems, or in a larger class in which the Hamiltonian function is convex with respect to the control. In addition it is proved that controls remain at distance at least $c\varepsilon$ from its bounds, where $c > 0$ is uniform for all sufficiently small ε . Asymptotic expansions of the solution for small ε might help for designing efficient algorithms; a rather specific case of control constraints is discussed in Alvarez, Bonnans and Laurent-Varin [1]. Nothing seems to have been done for problems with state constraints.

In practice an initial feasible point is often unknown. In that case it is advisable to rewrite the running constraint as

$$g(y(t), u(t)) - e(t) = 0; \quad e(t) \geq 0, \quad t \in [0, T], \quad (5.17)$$

and the penalized problem as

$$\begin{aligned} \text{Min } & \int_0^T \left[\ell(y(t), u(t)) - \varepsilon \sum_{i=1}^r \log[e_i(t)] \right] dt + \Phi(y(T)); \\ & \dot{y}(t) = f(y(t), u(t)); \quad g(y(t), u(t)) - e(t) = 0; \quad e(t) \geq 0, \quad t \in [0, T]; \\ & y(0) = y^0. \end{aligned} \quad (5.18)$$

Clearly problems (5.16) and (5.18) are equivalent. The advantage of formulation (5.18) is that, given a starting point for which the slack variable is positive, we may still consider Newton steps on the optimality system (even if the equality constraints are not satisfied). So our practical implementations are based on formulation (5.18).

The penalized problem (5.16) is unconstrained with an integral and final cost. Adding an additional state variable y_{n+1} such that $y_{n+1}(0) = 0$ and $\dot{y}_{n+1}(t) = \ell_\varepsilon(y(t), u(t))$ allows to reduce it to a problem with final cost only, that can be discretized by the Runge-Kutta schemes discussed in subsection 5.2. For a given value of ε the error analysis of that section applies when the steps vanish. The constants, however, will depend on ε .

An interesting open problem is to obtain error estimates for given parameter ε and discretization steps. This is obviously difficult, since without logarithmic penalty, one may encounter a reduction order when state constraints are active (it is known that in general a Runge-Kutta scheme applied to an algebraic differential system suffers from order reduction). In addition junction points for constraints (i.e., times when the constraint starts or stops being active) need a specific analysis.

Related to this is the question of the choice of a path of convergence, i.e., at which relative speed should the parameter ε and the discretization steps converge to 0. Again this is essentially an open problem. In our present implementation we simply require that the error estimate be not greater than a constant times ε .

The idea of logarithmic penalty could as well have been used on a discretized version of problem (5.14). In view of the discussion of section 5.2, a natural extension of discretization (5.2) would be

$$\begin{cases} \text{Min } & \Phi(y_N); \\ & y_{k+1} = y_k + h_k \sum_{i=1}^s b_i f(y_{ki}, u_{ki}), \quad k = 0, \dots, N-1, \\ & y_{ki} = y_k + h_k \sum_{j=1}^s a_{ij} f(y_{kj}, u_{kj}), \quad i = 1, \dots, s, \\ & 0 \leq g(y_{ki}, u_{ki}), \quad i = 1, \dots, s, \\ & y_0 = y^0. \end{cases} \quad (5.19)$$

where we use here the extended formulation: y has $n+1$ components and $f_{n+1} = \ell$. It appears that the logarithmic penalization of (5.19) is nothing

but the Runge-Kutta discretization of (5.18). This remark is useful since, if we estimate the distance of the solution u_h^ε of the discretized problem to the solution \bar{u} of the original one by the following inequalities (for conveniently chosen norms) then each norm in the right hand side corresponds either to a discretization estimate or to the estimate of variation due to logarithmic penalty:

$$\begin{cases} \|\bar{u} - u_h^\varepsilon\| \leq \|\bar{u} - u^\varepsilon\| + \|u^\varepsilon - u_h^\varepsilon\|, \\ \|\bar{u} - u_h^\varepsilon\| \leq \|\bar{u} - u_h\| + \|u_h - u_h^\varepsilon\|. \end{cases} \quad (5.20)$$

Here u^ε (resp. u_h) denotes the solution of problem (5.16) (resp. (5.19)).

5.5.2 The Dogleg procedure

The first-order optimality conditions of the optimal control problem with logarithmic penalty is a set of nonlinear equations, say of the form $F(X) = 0$ where X is the set of optimization parameters and Lagrange multipliers, and F is a smooth mapping with square Jacobian matrix. We solve the nonlinear equation by applying a Dogleg method to the least square formulation $\text{Min}_X \|F(X)\|^2$, as in Moré and Sorensen [62] and Powell [64].

5.6 Application to Goddard's problem

Goddard's problem, stated in 1919 (see e.g. Maurer [58] or Tsiotras and Kelley [71]) consists in maximizing the final altitude of a rocket with vertical ascent. The model involves three state variables: altitude h , speed v and mass m , and the thrust F as control variable. The objective is to maximize the final altitude, with a prescribed final mass, and free final time. The dynamics is

$$\begin{cases} \dot{h}(t) &= v(t), \\ \dot{v}(t) &= (F(t) - D(v(t), h(t)))/m(t) - g(h(t)), \\ \dot{m}(t) &= -F(t)/c \end{cases} \quad (5.21)$$

where D is the drag function, with arguments speed and altitude, $g(h)$ is the gravity field, and $c > 0$ is the speed of ejection of gas. The final mass is given, and we have a bound on thrust: $0 \leq F(t) \leq F_{\max}$.

Starting from a speed close to zero, it happens that the optimal thrust is not (as numerical experiments show) to set the thrust at F_{\max} as long as the final mass is not attained. The reason is that aerodynamic forces would, in that case, tend to reduce significantly the speed. Therefore (although this is apparently not proved yet) the optimal law is to set the thrust at F_{\max} for a certain time, then to set it to values in $(0, F_{\max})$ (this is the singular arc) and finally to set it to zero. The term singular arc comes from the fact that, since

the control appears linearly in the dynamics (and not in the cost function) it cannot be computed by minimizing the Hamiltonian function when the latter is a constant function of the control. There exists a nice piece of theory that allows to obtain the control law during the singular arc (see [22, 71]), but of course we do not use it in our numerical experiments.

Although our code may use arbitrary Runge-Kutta coefficients, we have used the Gauss-Legendre scheme of order 2 and of order 4, so that we may see how the code behaves when there are many time steps. The results are synthetized in Table 5.1 and Table 5.2, where for each major iteration, corresponding to the resolution of the penalized problem for a given value of ε . We display the values of the major iteration It , the number of dogleg iterations n_{it} , the size N of the mesh, the number of points to be added (predicted by the refinement procedure), the current value of ε and the threshold on error estimates. Observe the great accuracy of the refinement procedure, that essentially predicts in one shot the points to be added, for order 2 as well for order 4 schemes. The number of inner iterations remains small. It is fair to say, however, that the reduction procedure for ε is here quite conservative (division by 2). Stronger reductions will be performed in the other applications presented in this paper.

The optimal control and states are displayed on figures 5.1-5.4, for each value of the parameter ε . We observe the barrier effect for large ε , and the convergence to a three arc-structure, one of them being singular.

We draw in figures 5.6 and 5.7, for each value of ε , the switching function derivative of Hamiltonian function w.r.t. the control), whose expression is $p_v/m - p_m/c$. For our numerical test we choose $D(v, h) := K_D v^2 \exp(-\frac{h_c}{h_0}(h - h_0))$ and $g(h) = g_0 h_0^2 / h^2$, with $K_D = 310$, $c = 0.5$, $h_0 = 1$, $h_c = 500$ and $F_{\max} = 3.5$.

The density of mesh, after each major iteration is displayed in figures 5.8 and 5.9. The original mesh has 100 equal steps. We can observe on the final mesh a high density in the region corresponding to the singular arc.

5.7 Fuller's problem

Let us show how our method behaves when applied to Fuller's problem [39, 40]. This is a simple academic problem with, as we will see, a non regular junction point. The problem is as follows:

$$\text{Min } \frac{1}{2} \int_0^T x_1^2(t) dt; \quad x_1(t) = x_2(t), \quad \dot{x}_2(t) = u(t) \in [-1, 1], \quad t \in [0, T]; \quad x(0) = a,$$

It	N	n_{it}	Points	ε	E
1	100	13	+43	1.0e-3	5.0e-4
	143	3	+1		
	144	3	+0		
2	144	6	+63	5.0e-4	2.5e-4
	207	3	+1		
	208	3	+0		
3	208	6	+102	2.5e-4	1.25e-4
	310	3	+0		
4	310	6	+163	1.25e-4	6.25e-5
	473	3	+0		
5	473	5	+283	6.25e-5	3.125e-5
	756	3	+0		
6	756	5	+484	3.125e-5	1.5625e-5
	1240	3	+0		
7	1240	5	+862	1.5625e-5	7.8125e-6
	2102	2	+0		
8	2102	5	+1458	7.8125e-6	3.90625e-6
	3560	2	+0		
9	3560	5	+2663	3.90625e-6	1.95313e-6
	6223	2	+0		

Table 5.1: Goddard's Problem: order 2 scheme

It	N	n_{it}	Points	ε	E
1	100	14	+0	1.0e-3	5.0e-4
2	100	8	+0	5.0e-4	2.5e-4
3	100	8	+0	2.5e-4	1.25e-4
4	100	9	+0	1.25e-4	6.25e-5
5	100	10	+1	6.25e-5	3.125e-5
	101	6	+0		
6	101	15	+9	3.125e-5	1.5625e-5
	110	7	+1		
	111	5	+0		
7	111	7	+21	1.5625e-5	7.8125e-6
	132	8	+0		
8	132	9	+19	7.8125e-6	3.90625e-6
	151	8	+1		
	152	3	+0		
9	152	6	+39	3.90625e-6	1.95313e-6
	191	4	+0		
10	191	6	+69	1.95313e-6	9.76563e-7
	260	7	+0		

Table 5.2: Goddard's Problem: order 4 scheme

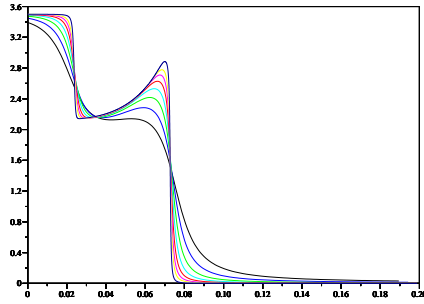


Figure 5.1: Thrust law / Time

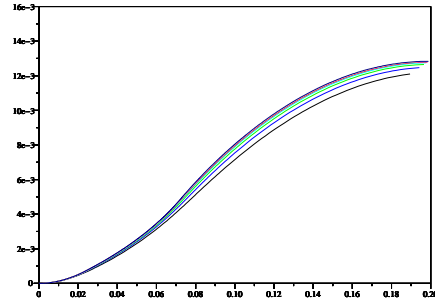


Figure 5.2: Altitude / Time

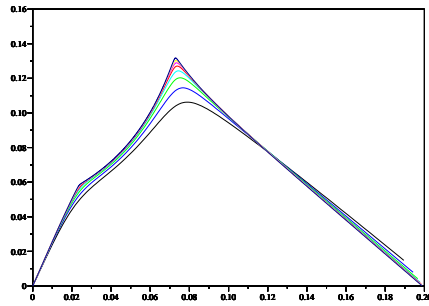


Figure 5.3: Velocity / Time

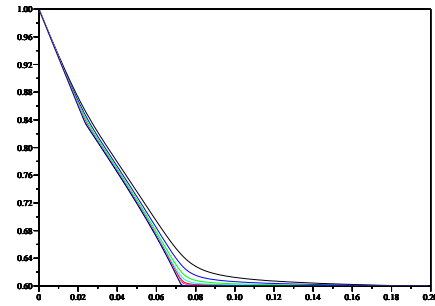


Figure 5.4: Mass / Time

Figure 5.5: Goddard's problem

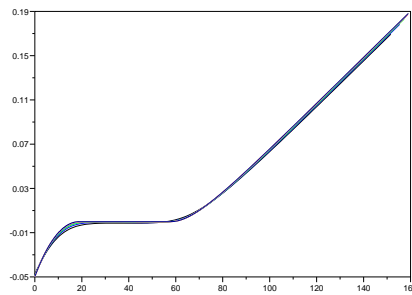


Figure 5.6: Switching function (Goddard)

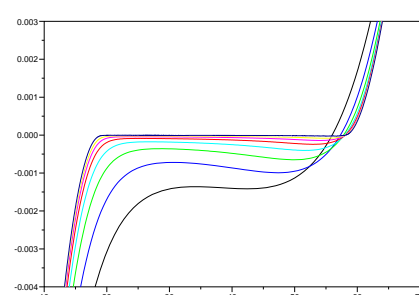


Figure 5.7: Zoom on Switching function

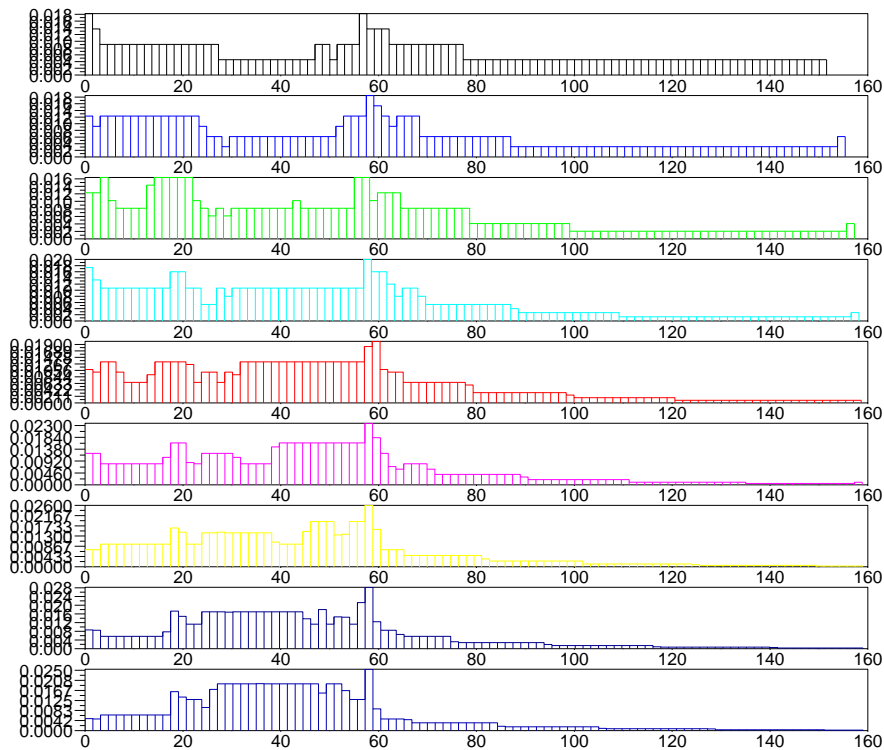


Figure 5.8: Mesh order 2 (Goddard)

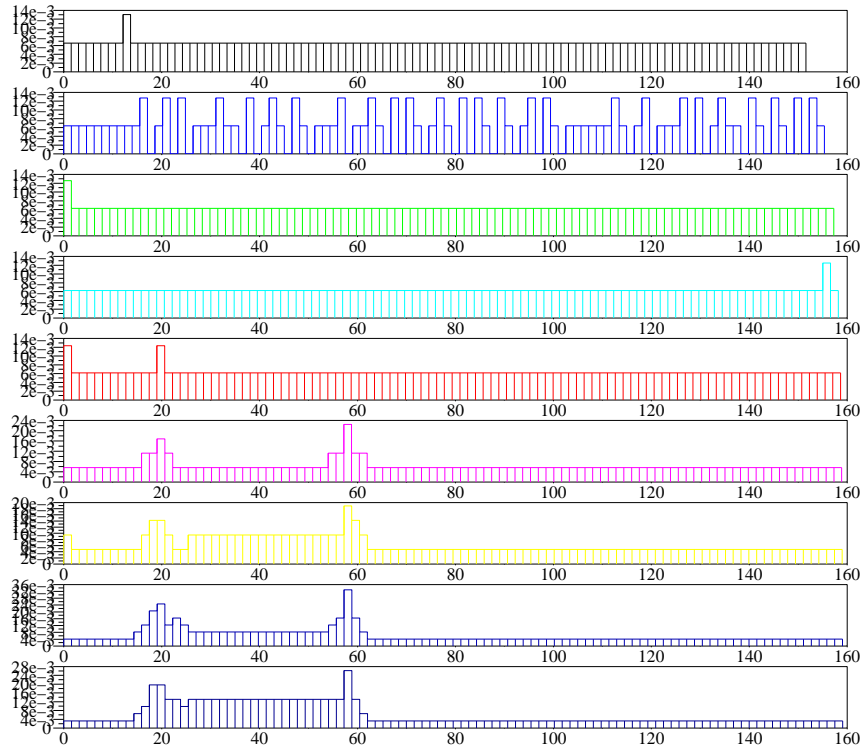


Figure 5.9: Mesh order 4 (Goddard)

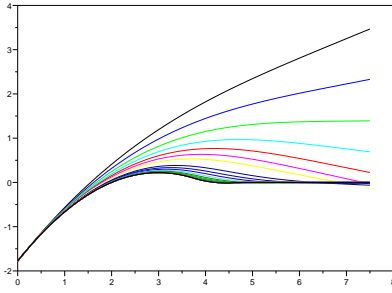
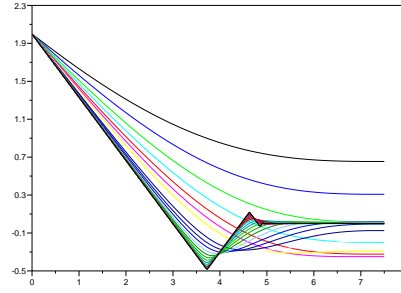
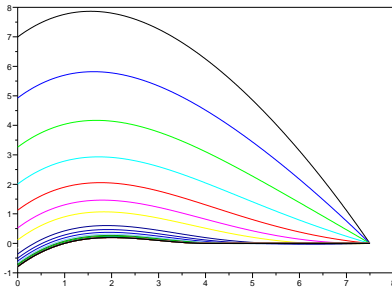
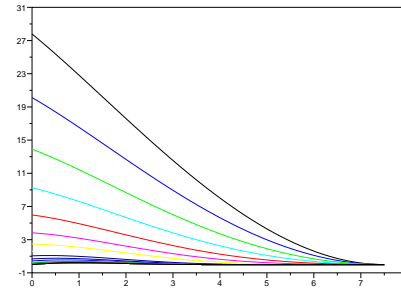
Figure 5.10: x_1 / TimeFigure 5.11: x_2 / TimeFigure 5.12: p_1 / TimeFigure 5.13: p_2 / Time

Figure 5.14: Fuller's problem

where $a \in \mathbb{R}^2$ is given. Note that the Hamiltonian function $H(x, u, p) = \frac{1}{2}x_1^2 + p_1x_2 + p_2u$ is, as in Goddard's problem, linear w.r.t. the control variable. It was shown in [39, 40] that an infinite number of switches may occur before reaching the singular arc where the state has a null value.

On this example the refinement procedure had difficulties, probably in relation with the above mentioned behavior, so we display the results with a fixed grid of 400 points. We set a to the value $(-1.778, 2.0)$ in order to obtain an infinite number of switches.

Figures 5.10 to 5.14 show the states, costate and control variables. It happens that the algorithm does not converge if ε decreases too rapidly. Therefore we choose a linear evolution of the form : $\varepsilon_{k+1} = \frac{1}{2}\varepsilon_k$. We start with $\varepsilon_0 = 16$, and stop for $\varepsilon_{36} = 4.65710^{-10}$.

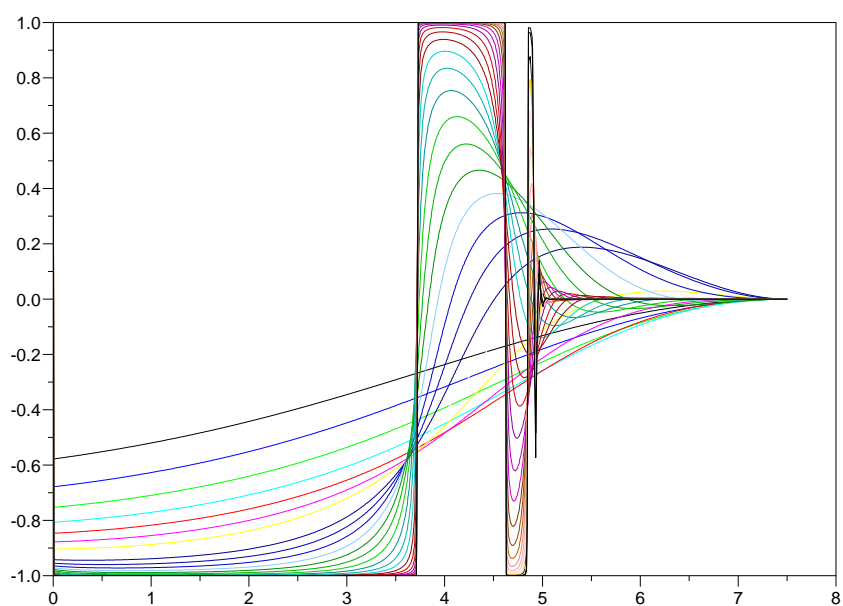


Figure 5.15: u / Time (Fuller)

5.8 Atmospheric Reentry

5.8.1 Model

In this section we study a problem of atmospheric reentry for a space shuttle, already considered in Betts [8, Chap 5, p. 133] to which we refer for a precise statement of state equation and constraints. It is enough to say that there are six state variables: altitude, longitude, latitude, velocity, flight path angle, and azimuth. The two control variables are the angle of attack and bank angle. There are bound constraints on controls, and an optional running constraints on the heating flux, all of them identical to those in [8]. In addition we have an optional running constraint of non positive flight path angle, which prevents skipping.

We maximize either the cross range or the forward range. Since we start at latitude 0 moving eastward, cross range maximization amounts to maximize the final latitude, while range maximization amounts to maximize the final longitude. We use again the Gauss-Legendre discretization schemes.

We number problems as follows (the third column refers to constraints other than bounds on control variables):

Problem Number	Cost function	Constraints
1	Max Cross range	None
2	Max Cross range	Heating
3	Max Cross range	Path Angle
4	Max Cross range	Heating and path angle
5	Max Range	None
6	Max Range	Heating
7	Max Range	Path Angle

5.8.2 Initialization procedure

As already mentioned, the initialization procedure is an important component for the success of the method, especially for reentry trajectories since it is known that the trajectory is very sensitive to the control.

Max cross range with only control constraints

We initialize variables by the following heuristics: we integrate the optimality system starting from $t = 0$ and with zero initial costate. Of course the final conditions are not satisfied.

Maximum cross range with heating flux or path angle constraints

For these problems we initialize the control, state and costate with the value obtained by solving the problem without heating flux or path angle constraints, with a given value of the penalty parameter $\varepsilon \geq 1$. Then we initialize the slack variables and Lagrange multipliers in the following way:

$$e_i = \min(\max(\tilde{g}_i, 1 - \sqrt{\varepsilon}), -1 + \sqrt{\varepsilon}) \quad (5.22)$$

$$\lambda = \varepsilon(e - 1)^{-1}; \quad \mu = \varepsilon(e + 1)^{-1}. \quad (5.23)$$

Here \tilde{g} is the scaled path constraint function, that (if the constraint holds) belongs to $[-1, 1]$. Thus the inequality conditions on the slack variables are satisfied.

For max range without any constraints

We initialize the problem with the solution obtained for the max cross range cost function, and with $\varepsilon = 16$.

For max range with each constraints

We initialise the problem with the solution obtained for the max range cost function, and with $\varepsilon = 1$.

5.8.3 Update of the penalty parameter

We update the penalty parameter ε in the following way :

$$\varepsilon_{k+1} = \min\left(\frac{1}{2}\varepsilon_k, \varepsilon_k^{3/2}\right).$$

5.8.4 Numerical results for cross range maximization

All figures display the evolution of some variable as a function of time. We first display the results for cross range maximization.

Cross range maximization with only control constraints (Problem 1)

We first display the results for a grid of 100 equidistant points, without refinement. Table 5.3 shows the evolution of the interior point parameter, the number of dogleg iterations for each ε , the CPU time for each ε and the performance index (here, the final latitude), to be compared with the value

34.1412 degree given in [8]. The following graph represents the evolution of the performance index as function of ε .

We observe that, although ε decreases rapidly in the last iterations, the number of Newton steps at each iteration remains small and, in fact, tends to decrease. The time needed for one Newton step is approximately 3 seconds. Figures 5.16 and 5.17 display the performance index and the angle of attack as function of ε .

In table 5.4 we can notice the use the refinement policy. We can notice that numerically, there is no difference between performance index for the resolution without refinement. The density of discretization points for obtaining this precision is displayed in the histogram of Figure 5.20.

There is a fast pull-up manoeuvre close to the final time in order to comply with the prescribed final value of the flight path angle. This can be seen in the display of the angle of attack, and accordingly the density of discretization points is higher at the end of the trajectory.

Figures 5.21 to 5.25 compare the outputs of problems 1 to 4.

Cross range maximization with flux heating constraint (Problem 2)

We display our results for the problem with heating flux (problem 2), without refinement of the initial grid of 100 equidistant points. The execution report is given in Table 5.5, and the values of heating flux and angle of attack, as function of time, for several values of ε , are given in Figures 5.18 and 5.19. We can observe that the angle of attack is now far from being constant, as it almost was in Problem 1. With this additional constraint we obtain a worse (as was to be expected) performance index, by about 10 %.

Cross range maximization with path angle constraint (Problem 3)

We add the constraint of non positive path flight angle. As for heating constraint, we initialize with the unconstrained problem trajectory, and obtain results in Table 5.6. It appears now that the degradation of the criterion is of only about 0.1 %, although the constraint was not satisfied in Problem 1.

Cross range maximization with heating flux and flight path angle constraint (Problem 4)

Problem 4 involves both constraints of flight path angle constraint and heating constraint. The algorithm converges until $\varepsilon = 0.0056$ but not for smaller values (perhaps a more conservative policy for the decrease of ε should be considered). See Table 5.7.

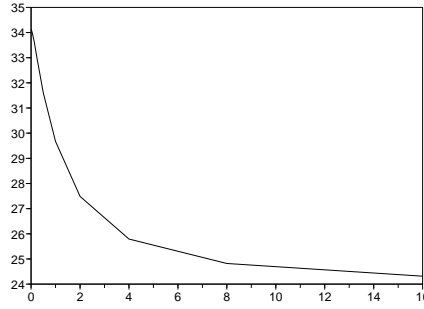


Figure 5.16: Final lat.(°)/ ε (Pb.1)

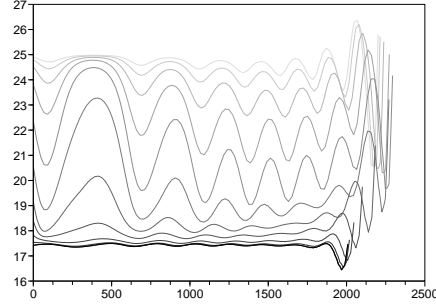


Figure 5.17: Ang. of at.(°)/T.(s) (Pb.1)

Graphs for evolution w.r.t. ε in Problem 1

5.8.5 Numerical results, for range maximization

In the modelisation chosen, we would like to try an other performance index. The natural new one is to maximize the forward range of the Space Shuttle. Figures 5.26 to 5.30 compare the outputs of the various range maximization problems.

Range maximization with only control constraints (Problem 5)

We show in table 5.8 the behavior of the algorithm. The number of dogleg iterations decreases and in the last major iterations is no more than 3. Note that the cost function is almost constant for $\varepsilon \leq 0.125$.

Range maximization with heating flux constraints (Problem 6)

For this problem the number of dogleg iterations is much larger. In the last iterations the cost function does not change much, and hence, we may hope that the last solution computed with $\varepsilon = 0.044$ is not far from the optimum. Interestingly, the problem seems to have three boundary arcs.

Range maximization with path angle constraints (Problem 7)

Here the smallest value of ε for which convergence occurs is $\varepsilon = 8.95e-4$ and again the cost function does not change much in the last iterations, so that we may hope to have computed a reasonably good approximate solution.

ε	Iterations	CPU time	Final latitude
16	37	1 min 12s	24.31
8	10	23s	24.82
4	8	20s	25.79
2	8	21s	27.48
1	7	20s	29.67
0.5	7	18s	31.59
0.25	6	18s	32.94
0.125	7	22s	33.66
$4.41942 \cdot 10^{-02}$	6	19s	34.01
$9.29068 \cdot 10^{-03}$	6	20s	34.11
$8.95512 \cdot 10^{-04}$	5	16s	34.1283
$2.67983 \cdot 10^{-05}$	5	17s	34.1288

Table 5.3: Problem 1: Cross range

ε	Iterations	Final latitude	Time steps	CPU time
16	1	24.31	100	3s
8	10	24.82	100	23s
4	8	25.79	100	23s
2	8+3	27.48	100+2	29s
1	7+3	29.68	102+42	37s
0.5	6+3+2	31.59	144+49+9	1min 45s
0.25	7+3	32.94	202+31	2min 40s
0.125	7+2	33.66	233+27	2min 56s
0.0441942	6+2	34.01	260+93	3min 56s
0.0092907	6+2+2	34.11	353+366+5	17min 9s

Table 5.4: Problem 1: Cross range with refinement

ε	Iterations	CPU time	Final latitude
0.1	13	1min 02s	27.85 deg
0.0316228	5	26s	29.62 deg
0.0056234	4	19s	30.42 deg
0.0004217	5	24s	30.59 deg
0.0000087	6	29s	30.61 deg

Table 5.5: Problem 2: Cross range, heating flux constraint

ε	Iterations	CPU time	Final latitude
0.1	14	39s	33.46
0.032	11	38s	33.95
0.0056	9	35s	34.06
0.00042	7	30s	34.087
0.0000087	9	41s	34.090

Table 5.6: Problem 3: Cross range, path angle constraint

ε	Iterations	CPU time	Final latitude
0.1	15	1 min 19s	26.98
0.032	11	1 min 12s	29.45
0.0056	11	1 min 7s	30.38

Table 5.7: Problem 4: Cross range, heating flux and path angle constraints

ε	Iterations	Longitude	CPU time
16	28	171.53	1min 16s
8	11	176.47	43s
4	9	181.13	36s
2	8	184.55	32s
1	7	186.27	28s
0.5	5	186.89	20s
0.25	3	187.08	14s
0.125	3	187.16	13s
0.0441942	3	187.16	13s
0.00929068	3	187.16	14s
0.000895512	2	187.16	10s
$2.67983e - 05$	2	187.16	13s

Table 5.8: Problem 5: Max range

ε	Iterations	Longitude	CPU time
1	386	167.95	24 min 06s
0.5	21	170.79	1 min 44s
0.25	20	172.36	1 min 36s
0.125	17	173.22	1 min 21s
0.0441942	18	173.86	1 min 26s

Table 5.9: Problem 6: Max range, heating flux constraint

ε	Iterations	Longitude	CPU time
0.125	11	178.09	1 min 09s
0.0441942	5	179.66	0 min 47s
0.00929068	6	180.42	0 min 53s
0.000895512	4	180.61	1 min 11s
$2.67983e - 05$	6	180.63	1 min 23s

Table 5.10: Problem 7: Max range, path angle constraint

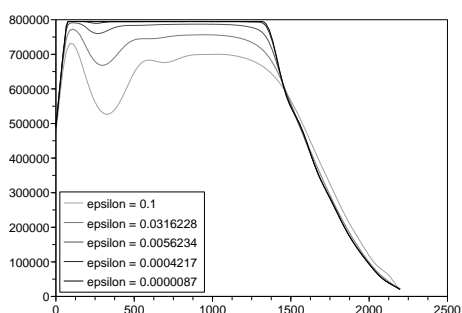
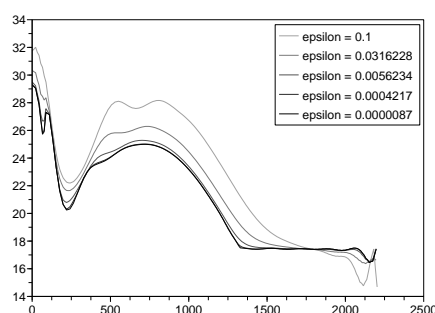
Figure 5.18: Heating (W/m^2)/T.(s)

Figure 5.19: Ang. of at.(°)/T.(s)

Graphs for evolution w.r.t. ε in Problem 2

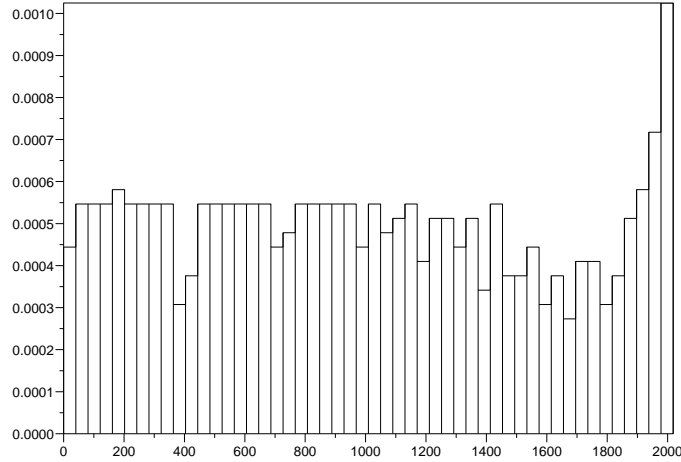


Figure 5.20: Final density mesh in Problem 1

5.9 Conclusion and perspectives

We have presented an approach for solving optimal control problems, based on the interior-point methodology combined with a refinement procedure. Our present implementation allows to solve efficiently various applied problems. The cost of a Newton step is of the same order as a simulation with a fully implicit scheme. The refinement procedure leads to higher computing times, but since the number of operations is proportional to the number of time steps, and the number of added steps seems to be minimal, this is the price to pay for precision (perhaps in some example we could test if some of the time steps could be merged). The software, however, encountered difficulties for the reentry problem when both constraints on flight path angle and heating flux are active, for small values of the parameter ε .

Theoretical advances might help to give precise error estimates (to the solution of the original problem) and to understand what is the optimal path in the (ε, E) plane (E is the estimate of integration errors). Also, our dogleg procedure is quite simple and would deserve to be improved by taking into account the nature of optimality conditions. The way of decreasing the parameter ε is important for the efficiency of the method. We also observed that factorization is much more expensive than solving (when measured by the number of operations; the need for memory is similar). Therefore a simplified Newton strategy might help. It is therefore clear that there is a lot of room for improvement for this method.

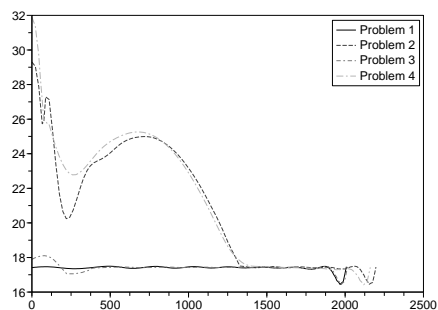


Figure 5.21: Ang. of at.(°)/T.(s)

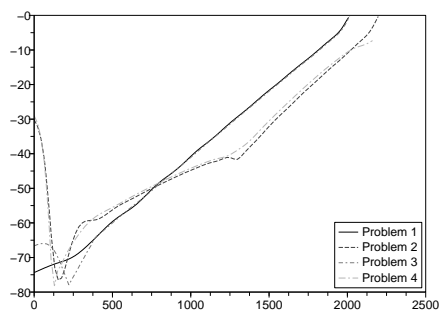


Figure 5.22: Bank ang.(°)/T.(s)

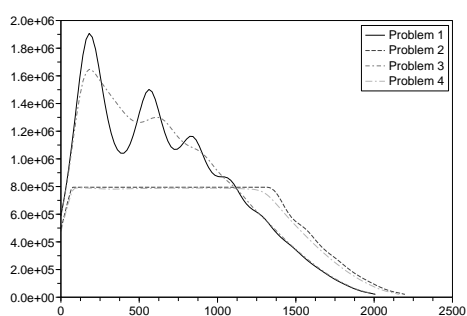
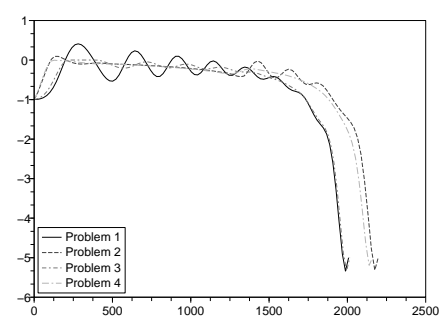
Figure 5.23: Heating(W/m²)/T.(s)

Figure 5.24: Fli. path ang.(°)/T.(s)

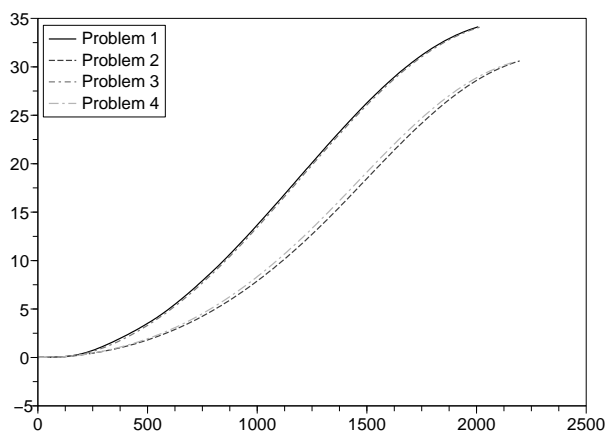


Figure 5.25: Lat.(°)/T.(s)

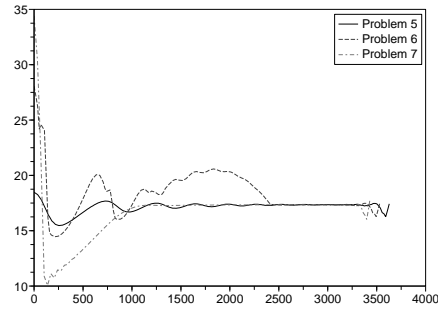


Figure 5.26: Ang. of at.(°)/T.(s)

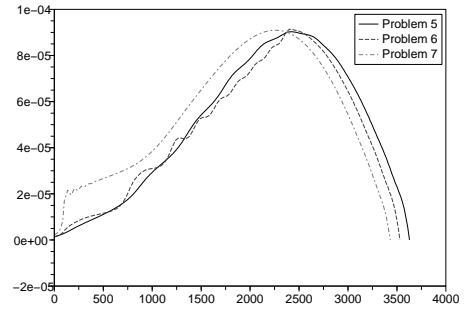


Figure 5.27: Bank ang.(°)/T.(s)

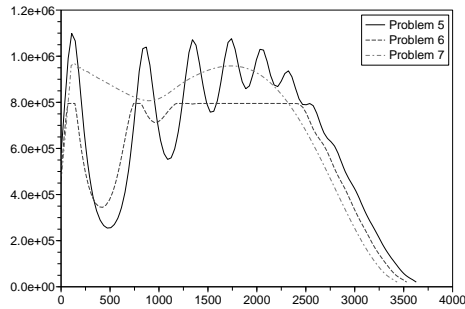


Figure 5.28: Heating (W/m²) / Time (s)

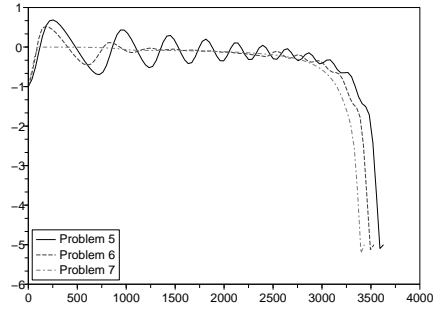


Figure 5.29: Fli. path ang.(°)/T.(s)

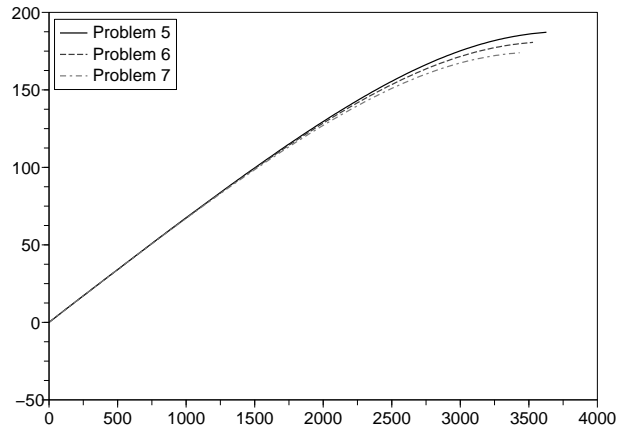


Figure 5.30: Long.(°)/T.(s)

Chapitre 6

Compléments numériques monoarc

Cette partie présente les résultats obtenus lors de l'utilisation de l'outil implémentant la méthode de point intérieur à deux autres cas de rentrée atmosphérique plus “réalistes” que les cas présentés précédemment dans ce chapitre. En effet, dans ces exemples l'atmosphère est interpolée sur des données physiques, les coefficients aérodynamiques sont tabulés en fonction de l'incidence et du nombre de Mach, ce nombre dépendant de la température et de la vitesse du véhicule. Ce travail a été effectué en collaboration avec David Castillo, stagiaire ONERA. Plus d'informations sur l'interpolation sont données dans [26].

6.1 Retour de booster Everest

Il s'agit du retour booster d'un concept TSTO. Cette trajectoire est intéressante, car le véhicule doit effectuer un demi-tour pour se rapprocher le plus du site de lancement, en l'occurrence Kourou. Ici, l'état est entièrement défini initialement, le temps final est libre, la masse est fixée, et nous comparons deux trajectoires, une avec contraintes de couloir d'incidence (en bleu), et l'autre sans cette contrainte (en rouge). Les Figures 6.1 à 6.6 représentent le comportement au cours du temps de certaines variables d'état, et la Figure 6.7 donne un aperçu de la trajectoire en 3 dimensions pour les cas avec et sans contrainte.

Les Figures 6.8 et 6.9 présentent l'évolution des commandes au cours du temps.

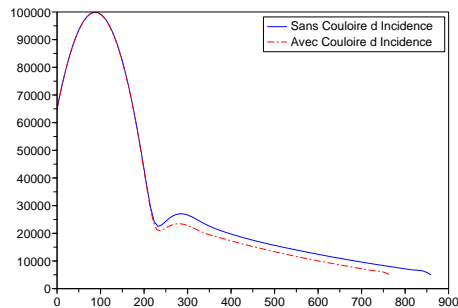


Figure 6.1: Altitude (m) / Temps (s)

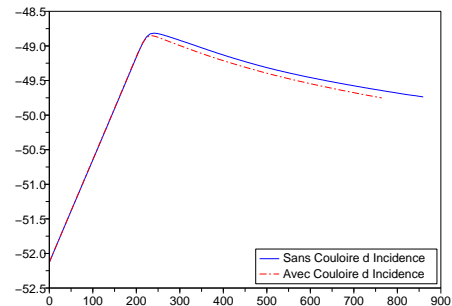


Figure 6.2: Longitude (°) / Temps (s)

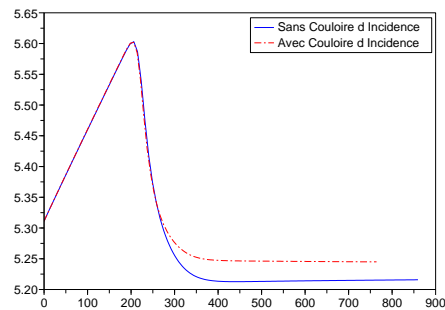


Figure 6.3: Latitude (°) / Temps (s)

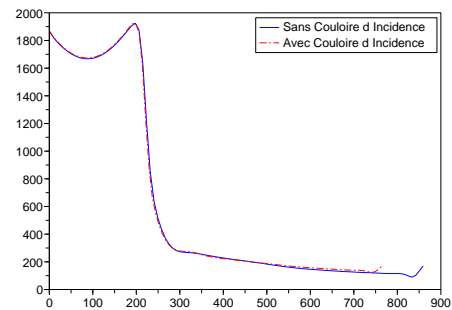


Figure 6.4: Vitesse (m/s) / Temps (s)

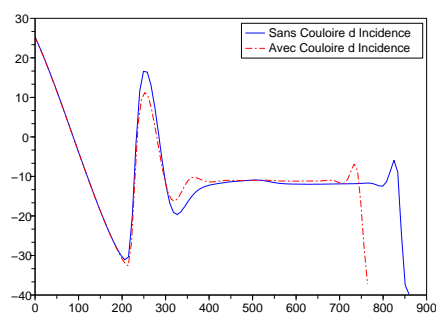


Figure 6.5: Pente (°) / Temps (s)

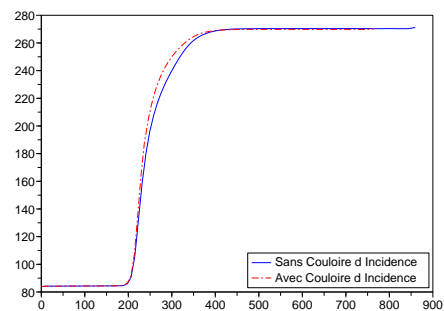


Figure 6.6: Azimut (°) / Temps (s)

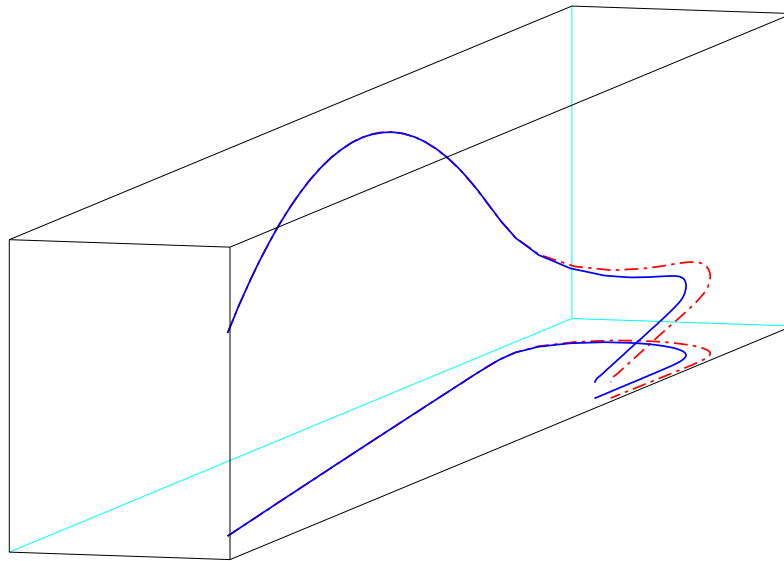


Figure 6.7: Visualisation 3D

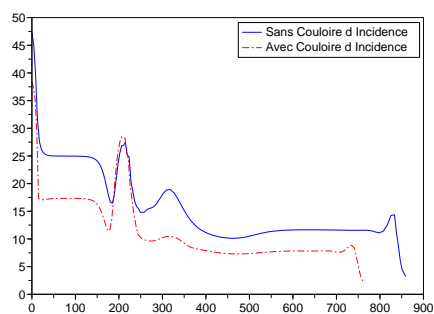


Figure 6.8: Incidence

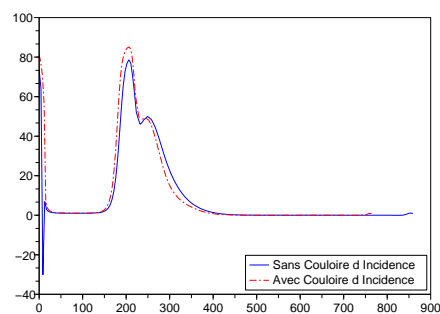


Figure 6.9: Gîte

6.2 Rentrée atmosphérique SOH

Il s'agit de la rentrée atmosphérique du véhicule SOH dans un cas dégradé, où le déploiement du deuxième étage n'a pu se faire. Ce cas est dimensionnant vis à vis du véhicule car celui-ci doit effectuer sa rentrée à "pleine charge" (la charge utile étant encore dans la soute). La trajectoire doit satisfaire une contrainte de non-rebond. En effet, les rebonds atmosphériques sont à éviter autant que possible, car ils sont difficilement contrôlables. Une variation infime au moment du rebond induit une erreur énorme sur le point de retombée suivant. De même que dans les cas précédents, deux courbes ont été représentées. La rouge représente la trajectoire sans la contrainte de non-rebond, et la bleue, celle obtenue après l'ajout de la contrainte.

Les Figures 6.10 à 6.15 représentent le comportement des variables du vecteur état pendant les deux trajectoires du cas SOH. La Figure 6.16 nous donne un aperçu tri-dimensionnel de la trajectoire, et les courbes des Figures 6.17 et 6.18 représentent les commandes en incidence et en gîte obtenues.

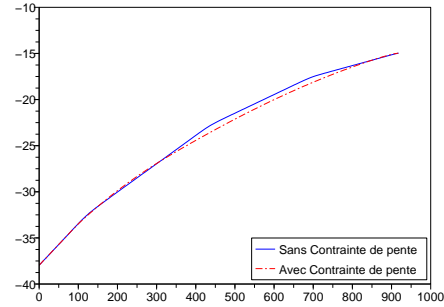
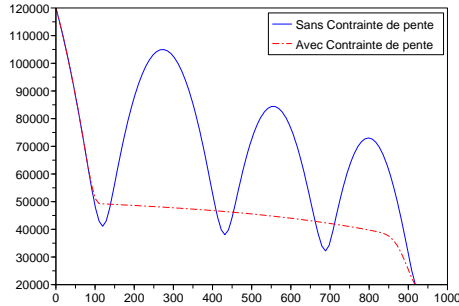


Figure 6.10: Altitude (m) / Temps (s) Figure 6.11: Longitude (°) / Temps (s)

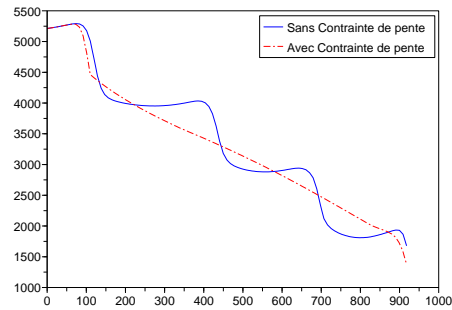
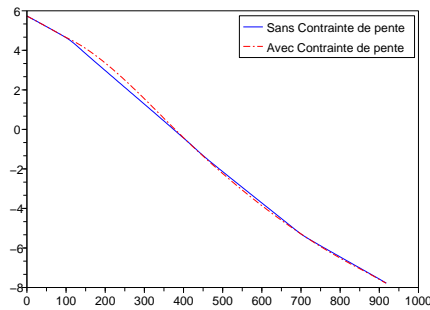


Figure 6.12: Latitude (°) / Temps (s) Figure 6.13: Vitesse (m/s) / Temps (s)

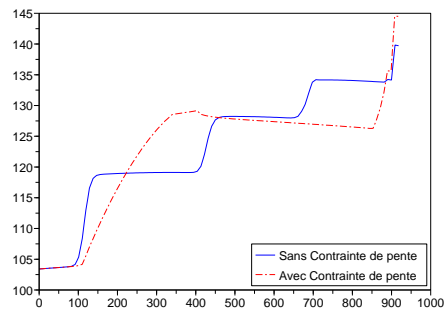
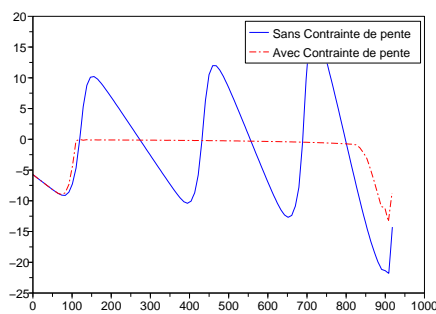


Figure 6.14: Pente (°) / Temps (s) Figure 6.15: Azimut (°) / Temps (s)

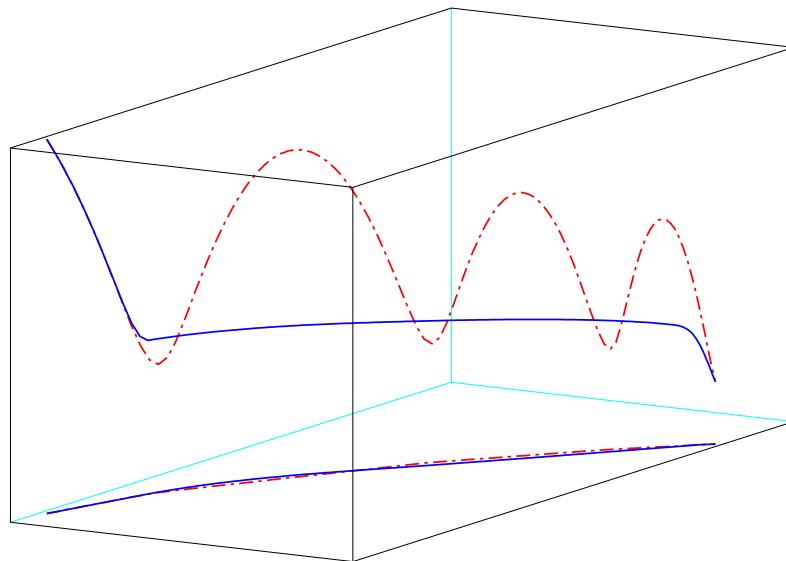


Figure 6.16: Trajectoire

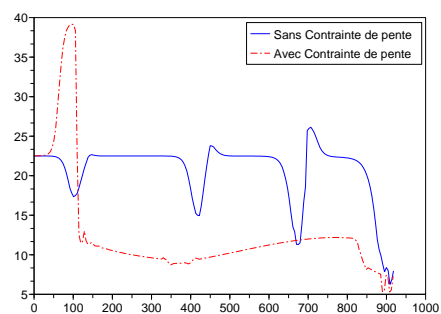


Figure 6.17: Incidence

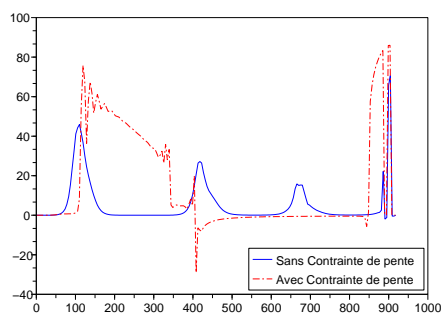


Figure 6.18: Gîte

Partie V

Calcul de trajectoire optimale multiarc

Chapitre 7

Optimisation multiarc

7.1 Graphe de scénario, dynamique et critère

7.1.1 Motivations

Remarque 32. La chronologie de mission d'un engin spatial peut inclure différents événements de nature à changer la dynamique du véhicule :

- largage de masse (coiffe, étage, masse utile)
- changement de régime moteur (allumage, extinction, modulation du débit)
- changement des caractéristiques aérodynamiques (déploiement d'un bouclier, largage de coiffe)
- assemblage avec un autre véhicule après rendez-vous (ce qui implique un changement de masse)

Remarque 33. Sur chaque arc du graphe, nous associons une équation différentielle ordinaire $\dot{y}_i = f_i(y_i, u_i)$ et en chaque sommet des relations de transmission. Il faut noter, de plus, que seuls des sommets de degré inférieur ou égal à trois seront étudiés dans ce chapitre. Cependant les manoeuvres de multi-séparation avec multi-rendez-vous ne conduisent qu'à une extension élémentaire des principes de ce chapitre. Ainsi, chaque sommet sera :

- soit un point de départ (lancement du véhicule - degré 1),
- soit un point d'arrivée (en général, atterrissage ou injection en orbite - degré 1),

- soit un changement d'arc (largage de masse telle qu'une coiffe ou d'un étage - degré 2),
- soit une séparation (booster-orbiteur des lanceurs futurs - degré 3),
- soit finalement un rendez-vous (orbiteur-station - degré 3).

7.1.2 Définitions préliminaires

Tel qu'il est défini dans le chapitre 4, nous rappelons qu'un *graphe orienté* est un couple d'ensemble (V, E) où V est l'ensemble des sommets et $E \subset V \times V$ l'ensemble des arcs.

Définition 34. Le *degré entrant* $d_e(x)$ d'un sommet x dans un graphe (V, E) est le nombre d'arcs entrant dans le sommet :

$$d_e(x) = \#\{(y, x) | y \in V \text{ et } (y, x) \in E\}$$

Le *degré sortant* $d_s(x)$ d'un sommet x dans un graphe (V, E) est le nombre d'arcs sortant dans le sommet :

$$d_s(x) = \#\{(x, y) | y \in V \text{ et } (x, y) \in E\}$$

Le *degré* $d(x)$ d'un sommet x dans un graphe (V, E) est la somme de ses degrés entrant et sortant :

$$d(x) = d_e(x) + d_s(x)$$

Définition 35. Un *graphe de scénario* sera un graphe orienté associé à un scénario de mission.

7.1.3 Variables du problème

Le graphe de scénario étant fixé, l'optimisation porte sur la commande de la dynamique sur les arcs et sur les paramètres non dynamiques (états initiaux, finaux et autres paramètres de conception ...). Ainsi à chaque arc $(i, j) \in E$ nous associons :

- un état $y_{ij}(t)$,
- une commande $u_{ij}(t)$,
- un vecteur de paramètre π_{ij} , influant sur la dynamique (en général il s'agit de masses et de dates de la chronologie de mission)

7.1.4 Conditions sur les arcs du graphe

Sur chaque arc, l'état du système sera soumis à une dynamique commandée ainsi qu'à des contraintes courantes.

Dynamique

Remarque 36. Sur chaque arc d'un graphe de scénario (V, E) , nous associons la dynamique du système correspondant, pour cela, il nous faut définir une fonction $\delta : E \rightarrow \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ qui, à chaque arc (i, j) , associe la dimension de y_{ij} , celle de u_{ij} et celle de π_{ij} .

$$\delta(i, j) = (\delta_y^{ij}, \delta_u^{ij}, \delta_\pi^{ij})$$

Définition 37. Étant donné un graphe de scénario (V, E) , la dynamique associée sera une application $f : E \rightarrow (\mathbb{R}^{\delta_y^{ij}} \times \mathbb{R}^{\delta_u^{ij}} \rightarrow \mathbb{R}^{\delta_\pi^{ij}})$ telle que :

- $f_{ij} : \mathbb{R}^{\delta_y^{ij}} \times \mathbb{R}^{\delta_u^{ij}} \rightarrow \mathbb{R}^{\delta_\pi^{ij}}$
- $\dot{y}_{ij} = f_{ij}(y_{ij}, u_{ij}, \pi_{ij}), \forall (i, j) \in E$

La condition de dynamique sera alors :

$$\forall (i, j) \in E, \dot{y}_{ij}(t) = f_{ij}(y_{ij}(t), u_{ij}(t), \pi_{ij}) \quad (7.1)$$

Contraintes courantes

Lors des différentes phases de vol, les engins aérospatiaux sont soumis à de nombreux phénomènes thermiques et mécaniques, ces véhicules ayant une structure d'une solidité bornée, les trajectoires que nous devons calculer doivent donc rester dans le domaine de résistance du véhicule. Nous aurons des contraintes courantes sur le système :

$$\forall (i, j) \in E, a_{ij} \leq g_{ij}(y_{ij}, u_{ij}, \pi_{ij}) \leq b_{ij} \quad (7.2)$$

Pour des raisons numériques nous introduisons les "variables d'écart". La condition (7.2) sera découpée en deux conditions :

$$\forall (i, j) \in E, a_{ij} \leq e_{ij} \leq b_{ij} \quad (7.3)$$

$$\forall (i, j) \in E, g_{ij}(y_{ij}, u_{ij}, \pi_{ij}) - e_{ij} = 0 \quad (7.4)$$

L'utilisation de ces variables d'écart, permet d'initialiser l'outil avec une trajectoire non-admissible, c'est à dire ne satisfaisant pas les contraintes courantes d'inégalité.

7.1.5 Conditions sur les sommets

Sur chaque sommet du graphe, nous associons des contraintes d'égalité et d'inégalité sur les variables d'état à l'instant correspondant au sommet $y_{ij}(t_i)$, ainsi que sur les paramètres statiques π_{ji} et de même pour $y_{ji}(t_i)$, π_{ji} .

Notation 38. Par la suite nous notons $E(i)$ l'ensemble des arcs débutant ou se terminant en i :

$$E(i) = (E \cap V \times \{i\}) \cup (E \cap \{i\} \times V)$$

Ainsi l'ensemble des variables concernées par les contraintes d'égalité au sommet i sera noté : $(y_e(t_i), \pi_e)_{e \in E(i)}$.

Conditions d'égalité

Les liens entre les différents états de chaque arc seront décrits par des conditions d'égalité sur chaque sommet. En ce qui concerne les sommets de degré 1, les conditions d'égalité seront les conditions initiales ou finales associées à ce sommet.

$$\forall i \in V, h_i((y_e(t_i), \pi_e)_{e \in E(i)}) = 0 \quad (7.5)$$

Conditions d'inégalité

Certains paramètres, états finaux ou initiaux peuvent accepter un certain degré de liberté dans un domaine que vont décrire ces conditions d'inégalité.

$$\forall v \in V, a_v \leq g_v((y_e(\tau_v^e), \pi_e)_{e \in E(v)}) \leq b_v \quad (7.6)$$

De même que pour les conditions d'inégalité courantes, nous introduisons des variables d'écart qui nous permettront une résolution plus efficace vis à vis de la méthode de point intérieur. Dans une première version du logiciel, ce type de contraintes ne sera pas pris en compte.

$$\forall v \in V, a_v \leq e_v \leq b_v \quad (7.7)$$

$$\forall i \in V, g_v((y_e(\tau_v^e), \pi_e)_{e \in E(v)}) - e_v = 0 \quad (7.8)$$

Nous avons choisi de noter τ_v^e l'instant initial ou final concerné sur ce sommet du graphe. C'est à dire que :

$$\tau_v^e = \begin{cases} t_e^d & \text{si } \exists w \in V, e = (v, w) \text{ et } e \in E \\ t_e^f & \text{si } \exists w \in V, e = (w, v) \text{ et } e \in E \end{cases} \quad (7.9)$$

7.1.6 Lien entre variables d'arc et de sommet

Aux sommets du graphe de scénario, des informations doivent être transmises de part et d'autre du sommet. Nous allons donc associer à chaque sommet, des *variables de jonction* $(J_i)_{i \in V}$ qui joueront le rôle de communication afin de bien séparer chaque arc.

Si nous notons $E(i)$ l'ensemble des arcs arrivant ou partant du sommet i , alors les conditions définissant les J_i seront :

$$J_i - (y_e(t_i), \pi_e)_{e \in E(i)} = 0 \quad (7.10)$$

Dans ces conditions, les conditions d'égalité aux sommets se réécrivent :

$$\forall v \in V, h_v(J_v) = 0 \quad (7.11)$$

7.1.7 Critère d'optimisation

Dans la modélisation actuelle, nous choisirons comme critère une somme de fonctions des variables associées à chaque sommet plus des critères intégraux sur les arcs :

$$\sum_{v \in V} \Psi_v(J_v) + \sum_{e \in E} \int_{t_e^d}^{t_e^f} \ell_e(y_e(t), u_e(t), \pi_e) dt \quad (7.12)$$

Remarque 39. Les critères concernant les paramètres et les états initiaux et finaux transiteront par les variables (J_i) .

Notation 40. L'état et le contrôle associé à l'arc $e = (i, j)$ seront notés y_e respectivement u_e et par conséquent l'adjoint p_e .

7.1.8 Problème de commande optimale sur un graphe de scénario

Définition 41. Un problème de contrôle optimal sur un graphe de scénario sera défini par un graphe de scénario (V, E) , une dynamique \mathcal{D} associée à ce graphe, des conditions de transition sur chaque sommet ainsi qu'un critère à optimiser. Ce problème est donc défini par l'ensemble de conditions (7.1) (7.10) (7.11) et le critère (7.12).

Définition 42. Un problème de commande optimale contraint sur un graphe de scénario sera un problème de commande optimale sur un graphe de scénario avec des contraintes supplémentaires associées aux sommets et aux arcs. Un tel problème est défini par un critère (7.12) ainsi que l'ensemble de conditions (7.1) (7.3) (7.4) (7.7) (7.8) (7.10) (7.11) et le critère (7.12).

7.2 Conditions d'optimalité

7.2.1 Cas sans contraintes d'inégalité

Dans un premier temps nous allons étudier le cas sans contraintes d'inégalité : Il s'agit alors d'optimiser les commandes ainsi que les paramètres associés aux sommets et ceux associés aux arcs. Pour cela, posons le lagrangien global :

$$\begin{aligned} \mathcal{L} = & \sum_{v \in V} [\Psi_v + \chi_v^\top h_v] + \sum_{e \in E} \left[\int_{t_e^d}^{t_e^f} \ell_e + [f_e - \dot{y}_e]^\top p_e dt \right] \\ & + \sum_{v \in V} [J_v - (y_e(\tau_v^e), \pi_e)_{e \in E(v)}]^\top \xi_v \end{aligned}$$

Dans un souci de lisibilité, nous avons choisi d'omettre les arguments des fonctions.

Notation 43. Sur chaque arc e de E , définissons H_e :

$$H_e(y, u, p) = \ell_e(y, u) + p^\top f_e(y, u)$$

ainsi que Φ_v sur chaque sommet $v \in V$:

$$\Phi_v = \Psi_v + \chi_v^\top h_v + [J_v - (y_e(t_v), \pi_e)_{e \in E(v)}]^\top \xi_v \quad (7.13)$$

Par l'intégration par parties sur chaque arc, nous obtenons :

$$\mathcal{L} = \sum_{v \in V} \Phi_v + \sum_{e \in E} \left\{ \int_{t_e^d}^{t_e^f} [H_e + \dot{p}_e y_e] dt + p_e(t_e^d)^\top y_e(t_e^d) - p_e(t_e^f)^\top y_e(t_e^f) \right\} \quad (7.14)$$

Conditions primales

- La dynamique sur chaque arc (7.1)
- les conditions d'égalité sur chaque sommet (7.5)
- La définition des variables de sommets (7.10)

Dynamique des états adjoints

Théorème 44. Sur l'arc e la dynamique de l'état adjoint p_e est :

$$\dot{p}_e(t) = -\frac{\partial H_e}{\partial y_e} \quad \forall e \in E \quad \forall t \in [t_e^d, t_e^f] \quad (7.15)$$

Démonstration. Conséquence de (7.14). ■

Conditions de transversalité

Théorème 45. Sur le sommet v les conditions de transversalité seront :

$$p_{ij}(t_i) = -\xi_i^{y_{ij}(t_i)} \quad (7.16)$$

$$p_{ji}(t_i) = +\xi_i^{y_{ji}(t_i)} \quad (7.17)$$

Conditions sur les paramètres

$$0 = \int_{t_i}^{t_j} \frac{\partial H_{ij}}{\partial \pi_{ij}} dt - \xi_i^{\pi_{ji}} - \xi_j^{\pi_{ji}} \quad (7.18)$$

Conditions sur la commande (Pontryaguine)

$$u_e(t) = \underset{w}{\operatorname{Argmin}} H_e(y_e(t), w, p_e(t)) \quad (7.19)$$

Conditions d'optimalité sur les sommets

$$\Psi'_v(J_v) + \chi_v^\top h'_v(J_v) + \xi_v = 0 \quad (7.20)$$

Remarque 46. Il est intéressant de noter que l'information liant les différents problèmes d'optimisation passe par les liens entre les états adjoints.

Les conditions d'optimalité du problème multiarc sans contraintes courantes seront donc les conditions : (7.1), (7.5), (7.10), (7.15), (7.16), (7.17), (7.18), (7.19) et (7.20).

7.2.2 Cas avec contraintes d'inégalité

Lorsque l'on ajoute des contraintes d'inégalité, on donne une formulation de type point intérieur par une perturbation des conditions de complémentarité. Ainsi, certaines contraintes du problème non contraint restent inchangées : (7.1), (7.5), (7.10) (7.16) et (7.17). alors que la dynamique des états adjoints ainsi que la condition d'optimalité de u sont modifiées en $(\forall (i, j) \in E \forall t \in [t_i, t_j])$:

$$\dot{p}_{ij} = -\frac{\partial H_{ij}}{\partial y_{ij}}(y_{ij}, u_{ij}, p_{ij}) + (\lambda + \mu)^\top \frac{\partial g_{ij}}{\partial y_{ij}}(y_{ij}, u_{ij}), \quad (7.21)$$

$$0 = \frac{\partial H_{ij}}{\partial u_{ij}}(y_{ij}, u_{ij}, p_{ij}) + (\lambda + \mu)^\top \frac{\partial g_{ij}}{\partial u_{ij}}(y_{ij}, u_{ij}). \quad (7.22)$$

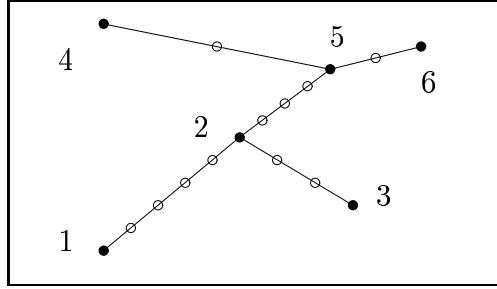


Figure 7.1: Exemple de graphe de scénario discrétisé

Dans les équations (7.21) et (7.22), les dépendances en temps ne sont pas explicitées car elles sont évidentes et alourdissent les formules. De plus, nous voyons apparaître de nouvelles conditions concernant les variables λ et μ :

$$0 = e_{ij}(t) - g_{ij}(y_{ij}(t), u_{ij}(t)) \quad \forall (i, j) \in E \quad \forall t \in [t_i, t_j] \quad (7.23)$$

$$0 = \varepsilon - \lambda_{ij}(t)(e_{ij}(t) + a) \quad \forall (i, j) \in E \quad \forall t \in [t_i, t_j] \quad (7.24)$$

$$0 = \varepsilon - \mu_{ij}(t)(e_{ij}(t) + b) \quad \forall (i, j) \in E \quad \forall t \in [t_i, t_j] \quad (7.25)$$

7.3 Résolution numérique du problème

Nous discrétisons les variables continues par un schéma de Runge-Kutta partitionné symplectique (voir chapitre 3).

7.3.1 Conditions d'optimalité discrètes

Du graphe de scénario au graphe liant les variables discrètes

Suite à la discrétisation, nous sommes passés d'un ensemble de problèmes de dimension infinie, à un ensemble de problèmes de dimension finie. Sur le graphe de scénario, cela peut s'interpréter comme un morcellement des arcs qui représentent la continuité des équations différentielles ordinaires. Après discrétisation, les arcs plus petits représentent le lien qui existe entre des variables. Pour résumer, dans le graphe de scénario, les sommets sont des liens entre des variables continues, et les arcs sont des équations différentielles alors que dans le graphe de discrétisation, les sommets sont des variables, et les arcs des équations liant ses variables.

Concernant notre exemple décrit figure B.1, après discrétisation, nous obtenons la figure 7.1.

Dans la figure 7.1 les cercles noirs représentent les variables de jonctions (qui elles sont inchangées même après discrétisation) et les cercles blancs,

les variables discrétisées. Nous nous trouvons alors avec un nouveau jeu de variables $((a_i)_{i \in V}, (\tilde{b}_{ij})_{(i,j) \in E})$ qui conserveront le même type de lien décrit dans la structure de matrice de la Figure B.2.

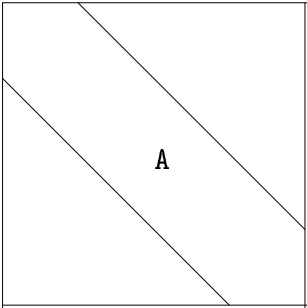
7.4 Algèbre linéaire

Une partie importante du succès de la méthode passe par la résolution efficace du système linéaire. Or les équations concernant les variables d'état discrétisées ne lient entre elles que les variables temporellement adjacentes, c'est à dire que y_k n'a un lien qu'avec y_{k+1} et y_{k-1} , ce qui résulte du choix d'une méthode à un pas. Ainsi avec un ordonnancement des équations et des variables adéquat, nous pourrions obtenir une structure bande. De plus, en ce qui concerne les variables d'écart, l'élimination sera simple car cette élimination revient à inverser la matrice identité. Finalement, il reste les paramètres qui, eux, sont liés à pratiquement toutes les équations. Il nous faut alors effectuer une réduction et utiliser une inversion numérique pleine.

Cette partie s'intéressera uniquement à l'aspect théorique de l'algèbre linéaire, les détails de programmation seront rassemblés dans l'annexe B.

7.4.1 Algèbre linéaire réursive

Sur chacun des arcs, nous avons une dynamique sur l'état à résoudre (**Dyn_y**), une sur l'état adjoint (**Dyn_p**), une condition sur la commande (**Dyn_u**), une condition sur les variables d'écart (**Dec**), une condition de complémentarité (**Lam_Mu**) ainsi que des conditions initiales et finales (**CI**) et (**CF**) à résoudre et des conditions sur les paramètres (**Cond_param**). Ainsi en regroupant les équations suivant un ordre particulier, nous ferons apparaître une structure bande dans la jacobienne :

	(y, u, p, λ, μ)	(e)	(y_p)
(CI)		B	D
(Dyn_y)			
(Dyn_p)			
(Lam_Mu)			
(Cond_u)			
(CF)			
(Dec)	C	I	
(Cond_param)	E		F

Ainsi, pour appliquer une résolution du type Newton, il nous faut une librairie d'algèbre linéaire gérant ce type de matrices de façon efficace. Par la suite, cette même librairie devra inverser des systèmes dont la structure est similaire à celle présentée à propos des graphes de scénarios; en effet, la structure du graphe de scénario se retrouve dans la jacobienne des conditions d'optimalité.

7.4.2 Type inductif de matrice

L'idée est d'utiliser le concept de *type inductif* défini par des *axiomes* et des *constructeurs* [51]. Ici, les axiomes seront les *matrices pleines*, *matrices bandes*, *matrices creuses*, *identités*, nuls et les constructeurs seront la construction quatre blocs (matrice construite par concaténation de quatre matrices) et la construction graphe (matrice associée à un graphe de scénario).

Par la suite, des fonctions seront définies sur cet ensemble inductif. Ces fonctions seront les fonctions de factorisation et d'inversion.

Remarque 47. En réalité la plupart des fonctions ne seront définies que sur un sous ensemble de l'ensemble décrit. En particulier, l'inversion de matrice creuse ne sera pas effectivement programmée car on pourra montrer que celle-ci ne serait pas utilisée dans le cadre de notre problème.

Notation 48. Choisissons de noter les ensembles suivants :

MP Ensemble des matrices pleines,

MB Ensemble des matrices bandes,

MC Ensemble des matrices creuses,

MI Ensemble des matrices identité,

MN Ensemble des matrices nulles,

Nous noterons notre ensemble de matrices \mathcal{M} .

Définition 49. L'ensemble des matrices considérées est défini comme suit :

- **Axiomes** $(MP \cup MB \cup MC \cup MI \cup MN) \subset \mathcal{M}$

- **Constructeur**

- $\forall A, B, C, D \in \mathcal{M}, \text{Blocs4}(A, B, C, D) \in \mathcal{M}$ où

$$\text{Blocs4}(A, B, C, D) = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right) \in \mathcal{M}$$

- $\forall g = (E, V), \forall (\alpha_e)_{e \in E}, (\beta_e^d, \beta_e^f)_{e \in E}, (\gamma_e^d, \gamma_e^f)_{e \in E}, (\delta_v)_{v \in V}$

$$\text{Graphe}(g, (\alpha_e), (\beta_e^d, \beta_e^f), (\gamma_e^d, \gamma_e^f), (\delta_v)) \in \mathcal{M}$$

Toutes les fonctions sur cet ensemble devront être définies sur l'ensemble des axiomes ainsi que sur le constructeur. D'un point de vue programmation, l'ensemble inductif sera un type récursif. Pour ne pas alourdir le formalisme de l'agencement des blocs de la construction "graphe", nous définissons celle-ci sur un exemple décrit en annexe pour le graphe de scénario Figure B.1 à la page 162. La Figure B.2 représente la structure de la construction associée à ce graphe.

7.4.3 Description détaillée

Matrices pleines

Les matrices pleines seront utilisées lorsque la matrice considérée a un taux de remplissage fort et n'a pas une structure particulière. Si la matrice est carrée, une fonction de factorisation QR est disponible. Pour définir les matrices pleines, nous avons besoin des :

- Dimensions : hauteur **n1**, longueur **n2**
- Valeurs numériques : un tableau **data** de taille **n1**×**n2**

Matrices bandes

Ces matrices carrées sont associées à un solveur QR bande. Pour définir les matrices bandes, nous avons besoin des :

- Dimensions : longueur du côté **n**, largeur de bande **p**
- Valeur numériques : un tableau **data** de taille $n+2 \times n \times p - p^2 - p$

Matrices creuses

Lorsque le taux de remplissage d'une matrice est faible, il est judicieux d'utiliser une structure adaptée de matrice creuse et ainsi des algorithmes de calcul adapté (produit scalaire, produit matrice vecteur ...). En ce qui nous concerne, nous avons choisi de représenter les matrices creuses avec les données suivantes :

- Dimensions : hauteur **n1**, longueur **n2**
- Nombre de valeurs non nulles : **nnzeros**
- Indices de ligne des valeurs non nulles : un tableau d'entier **i** de taille **nnzeros**
- Pointeur sur les fins de colonnes : tableau d'entier **j** de taille **n2**
- Données : un tableau de réels **data** de taille **nnzeros**

Exemple 50. Soit la matrice :

$$m = \begin{pmatrix} 0 & 2 & 0 & 0 \\ 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 2 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

On aura :

- **n1=5, n2=4, nnzeros=6,**
- **i=[2,1,4,3,4,5], j=[1,3,3,6] et data=[6,2,4,1,2,3]**

Remarque 51. Il est à noter que ces matrices ne sont pas munies de fonction de factorisation ni d'inversion.

Matrices identité

Ces matrices sont caractérisées par leur dimension.

- Dimension : longueur du côté n

Matrices nulles

- Dimensions : largeur $n1$, hauteur $n2$.

7.4.4 Factorisation et inversion des matrices axiomes et 4 blocs

Cette section étudie le problème de la factorisation et de l'inversion numérique de systèmes linéaires décrits par une matrice de l'ensemble \mathcal{M} .

Factorisation et l'inversion des matrices axiomes

Seules les matrices pleines et bandes seront effectivement inversées par une factorisation orthogonale QR.

Factorisation et l'inversion des matrices 4 blocs

Distinguons deux situations : la première, plus générale, sera celle où le bloc en bas à droite (nommée D) sera quelconque, et la deuxième sera celle où ce bloc sera la matrice identité. Dans le but d'améliorer au plus l'efficacité de la méthode, les résolutions devront être adaptées à chacune de ces situations. Nous noterons ici n la taille du côté de A , et p la taille du côté de D .

Première situation : Matrice D quelconque. Pour résoudre le système linéaire :

$$\begin{aligned} Ax + By &= b_x \\ Cx + Dy &= b_y \end{aligned}$$

Nous calculons successivement :

$$D_{red} = D - CA^{-1}B \quad (7.26)$$

$$y = D_{red}^{-1}(b_y - CA^{-1}b_x) \quad (7.27)$$

$$x = A^{-1}(b_x - By) \quad (7.28)$$

La première étape de calcul (7.26) demande donc une factorisation de A puis, p inversions avec cette factorisation. L'étape (7.27) ne demande qu'une seule

factorisation et résolution avec la matrice D_{red} . Finalement (7.28) demande une résolution de la matrice A , déjà factorisée.

En terme de calcul, nous aurons effectué 1 factorisation de A et $p + 1$ résolutions linéaires de taille n , ainsi qu'une factorisation et résolution de D .

Seconde situation (spécifique) : Nous remarquons que si D est la matrice identité, la méthode de résolution précédente n'est pas du tout adaptée. C'est pourquoi nous allons envisager une autre résolution dans ce cas là. Nous cherchons donc à résoudre le système linéaire :

$$\begin{aligned} Ax + By &= b_x \\ Cx + y &= b_y \end{aligned}$$

Les étapes de calcul sont les suivantes :

$$A_{red} = A - BC \quad (7.29)$$

$$x = A_{red}^{-1}(b_x - Bb_y) \quad (7.30)$$

$$y = (b_y - Cy) \quad (7.31)$$

Dans les cas que nous traitons, la matrice A est de type bande et il en est de même pour $A - BC$ car les variables éliminées sont des variables d'écart.

7.4.5 Factorisation et inversion des matrices "Graphe"

La figure B.2 montre la structure de la jacobienne des conditions d'optimalité. Pour résoudre ce système il faut choisir l'ordre d'élimination des différentes variables. Ici, deux choix s'offrent à nous :

- Nous éliminons les variables de sommet d'abord, puis celles d'arc ensuite,
- ou bien nous éliminons les variables d'arc d'abord, puis celles de sommet.

Pour comparer ces deux possibilités, nous allons admettre que les blocs associés aux arcs sont tous bande de largeur p et de longueur n , de plus, nous considérerons des dimensions de variables de sommet n_i , telles que :

$$n_i < n, \quad \forall i \in V$$

La complexité d'élimination des variables d'arc d'abord est au plus de :

$$\sum_{(i,j) \in E} [np^2 + (n_i + n_j)np] + \left(\sum_i n_i \right)^3$$

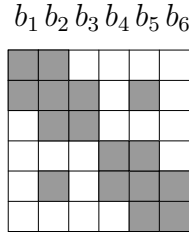


Figure 7.2: Structure de la jacobienne réduite

où $(\sum_i n_i)^3$ est l'inversion de la matrice réduite de petite dimension avec une factorisation QR pleine.

Ainsi la seconde solution apparaît comme la plus économique. Considérons donc la jacobienne réduite, obtenue en éliminant les variables d'arc. Les blocs non nuls de la jacobienne réduite correspondent aux sommets adjacents du graphe. La figure 7.2 représente la jacobienne réduite dans le cas de l'exemple B.1.

L'algorithme de résolution d'un système linéaire impliquant une construction "graphe" de la matrice sera donc la suivante. Étant donné un graphe $g = (E, V)$, et les matrices $(\alpha_e)_{e \in E}$, $(\beta_e^d, \beta_e^f)_{e \in E}$, $(\gamma_e^d, \gamma_e^f)_{e \in E}$, $(\delta_v)_{v \in V}$, le système que l'on cherche à résoudre est le suivant pour tout arc $e \in E$ et tout sommet $v \in V$:

$$\alpha_e x_e + \beta_e^d x_{d(e)} + \beta_e^f x_{f(e)} = b_e, \quad (7.32)$$

$$\delta_v x_v + \sum_{e \in E^d(v)} \gamma_e^d x_e + \sum_{e \in E^f(v)} \gamma_e^f x_e = b_v. \quad (7.33)$$

Les notations $d(e)$ et $f(e)$ désignent respectivement le sommet de départ de l'arc e et celui de fin. De plus les ensembles $E^d(v)$ et $E^f(v)$ représentent :

$$\begin{aligned} E^s(v) &= \{e \in E, v = s(e)\}, \\ E^f(v) &= \{e \in E, v = f(e)\}. \end{aligned}$$

Par élimination des variables d'arc x_e nous obtenons le système :

$$\begin{aligned} \delta_v x_v - \sum_{e \in E^d(v)} \gamma_e^d \alpha_e^{-1} (\beta_e^d x_{d(e)} + \beta_e^f x_{f(e)}) \\ - \sum_{e \in E^f(v)} \gamma_e^f (\beta_e^d x_{d(e)} + \beta_e^f x_{f(e)}) = b_v - \sum_{e \in E^d(v)} \gamma_e^d \alpha_e^{-1} b_e \\ - \sum_{e \in E^f(v)} \gamma_e^f \alpha_e^{-1} b_e \end{aligned} \quad (7.34)$$

Ce système “réduit” ne fait intervenir que des variables de sommets; nous le réécrivons comme :

$$\Delta_V X_V = B_V \quad (7.35)$$

Après avoir résolu ce système réduit comme nous allons le voir dans la section suivante, on obtient les variables d'arc par :

$$x_e = \alpha_e^{-1} (b_e - \beta_e^d x_{d(e)} - \beta_e^f x_{f(e)})$$

7.4.6 Résolution du système réduit

La résolution du système réduit peut être vu dualement comme une succession d'éliminations de sommets dans le graphe de scénario B.1 ou bien comme une élimination de blocs dans la matrice réduite. Si l'on associe un coût à chaque élimination, nous tombons dans le cadre de l'optimisation discrète, où il nous faut choisir la politique d'élimination des sommets d'un graphe. A l'élimination d'un sommet nous allons associer le coût ainsi qu'un graphe résiduel qu'il faudra à nouveau réduire.

Nous pouvons comparer deux éliminations différentes :

- Élimination d'une feuille
- Élimination d'un sommet interne (non-feuille)

Cependant, l'élimination d'un sommet interne crée des liens entre ses voisins, ce qui ne semble pas efficace en terme de calcul puisque l'on ajoute de nouvelles inversions à effectuer. Nous excluons ainsi cette méthode d'élimination et avons à choisir parmi tous les sommets feuilles. En effet cette méthode est optimale en termes de temps de calcul si tous les blocs sont de la même taille. La factorisation associée à une sommet i sera donc de l'ordre de n_i^3 et l'élimination de celui-ci aura la complexité :

$$\sum_{j \text{ voisin de } i} n_i^2 n_j$$

Donnons nous un graphe $g = (E, V)$ après élimination du sommet i le nouveau graphe associé au reste du système linéaire à résoudre sera $g' = (E', V')$ où :

- $V' = V \setminus \{i\}$
- $E' = \{(j, k) \in E \mid j \neq i \text{ et } k \neq i\} \cup \{(j, k) \in V^2 \mid (j, i) \in E \text{ et } (i, k) \in E\}$

Ainsi, étant donné un ordre sur l'ensemble des sommets V , nous pouvons calculer une complexité en conséquence. Notre but est donc de trouver cet ordre qui minimise la complexité.

Remarque 52. Nous conjecturons qu'il est optimal de procéder par élimination des sommets associés à un n_i minimal parmi les feuilles. L'algorithme correspondant est :

```

Solve( $g$ )
If  $g = (\emptyset, \{i_0\})$ 
Then Compute the resolution for  $i_0$ 
Else
   $I :=$  Set of leafs of  $g$ 
  Define  $i_r$  such that  $n_{i_r} = \min_{i \in I} n_i$ 
  Compute the resolution for  $i_r$ 
  Build  $g'$ 
  Solve( $g'$ )
End If

```

7.5 Applications

L'algorithme multiarc étant décrit, il faut maintenant le valider sur des cas d'école dans un premier temps, puis sur un cas réaliste dans un deuxième temps.

7.5.1 Cas académique

Description du problème

Le problème est décrit sur un graphe de scénario à trois arcs et quatre sommets définis par :

$$V = \{1, 2, 3, 4\} \text{ et } E = \{(1, 2), (2, 3), (2, 4)\}$$

$$\text{Min} \int_0^1 \sum_{e \in E} x_e(t) \frac{d}{dt}; \dot{x}_e(t) = u_e(t), \forall e \in E;$$

ε	Dogleg	CPU
0.5	7	3s
0.25	5	2s
0.125	4	2s
4.419e-2	5	2s
9.291e-3	5	1s
8.955e-4	6	2s
2.679e-5	4	1s
1.387e-7	2	1s
5.167e-11	1	1s

Table 7.1: Rapport d'exécution

Avec comme contraintes aux sommets :

$$x_{12}(0) = x_{23}(1) = x_{24}(1) = 0; x_{12}(1) + x_{23}(0) + x_{24}(0) = 1;$$

Et contrainte courante :

$$|u_e(t)| \leq 1, \forall e \in E$$

Résolution

Nous avons choisi une discrétisation de 100 points pour chaque arc, et nous faisons évoluer le paramètre de point intérieur à partir de 0.5. Le Tableau 7.1 décrit le nombre d'itérations Dogleg pour chaque valeur de ε .

Résultats

L'algorithme a fonctionné correctement sur ce problème simple. Les figures 7.3 et 7.4 présentent le comportement des variables d'état (x_{12}, x_{23}, x_{24}) pour les différentes valeurs de ε , ainsi que les commandes correspondantes.

7.5.2 Bi-étage en configuration siamoise

Ce cas inspiré du concept FSSC16 est le cas dont les arcs du graphe scénario sont arborescents. La Figure 7.5 représente le graphe de scénario. Les détails de ce modèle sont donnés dans le chapitre 2. L'exécution de l'algorithme a donné le Tableau 7.2.

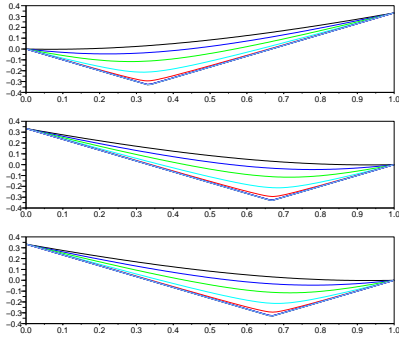
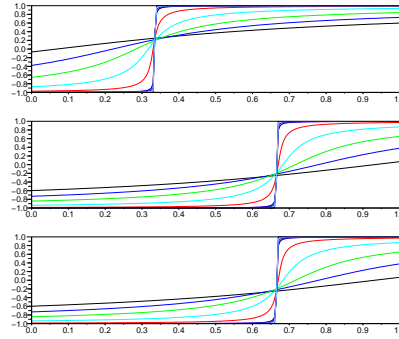
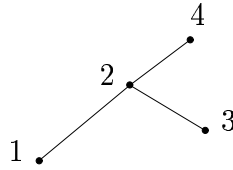
Figure 7.3: $(x_{12}(t), x_{23}(t), x_{24}(t))$ Figure 7.4: $(u_{12}(t), u_{23}(t), u_{24}(t))$ 

Figure 7.5: Graphe de scénario de la mission

ε	Iterations	temps CPU	$\ F(X)\ $
0.1	50	11 min 48 s	0.005481
0.05	50	12 min 23 s	0.0023319
0.03	50	16 min 28 s	7.91877e-05
0.02	50	11 min 21 s	0.000168043
0.01	50	14 min 33 s	0.000241436
0.001	24	06 min 00 s	3.60972e-11
0.0001	5	01 min 09 s	2.01731e-14
1e-05	2	00 min 28 s	6.6556e-12
5e-06	2	00 min 30 s	5.21109e-14
1e-06	2	00 min 27 s	1.11759e-12

Table 7.2: Rapport d'exécution

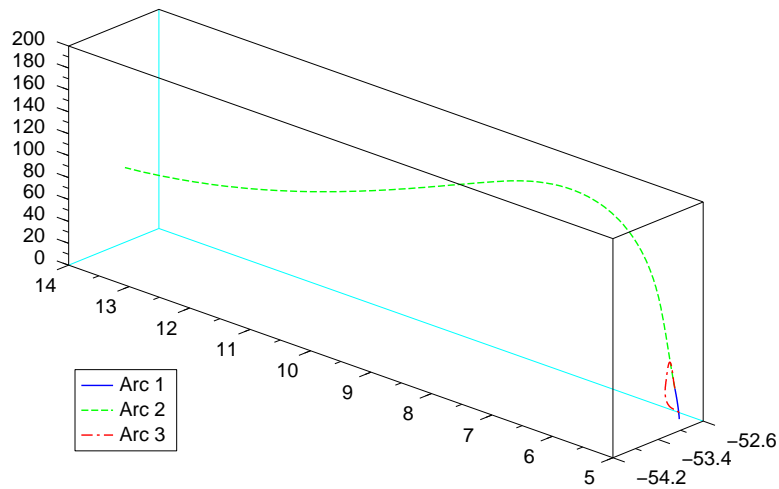


Figure 7.6: Vue 3D des 3 arcs

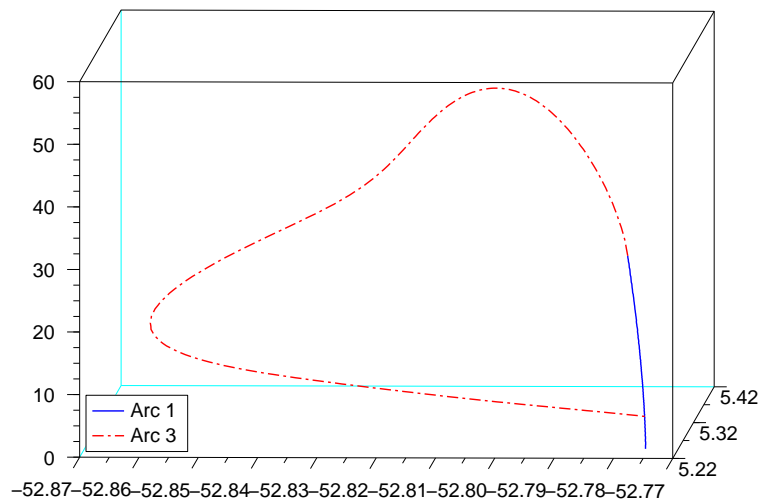


Figure 7.7: Vue 3D des arcs 1 et 3

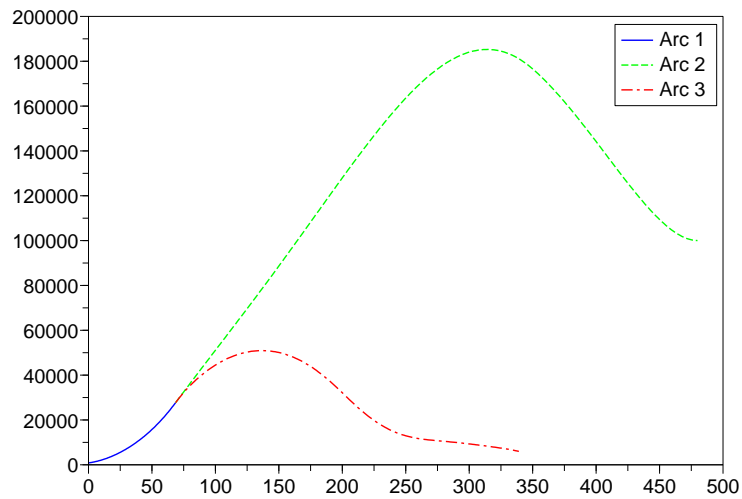


Figure 7.8: Altitude (m) / Temps (s)

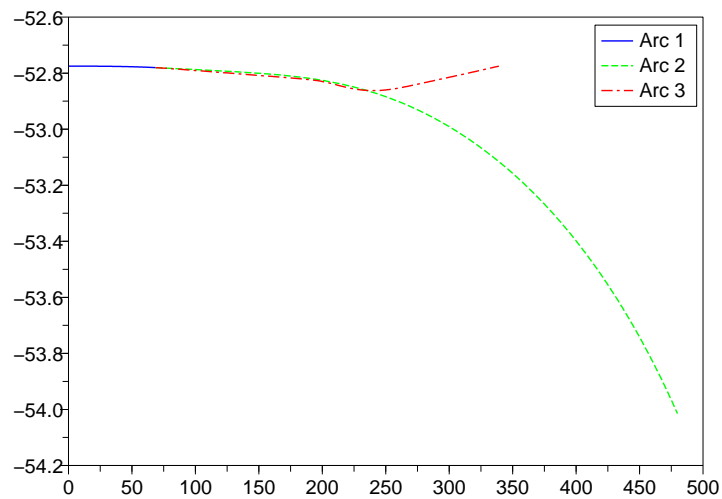


Figure 7.9: Longitude (°) / Temps (s)

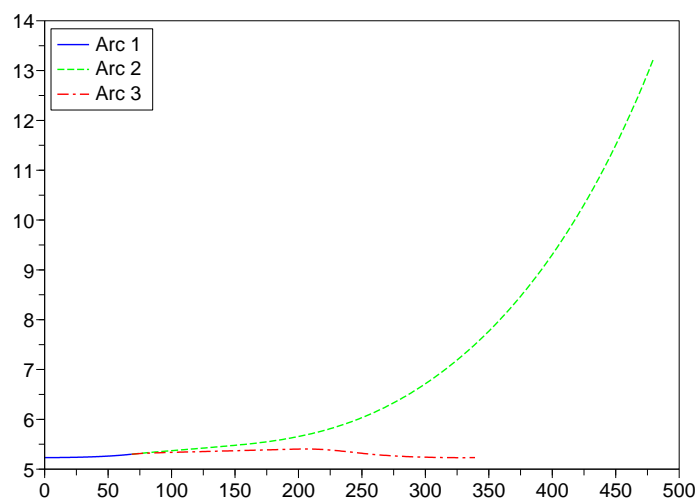


Figure 7.10: Latitude (°) / Temps (s)

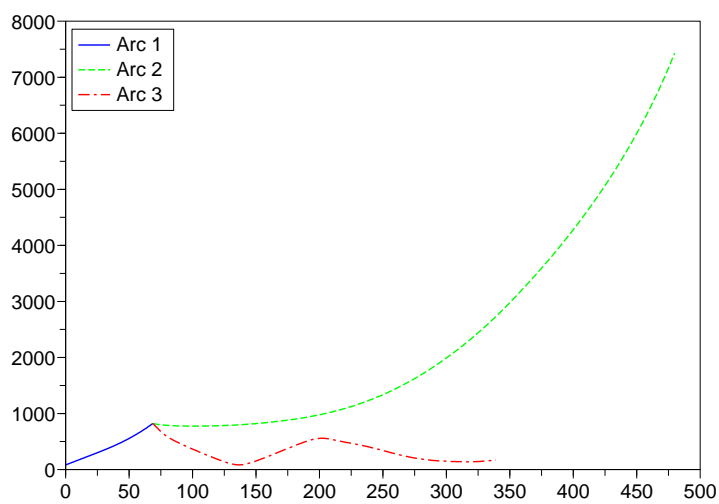
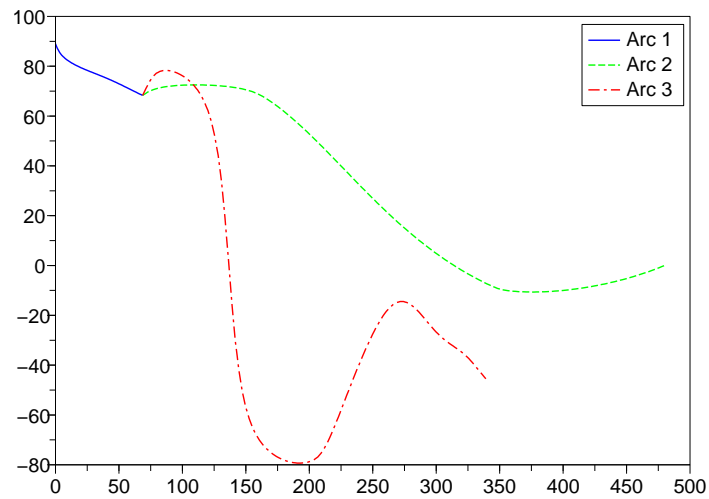
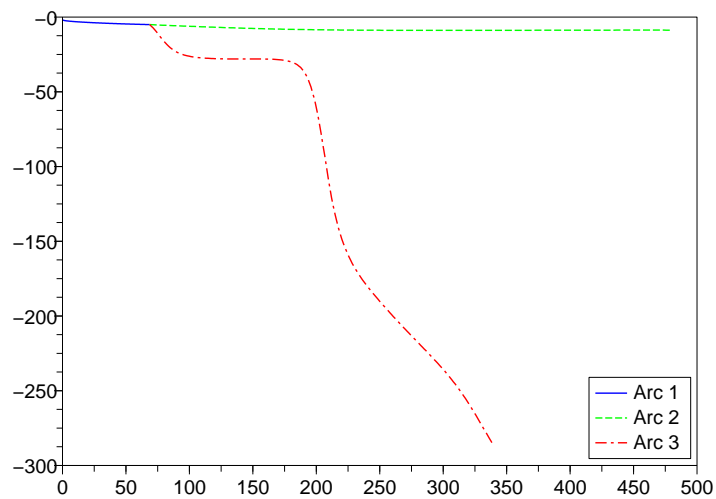


Figure 7.11: Vitesse (m/s) / Temps (s)

Figure 7.12: Pente ($^{\circ}$) / Temps (s)Figure 7.13: Azimut ($^{\circ}$) / Temps (s)

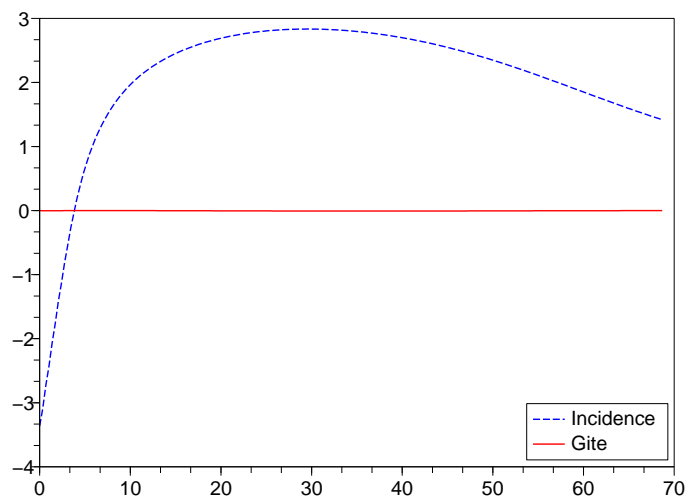


Figure 7.14: Commande - Arc 1 (°)

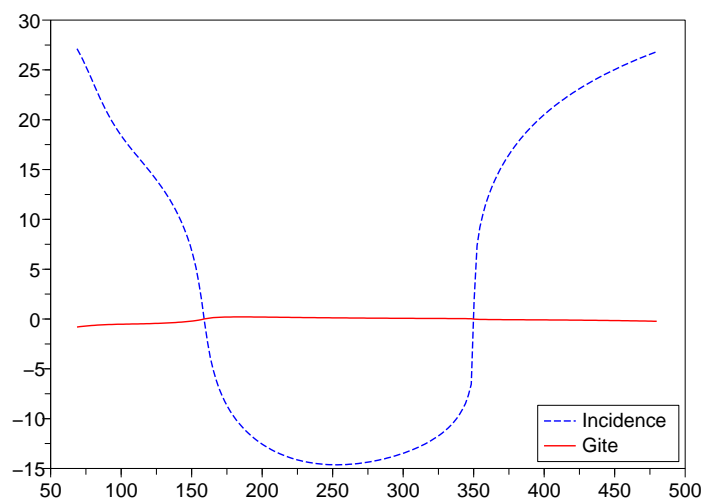


Figure 7.15: Commande - Arc 2 (°)

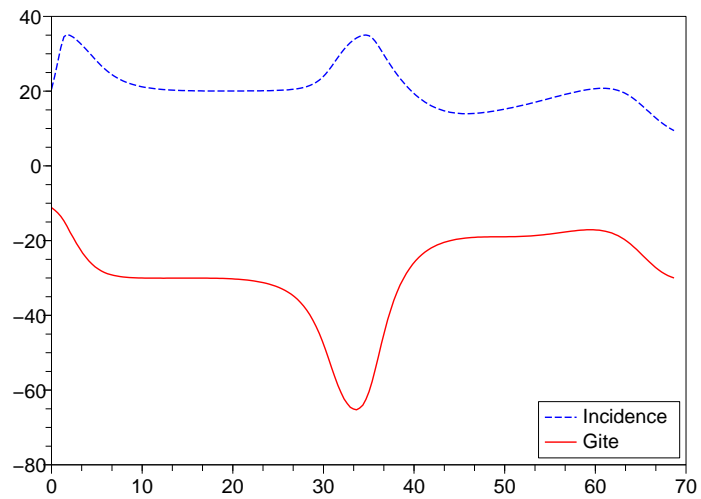


Figure 7.16: Commande - Arc 3 (°)

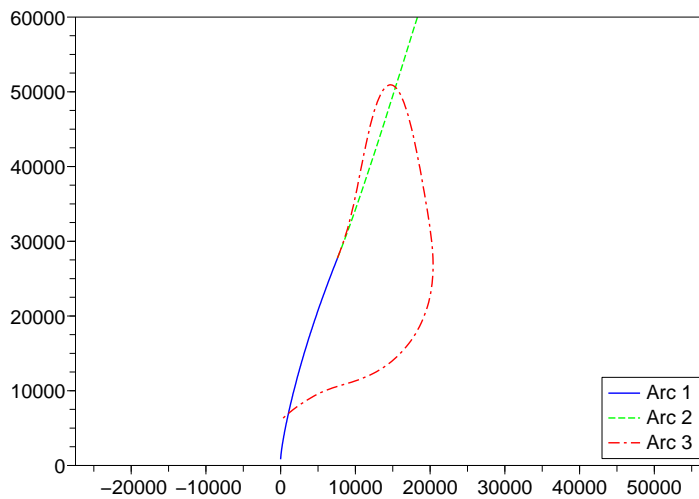


Figure 7.17: Altitude (m) / Distance (m)

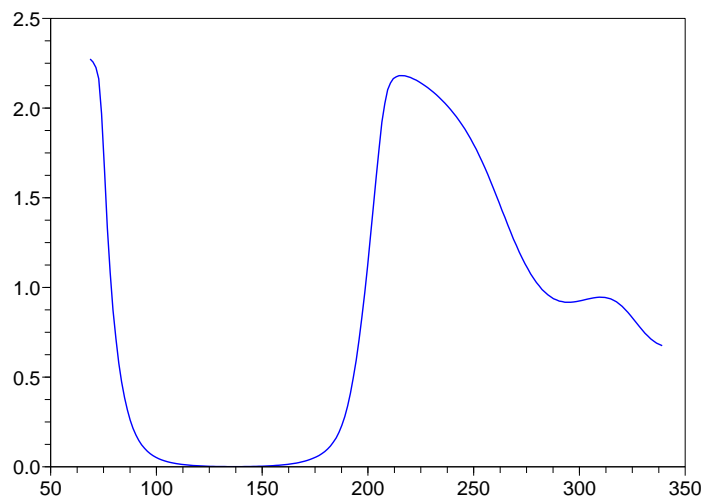


Figure 7.18: Facteur de charge - Arc 3 (g)

Annexe A

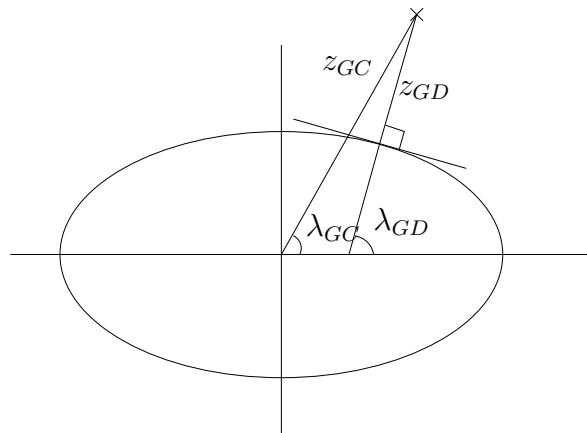
Détail de modélisation

Cette annexe contient un certain nombre de détails sur la modélisation.

A.1 L'atmosphère

A.1.1 Altitude *géocentrique*, *géodésique* et *géopotentielle*

L'altitude géocentrique d'une fusée est la distance au sol mesurée suivant la direction du centre de la Terre vers le point considéré. L'altitude géodésique est l'altitude mesurée suivant la verticale locale. Le schéma suivant représente la nuance qui existe entre ces deux altitudes z_{GD} et z_{GC} et les deux longitudes λ_{GD} et λ_{GC} . Dans le cas d'une Terre ronde, ces deux altitudes sont confondues.



Pratiquement, nous ne ferons pas de distinction entre les altitudes géocentrique et géodésique, que nous noterons z , mais par contre l'altitude géopotentielle H restera distinguée.

Cette dernière correspond à une altitude dans un monde fictif, où la force de pesanteur serait uniforme, c'est à dire qu'un corps de masse m sous une accélération de la pesanteur de g_0 passant d'une altitude 0 à une altitude H emmagasinerait une énergie potentielle de mg_0H , alors que, pour obtenir cette même énergie, il faudrait réellement monter à une altitude de z . Nous obtenons alors la relation :

$$\begin{aligned} mg_0H &= \int_0^z mg(h)dh \\ &= \int_0^z mg_0 \frac{R_e^2}{(h + R_e)^2} dh \\ H &= \frac{zR_e}{z + R_e} \end{aligned}$$

où z est l'altitude géocentrique, H l'altitude équipotentielle, R_e le rayon de la Terre et lorsque le J_2 est négligé.

A.1.2 Modélisation de l'*atmosphère*

Les calculs de trajectoire nécessitent la modélisation de l'atmosphère terrestre. Pour cela, il existe déjà de nombreux modèles utilisés dans beaucoup d'applications aéronautiques et aérospatiales.

Température

Dans certains modèles standards, la température est donnée en fonction de l'altitude géopotentielle selon des fonctions linéaires par morceaux. Pour le modèles US-76 cette information est donnée jusqu'à une altitude de 86 km.

Masse volumique

La masse volumique ρ se calcule en fonction de la température et de la pression selon la loi des gaz parfaits :

$$\rho = \frac{pM}{RT}$$

où R est la constante universelle des gaz et M la masse molaire de l'air. Nous pouvons aussi lier la variation de la masse volumique aux variations de T et p :

$$\rho = \rho_0 \frac{pT_0}{p_0T}$$

Pression atmosphérique

La pression est en fait calculée par intégration de l'équation de Laplace en altitude équipotentielle, c'est à dire comme si la gravité était constante.

Tranche d'atmosphère isotherme

Dans le cas d'une atmosphère isotherme, l'équation se formule et s'intègre comme suit :

$$\frac{dp}{p} = -\frac{g_0 M dH}{RT_0}; p(H) = p_0 e^{-\frac{Mg_0}{RT_0}(H-H_0)} \quad (\text{A.1})$$

Tranche d'atmosphère non-isotherme

Si, sur une tranche d'atmosphère, la température varie linéairement avec l'altitude géopotentielle, à une masse molaire de l'air constante, alors la pression se déduit du modèle de température en intégrant l'équation.

En effet, si $T = T_0 + k(H - H_0)$ avec $k \neq 0$, alors

$$\frac{dp}{p} = -\frac{g_0 M dH}{R(T_0 + kH)}; p(H) = p_0 \left(\frac{T_0 + k(H - H_0)}{T_0} \right)^{-\frac{Mg_0}{Rk}} \quad (\text{A.2})$$

Si, par contre, sur une tranche, $k = 0$, alors on retrouve le cas isotherme (A.1).

La vitesse du son

La notion de vitesse du son n'a de sens que pour des altitudes inférieures à 80-90 km. L'expression utilisée pour calculer la vitesse locale du son dépend de $\gamma = \frac{C_p}{C_v} = 1,4$, le rapport des chaleurs spécifiques de l'air (à pression constante et à volume constant) :

$$a = \sqrt{\frac{\gamma RT}{M}}$$

où M est la masse molaire moyenne de l'air.

Données du *standard US 76*

Constantes du modèle

- Constante des gaz parfaits : $R = 8,31432 [J/mol.K]$
- Rayon moyen de la Terre : $R_e = 6\,356\,766,0 [m]$

- Valeur de la gravité au niveau de la mer : $g_0 = 9,806\,65 \text{ [m/s}^2\text{]}$
- Masse molaire moyenne de l'air : $M = 28,9644 \cdot 10^{-3} \text{ [kg/mol]}$
- Rapport des chaleurs spécifiques : $\gamma = \frac{C_p}{C_v} = 1,4$

Le gradient de température

Altitude géocentrique Z	Altitude équipotentielle H	Gradient dT/dH
$[0, \cdot]$	$[0, 11]$	$-6,5$
$[\cdot, \cdot]$	$[11, 20]$	0
$[\cdot, \cdot]$	$[20, 32]$	$1,0$
$[\cdot, \cdot]$	$[32, 47]$	$2,8$
$[\cdot, \cdot]$	$[47, 51]$	0
$[\cdot, \cdot]$	$[51, 71]$	$-2,8$
$[\cdot, 86]$	$[71, 84.852]$	$-2,0$

Paramètres variables du modèle

- Pression de l'air au niveau de la mer : $P_0 = 101\,325 \text{ [Pa]}$ par défaut
- Température de l'air au niveau de la mer : $T_0 = 288,15 \text{ [K]} = 15[^\circ\text{C}]$

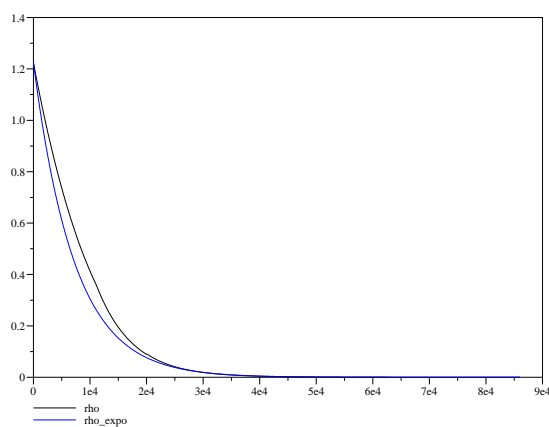


Figure A.1: Comparaison entre la modélisation exponentielle et US-76.

Annexe B

Détail du système linéaire

B.1 Résolution et algèbre-linéaire multi-arc

Comme nous avons pu le voir dans le chapitre 7, toutes les équations du problème multi-arc sont décrites.

Après avoir écrit toutes les conditions d'optimalité du problème multi-arc, il nous faut décrire l'algèbre linéaire sous-jacente à la résolution de ce type d'équations. En effet, les conditions distribuées dans le temps, du type de la dynamique sur l'état et l'état adjoint, se traduiront par un bloc bande. Par contre, les liens entre les arcs du graphe créeront des blocs non nuls dans la jacobienne et feront disparaître la propriété de matrice bande. Pour cette raison, il faut adapter la résolution du système linéaire afin d'être le plus efficace possible.

B.1.1 Détail des conditions d'optimalité discrétisées

Sur chaque arc

Dans ce groupe de conditions, nous omettrons volontairement l'exposant sur les variables décrivant leur appartenance à l'arc $e \in E$. Ce qui signifie que y_k doit être lu y_k^e .

Dynamique de l'état

$$\begin{cases} y_{k+1} = y_k + h_k \sum_{i=1}^s b_i f(u_{ki}, y_{ki}, y_p, t_{ki}) \\ y_{ki} = y_k + h_k \sum_{j=1}^s a_{ij} f(u_{kj}, y_{kj}, y_p, t_{kj}) \end{cases} \quad (Dyn_y)$$

avec $t_{ki} = t_k + c_i h_k$.

Nous noterons $(Dyn_y)_{k+1}^0$ pour $y_{k+1} = y_k + h_k \sum_{i=1}^s b_i f(u_{ki}, y_{ki}, y_p, t_{ki})$ et $(Dyn_y)_k^i$ pour $y_{ki} = y_k + h_k \sum_{j=1}^s a_{ij} f(u_{kj}, y_{kj}, y_p, t_{kj})$, k allant de 0 à $N-1$, et i de 1 à s .

Dynamique de l'état adjoint

$$\begin{cases} p_{k+1} = p_k + h_k \sum_{i=1}^s b_i \phi(u_{ki}, y_{ki}, y_p, p_{ki}, \lambda_{ki}, \mu_{ki}, t_{ki}) \\ p_{ki} = p_k + h_k \sum_{j=1}^s \hat{a}_{ij} \phi(u_{kj}, y_{kj}, y_p, p_{kj}, \lambda_{kj}, \mu_{kj}, t_{kj}) \end{cases} \quad (Dyn_p)$$

où :

$$\phi(u, y, y_p, p, \lambda, \mu, t) = -(\ell_y(u, y, y_p, t) + f_y(u, y, y_p, t)^T p + g_y(u, y, y_p, t)^T (\lambda + \mu))$$

Conditions sur la commande

$$\begin{cases} 0 = \psi(u_k, y_k, y_p, p_k, \lambda_k, \mu_k, t_k) \\ 0 = \psi(u_{ki}, y_{ki}, y_p, p_{ki}, \lambda_{ki}, \mu_{ki}, t_{ki}) \end{cases} \quad (Cond_u)$$

où :

$$\psi(u, y, y_p, p, \lambda, \mu, t) = \ell_u(u, y, y_p, t) + f_u(y, u, y_p, t)^T p + g_u(u, y, y_p, t)^T (\lambda + \mu)$$

Conditions sur les paramètres

$$\sum_{k=0}^N h_k \sum_{i=1}^s b_i \xi(u_{ki}, y_{ki}, y_p, p_{ki}, \lambda_{ki}, \mu_{ki}, t_{ki}) - \xi_{p,(v,w)}^v = 0 \quad (Cond_param)$$

où :

$$\xi(u, y, y_p, p, \lambda, \mu, t) = \ell_{y_p}(u, y, y_p, t) + f_{y_p}(y, u, y_p, t)^T p + g_{y_p}(u, y, y_p, t)^T (\lambda + \mu)$$

Conditions de complémentarité

$$\begin{cases} \varepsilon \mathbf{1} = \lambda_k (a - g(u_k, y_k, y_p, t_k)) \\ \varepsilon \mathbf{1} = \mu_k (b - g(u_k, y_k, y_p, t_k)) \\ \varepsilon \mathbf{1} = \lambda_{ki} (a - g(u_{ki}, y_{ki}, y_p, t_{ki})) \\ \varepsilon \mathbf{1} = \mu_{ki} (b - g(u_{ki}, y_{ki}, y_p, t_{ki})) \end{cases} \quad (Lam_Mu^1)$$

Avec $\lambda_k < 0$, $\lambda_{ki} < 0$ et $\mu_k > 0$, $\mu_{ki} > 0$

Variables d'écart Nous ajoutons donc pour chaque quantité $g(u_k, y_k, y_p, t_k)$ et $g(u_{ki}, y_{ki}, y_p, t_{ki})$ une variable de décalage (ou d'écart) e_k et e_{ki} liée à $g(u_k, y_k, y_p, t_k)$ et $g(u_{ki}, y_{ki}, y_p, t_{ki})$. Ainsi, il apparait un nouvel ensemble de conditions d'écart (Dec) et l'ensemble de conditions (Lam_Mu) se réécrit :

$$\begin{cases} 0 = g(u_k, y_k, y_p, t_k) + e_k \\ 0 = g(u_{ki}, y_{ki}, y_p, t_{ki}) + e_{ki} \end{cases} \quad (Dec)$$

$$\begin{cases} \varepsilon \mathbf{1} = \lambda_k(e_k + a) \\ \varepsilon \mathbf{1} = \mu_k(e_k + b) \\ \varepsilon \mathbf{1} = \lambda_{ki}(e_{ki} + a) \\ \varepsilon \mathbf{1} = \mu_{ki}(e_{ki} + b) \end{cases} \quad (Lam_Mu^2)$$

Conditions de transversalité

$$\begin{cases} p_0 + \xi_{y_0} = 0 \\ p_N - \xi_{y_N} = 0 \end{cases} \quad (Trans)$$

où κ vaut 1 si $e = (v, w)$ avec $w \in V$ et N si $e = (w, v)$ avec $w \in V$

Sur chaque sommet

Pour chaque sommet, nous avons un ensemble de conditions définissant la variable J_v associée au sommet $v \in V$, un ensemble de conditions d'égalité, et les conditions d'optimalité sur ce sommet.

Conditions de définition de J

$$J^v - (y_\kappa^e, y_p^e)_{e \in E(v)} \quad (Def_J)$$

où κ vaut 1 si $e = (v, w)$ avec $w \in V$ et N si $e = (w, v)$ avec $w \in V$

Conditions d'égalité

$$h_v(J_v) = 0 \quad (CEg)$$

Conditions d'optimalité

$$\Psi'_v(J_v) + \chi_v^\top h'_v(J_v) + \xi_v \quad (CO)$$

Il s'agit maintenant d'écrire la jacobienne du système tout en assurant son "inversibilité" si le problème est bien posé. Pour cela, il suffit que les blocs associés à chaque arc soient tous la linéarisation d'un problème de commande optimale lui-même bien posé. Pour cela, nous allons regrouper nos équations

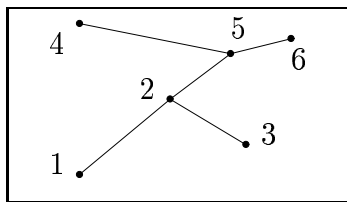


Figure B.1: Exemple de graphe de scénario

de manière à ce que chaque bloc associé à un arc soit la linéarisation d'un problème où l'état initial est fixé et l'état final libre, ce qui signifie que les équations de définition de J (Def_J) ont été éclatées avec les conditions de transversalité (Trans).

B.1.2 Jacobienne du système

Quel que soit le type de problème considéré, avec ou sans contraintes d'inégalité, la jacobienne du système conservera un aspect identique.

Notation 53. Nous allons noter $(a_{ij})_{(i,j) \in E}$ les variables associées aux arcs du graphe $((y_{ij}(t), u_{ij}(t), p_{ij}(t), \pi_{ij}, \lambda_{ij}(t), \mu_{ij}(t), e_{ij}(t))_{t \in]t_i, t_j[})$ et $(b_v)_{v \in V}$ les variables associées aux sommets du graphe (J_v, ξ_v, χ_v) .

Remarque 54. Ici, nous manipulons des variables de dimension infinie, par exemple l'état y_{ij} est une fonction du temps, la jacobienne décrite ci-dessous ne représente que les liens linéaires entre les variables finies ou infinies.

Avec ces notations, nous pouvons écrire le lagrangien sous la forme :

$$\mathcal{L} = \sum_{i \in V} L_i(b_i) + \sum_{(i,j) \in E} L_{ij}(a_{ij}, b_i, b_j)$$

Prenons comme exemple le graphe de scénario suivant $V = \{1, 2, 3, 4, 5, 6\}$ et $E = \{(1, 2), (2, 3), (2, 5), (4, 5), (5, 6)\}$, le dessin de ce graphe étant la figure B.1.

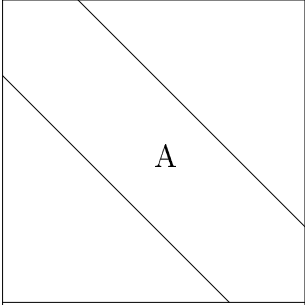
La jacobienne des conditions d'optimalité sera alors la figure B.2.

a_{12}	a_{23}	a_{25}	a_{45}	a_{56}	b_1	b_2	b_3	b_4	b_5	b_6
α_{12}					β_{12}^d	β_{12}^f				
	α_{23}				β_{23}^d	β_{23}^f				
		α_{25}			β_{25}^d			β_{25}^f		
			α_{45}					β_{45}^d	β_{45}^f	
				α_{56}					β_{56}^d	β_{56}^f
γ_{12}^d					δ_1					
γ_{12}^f	γ_{23}^d	γ_{25}^d				δ_2				
	γ_{23}^f						δ_3			
			γ_{45}^d					δ_4		
		γ_{25}^f	γ_{45}^f	γ_{56}^d					δ_5	
				γ_{56}^f						δ_6

Figure B.2: Structure de la jacobienne

B.1.3 Détails de la jacobienne du système

Pour chaque arc (Matrice de type α)

	(y, u, p, λ, μ)	(e)	(y_p)
(Trans/Def_J)		B	D
(Dyn_p)			
(Dyn_y)			
(Lam_Mu ²)			
(Cond_u)			
(Dec)	C	I	
(Cond_param)	E		F

Description de A

- Type : Matrice bande
- Dimensions : $n = (2n_y + n_u + 2n_g) * (n * (s + 1) + 1)$, $p = (s + 1) * (2n_y + n_u + 2n_g) + n_y - 1$

Nous allons maintenant expliciter les différents sous-blocs de cette matrice :
Les tableaux B.1 et B.2 numérotent les blocs de la matrice bande A.

Table B.3: Les sous-blocs de A

N^o	Formule
1	$\partial[F_1(X)]/\partial y_0$ 1 sur I_0
2	$\partial[F_1(X)]/\partial p_0$ 1 sur I_0
3	$\partial[F_2(X)]/\partial y_0$ $(-I, -I, \dots, -I)^T$
4	$\partial[F_2(X)]/\partial y_{0l}$ $\begin{cases} I - h_0 a_{ii} f_y(u_{0i}, y_{0i}) & \text{si } i = l \\ -h_0 a_{il} f_y(u_{0l}, y_{0l}) & \text{sinon} \end{cases}$
5	$\partial[F_2(X)]/\partial u_{0l}$ $-h_0 a_{il} f_u(u_{0l}, y_{0l})$
6	$\partial[F_3(X)]/\partial \lambda_0$ $-\text{Diag}(e_0 + a)$
7	$\partial[F_4(X)]/\partial \lambda_{0l}$ $-\text{Diag}(e_{0l} + a)$ si $i = l$
8	$\partial[F_5(X)]/\partial \mu_0$ $-\text{Diag}(e_0 + b)$

Table B.3: Les sous-blocs de A

N^o	Formule
9	$\partial[F_6(X)]/\partial\mu_{0l} \quad -\text{Diag}(e_{0l} + b) \text{ si } i = l$
10	$\partial[F_7(X)]/\partial y_{0l} \quad h_0 \hat{a}_{il} (f_{yy}(u_{0l}, y_{0l})^\top p_{0l} + l_{yy}(u_{0l}, y_{0l}) + g_{yy}(u_{0l}, y_{0l})^\top (\lambda_{0l} + \mu_{0l}))$
11	$\partial[F_7(X)]/\partial \lambda_{0l} \quad h_0 \hat{a}_{il} g_y(u_{0l}, y_{0l})^\top$
12	$\partial[F_7(X)]/\partial \mu_{0l} \quad h_0 \hat{a}_{il} g_y(u_{0l}, y_{0l})^\top$
13	$\partial[F_7(X)]/\partial p_{0l} \quad \begin{cases} I + h_0 \hat{a}_{ii} f_y(u_{0i}, y_{0i})^\top & \text{si } i = l \\ h_0 \hat{a}_{il} f_y(u_{0l}, y_{0l})^\top & \text{sinon} \end{cases}$
14	$\partial[F_7(X)]/\partial p_0 \quad (-I, -I, \dots, -I)^\top$
15	$\partial[F_7(X)]/\partial u_0 \quad h_0 \hat{a}_{il} (f_{yu}(u_{0l}, y_{0l})^\top p_{0l} + l_{yu}(u_{0l}, y_{0l}) + g_{yu}(u_{0l}, y_{0l})^\top (\lambda_{0l} + \mu_{0l}))$
16	$\partial[F_8(X)]/\partial y_{0l} \quad h_0 b_l (f_{yy}(u_{0l}, y_{0l})^\top p_{0l} + l_{yy}(u_{0l}, y_{0l}) + g_{yy}(u_{0l}, y_{0l})^\top (\lambda_{0l} + \mu_{0l}))$
17	$\partial[F_8(X)]/\partial \lambda_{0l} \quad h_0 b_l g_y(u_{0l}, y_{0l})^\top$
18	$\partial[F_8(X)]/\partial \mu_{0l} \quad h_0 b_l g_y(u_{0l}, y_{0l})^\top$
19	$\partial[F_8(X)]/\partial p_{0l} \quad h_0 b_l f_y(u_{0l}, y_{0l})^\top$
20	$\partial[F_8(X)]/\partial p_0 \quad -I$
21	$\partial[F_8(X)]/\partial u_{0l} \quad h_0 b_l (f_{yu}(u_{0l}, y_{0l})^\top p_{0l} + l_{yu}(u_{0l}, y_{0l}) + g_{yu}(u_{0l}, y_{0l})^\top (\lambda_{0l} + \mu_{0l}))$
22	$\partial[F_8(X)]/\partial p_1 \quad I$
23	$\partial[F_9(X)]/\partial y_{0l} \quad \begin{matrix} f_{uy}(y_{0l}, u_{0l})^\top p_{0l} + \ell_{uy}(u_{0l}, y_{0l}) \\ + g_{uy}(y_{0l}, u_{0l})^\top (\lambda_{0l} + \mu_{0l}) \end{matrix} \text{ si } i = l$
24	$\partial[F_9(X)]/\partial \lambda_{0l} \quad g_u(y_{0l}, u_{0l})^\top \text{ si } i = l$
25	$\partial[F_9(X)]/\partial \mu_{0l} \quad g_u(y_{0l}, u_{0l})^\top \text{ si } i = l$
26	$\partial[F_9(X)]/\partial p_{0l} \quad f_u(y_{0l}, u_{0l})^\top \text{ si } i = l$
27	$\partial[F_9(X)]/\partial u_{0l} \quad \begin{matrix} f_{uu}(y_{0l}, u_{0l})^\top p_{0l} + \ell_{uu}(u_{0l}, y_{0l}) \\ + g_{uu}(y_{0l}, u_{0l})^\top (\lambda_{0l} + \mu_{0l}) \end{matrix} \text{ si } i = l$
28	$\partial[F_{10}(X)]/\partial y_0 \quad \begin{matrix} f_{uy}(y_0, u_0)^\top p_0 + \ell_{uy}(u_0, y_0) \\ + g_{uy}(y_0, u_0)^\top (\lambda_0 + \mu_0) \end{matrix}$
29	$\partial[F_{10}(X)]/\partial \lambda_0 \quad g_u(y_0, u_0)^\top$
30	$\partial[F_{10}(X)]/\partial \mu_0 \quad g_u(y_0, u_0)^\top$
31	$\partial[F_{10}(X)]/\partial p_0 \quad f_u(y_0, u_0)^\top$
32	$\partial[F_{10}(X)]/\partial u_0 \quad \begin{matrix} f_{uu}(y_0, u_0)^\top p_0 + \ell_{uu}(u_0, y_0) \\ + g_{uu}(y_0, u_0)^\top (\lambda_0 + \mu_0) \end{matrix}$
33	$\partial[F_{11}(X)]/\partial y_0 \quad -I$
34	$\partial[F_{11}(X)]/\partial y_{0l} \quad -h_0 b_l f_y(u_{0l}, y_{0l})$
35	$\partial[F_{11}(X)]/\partial u_{0l} \quad -h_0 b_l f_u(u_{0l}, y_{0l})$
36	$\partial[F_{10N-2}(X)]/\partial p_N \quad I$

Table B.3: Les sous-blocs de A

N^o	Formule
37	$\partial[F_{10N+1}(X)]/\partial y_N$ I
38	$\partial[F_{10N+2}(X)]/\partial \lambda_N$ $-\text{Diag}(e_N + a)$
39	$\partial[F_{10N+3}(X)]/\partial \mu_N$ $-\text{Diag}(e_N + b)$
40	$\partial[F_{10N+4}(X)]/\partial y_N$ $1 \text{ sur } I_T$
41	$\partial[F_{10N+4}(X)]/\partial p_N$ $1 \text{ sur } \bar{I}_T$
42	$\partial[F_{10N+5}(X)]/\partial y_N$ $f_{uy}(y_N, u_N)^\top p_N + \ell_{uy}(u_N, y_N)$ $+ g_{uy}(y_N, u_N)^\top (\lambda_N + \mu_N)$
43	$\partial[F_{10N+5}(X)]/\partial p_1$ $g_u(y_N, u_N)^\top$
44	$\partial[F_{10N+5}(X)]/\partial p_N$ $g_u(y_N, u_N)^\top$
45	$\partial[F_{10N+5}(X)]/\partial y_N$ $f_u(y_N, u_N)^\top$
46	$\partial[F_{10N+5}(X)]/\partial y_N$ $f_{uu}(y_N, u_N)^\top p_N + \ell_{uu}(u_N, y_N)$ $+ g_{uu}(y_N, u_N)^\top (\lambda_N + \mu_N)$

La table B.3 détaille les sous blocs de A.

Description de B

- Type : Matrice creuse
- Dimensions : $n_1 = (2n_y + n_u + 2n_g)(n(s+1) + 1)$, $n_2 = n_g(n(s+1) + 1)$

Cette matrice est composée de sous blocs diagonaux $\text{Diag}(\lambda_i)$ et $\text{Diag}(\mu_i)$. Il y a donc $2n(s+1) + 1$ blocs qui sont de la forme représentée dans les Figures B.3 et B.4

Description de C

- Type : Matrice creuse
- Dimensions : $n_1 = n_g(n(s+1) + 1)$, $n_2 = (2n_y + n_u + 2n_g)(n(s+1) + 1)$

Cette matrice est composée de sous blocs étant des g_y et g_u . Sa structure est représentée dans les Figures B.5 et B.6

Description de I

- Type : Matrice identité
- Dimensions : $n_g(n(s+1) + 1)$

1 3 4			2	5					
	6 7								
		8 9							
10 16	11 17	12 18	13 14 19 20	15 21				22	
23 28	24 29	25 30	26 31	27 32					
33 34				35

Table B.1: Numérotation des blocs en haut à gauche

⋮	⋮	⋮	⋮	⋮
			36	
37				
	38			
		39		
40			41	
42	43	44	45	46

Table B.2: Numérotation des blocs en bas à droite

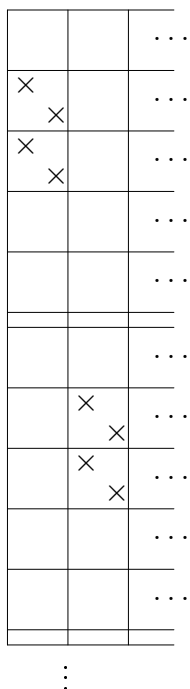


Figure B.3: Premières cases de B



Figure B.4: Dernières cases de B

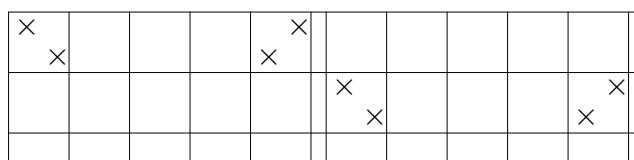


Figure B.5: Premières cases de C

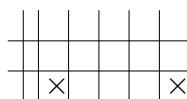


Figure B.6: Dernières cases de C

	J	χ	ξ
(Def_J/Trans)	A		B
(CO)	C	D	E
(CEg)	F		

Figure B.7: Structure des blocs de type δ **Description de D**

- Type : Matrice pleine
- Dimensions : $n_1 = (2n_y + n_u + 3n_g)(n(s+1) + 1)$, $n_2 = n_p(n(s+1) + 1)$

Description de E

- Type : Matrice pleine
- Dimensions : $n_1 = n_p(n(s+1) + 1)$, $n_2 = (2n_y + n_u + 3n_g)(n(s+1) + 1)$

Description de F

- Type : Matrice pleine
- Dimensions : $n_1 = n_p(n(s+1) + 1)$, $n_2 = n_p(n(s+1) + 1)$

Pour chaque sommet (Matrice de type δ)

Pour chaque sommet de la matrice, correspond un bloc de type δ . La Figure B.7 représente le découpage en sous blocs de cette partie de la jacobienne.

Les blocs A et B des matrices de type δ seront des morceaux de matrices identité selon que la linéarisation de la condition concernée est (Def_J) ou (Trans).

Le bloc C sera $\Psi''(J_v) + \chi_v^\top h_v''(J_v)$, D sera $h_v'(J_v)^\top$ et E sera l'identité.

Le bloc F lui sera simplement $h_v'(J_v)$.

Les blocs de type β et γ sont des matrices très creuses contenant des valeurs non nulles provenant de la linéarisation de (Def_J) et (Trans).

Annexe C

Participation à des conférences et publications

Articles acceptés ou soumis :

- Bonnans Frédéric et Laurent-Varin Julien - *Computation of order conditions for symplectic partitioned Runge-Kutta schemes with application to optimal control*, Numerische Mathematik, à paraître. Révision du Rapport de Recherche INRIA 5398 (Décembre 2004).
- Bérend Nicolas, Bonnans Frédéric, Haddou Mounir, Laurent-Varin Julien et Talbot Christophe *An Interior-Point Approach to Trajectory Optimization*, soumis à Journal of Guidance, Control, and Dynamics, version révisée (octobre 2005). Révision du Rapport de Recherche INRIA 5613 (Juin 2005).

Congrès avec publication :

- Bérend Nicolas, Bonnans Frédéric, Haddou Mounir, Laurent-Varin Julien et Talbot Christophe, *A Preliminary Interior Point Algorithm For Solving Optimal Control Problems*, Proc. Fifth Int. Conf. on Launcher Technology, Madrid, November 25-27, 2003. Aussi Rapport ONERA TP 2003-163.
- Bérend Nicolas, Bonnans Frédéric, Haddou Mounir, Laurent-Varin Julien et Talbot Christophe, *On the Refinement of Discretization for Optimal Control Problems*, Proc. 16th IFAC Symposium on Automatic Control in Aerospace, 14-18 june 2004, St. Petersburg, Russia.
- Bérend Nicolas, Bonnans Frédéric, Haddou Mounir, Laurent-Varin Julien et Talbot Christophe, *Fast Linear Algebra for Multiarc Trajectory Op-*

timization. In “Large Scale Nonlinear Optimization”, Proc. Erice workshop, G. Di Pillo and M. Roma eds, Springer Verlag, à paraître (2005).

- Bérend Nicolas, Bonnans Frédéric, Laurent-Varin Julien et Talbot Christophe, *An efficient optimization method dealing with global RLV (ascent and branching) trajectories*, 56th International Astronautical Congress 2005 - Fukuoka, (Japon) - Octobre 2005.

Communication à congrès :

- Laurent-Varin Julien, *An interior points algorithm for optimal control problems*, 18th International Symposium on Mathematical Programming, Copenhagen, Denmark - August 2003.
- Laurent-Varin Julien, *Raffinement optimal de la grille de discrétisation d'un problème de control optimal*, Journées du groupe MODE, Le Havre, 25-27 Février 2004.
- Laurent-Varin Julien, *Optimal Control applied to future space launcher trajectory*, Franco-German-Spanish Conference on Optimization, Avignon, 20-24 Septembre 2004.
- Laurent-Varin Julien, *Interior point algorithm for the optimization of a space shuttle re-entry trajectory* 7ème Congrès Franco-Chilien de Mathématiques appliquées, Santiago (Chile), 14-18 Janvier 2005.
- Laurent-Varin Julien, *Optimisation de trajectoire pour les lanceurs spatiaux futurs*, 2ème Congrès National de Mathématiques appliquées, Evian, 23-27 Mai 2005.
- Laurent-Varin Julien, *Interior point method for state-constrained trajectory optimization*, 22nd IFIP Conference on System Modeling and Optimization, Turin 18-22 Juillet 2005.

Index

- arbre, 49
 - bicolore, 54
 - connexe, 64
 - libre, 64
 - libre orienté, 59, 65
 - raciné, 64
 - raciné bi-couleur, 65
- atmosphère, 154
- axiomes, 136
- circuit, 64
- composantes connexes, 66
- constructeurs, 136
- couloir de rentrée, 33
- degré, 128
 - entrant, 128
 - sortant, 128
- densité, 50
- différentielle élémentaire, 50
- EDO, 45
- FESTIP, 38
- géocentrique, 153
- géodésique, 153
- géopotentielle, 153
- graphe, 64
 - orienté bi-couleur, 65
 - de scénario, 128
 - orienté, 64, 128
- gravité, 25
- méthode
 - de collocation, 47
 - de Runge-Kutta, 46
 - de Runge-Kutta partitionnée, 48
 - de transcription, 12
 - directe, 12
 - indirecte, 12
- masse volumique, 154
- matrice
 - bande, 136
 - creuse, 136
 - identité, 136
 - pleine, 136
- ordre
 - d'un schéma, 51
 - d'un arbre, 50
- point vernal, 22
- polynôme de collocation, 47
- pression atmosphérique, 155
- Problème
 - aux deux bouts, 95
 - de Cauchy, 95
- pseudo-Hamiltonien, 61
- RK, 46
- série B, 50
- série P, 55
- standard US 76, 155
- symétrie, 50
- systèmes partitionnés, 45
- température, 154
- trajectoire

multiarc, 21
triède géographique local, 23
TSTO, 38
type inductif, 136
vitesse du son, 155

Bibliographie

- [1] F. Alvarez, J.F. Bonnans, and J. Laurent-Varin. Expansion of solutions of optimal control problems with logarithmic penalty: a special case. En préparation.
- [2] M. Bardi and I. Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Systems and Control: Foundations and Applications. Birkhäuser, Boston, 1997.
- [3] G. Barles. *Solutions de viscosité des équations de Hamilton-Jacobi*, volume 17 of *Mathématiques et Applications*. Springer, Paris, 1994.
- [4] R. Bellman. *Dynamic programming*. Princeton University Press, Princeton, 1961.
- [5] N. Bérend, J.F. Bonnans, J. Laurent-Varin, M. Haddou, and C. Talbot. Fast linear algebra for multiarc trajectory optimization. Nonconvex Optimization and Its Applications Series. Springer to appear, 2005. proc. Erice conf. 22/06/2004-01/07/2004.
- [6] M. Bergounioux, M. Haddou, M. Hintermüller, and K. Kunisch. A comparison of a Moreau-Yosida-based active set strategy and interior point methods for constrained optimal control problems. *SIAM Journal on Optimization*, 11:495–521 (electronic), 2000.
- [7] J.T. Betts. Survey of numerical methods for trajectory optimization. *AIAA J. of Guidance, Control and Dynamics*, 21:193–207, 1998.
- [8] J.T. Betts. *Practical methods for optimal control using nonlinear programming*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001.
- [9] J.T. Betts, T. Bauer, W. Huffman, and K. Zondervan. Solving the optimal control problem using a nonlinear programming technique part 1:

- General formulation. *American Institute of Aeronautics and Astronautics. Astrodynamics Conference, Seattle, WA, Aug. 20-22, 1984.* 8 p., 1984.
- [10] J.T. Betts, T. Bauer, W. Huffman, and K. Zondervan. Solving the optimal control problem using a nonlinear programming technique part 2: Optimal shuttle ascent trajectories. *American Institute of Aeronautics and Astronautics. Astrodynamics Conference, Seattle, WA, Aug. 20-22, 1984.* 12 p., 1984.
 - [11] J.T. Betts, S.K. Eldersveld, P.D. Frank, and J.G. Lewis. An interior-point algorithm for large scale optimization. In *Large-scale PDE-constrained optimization (Santa Fe, NM, 2001)*, volume 30 of *Lect. Notes Comput. Sci. Eng.*, pages 184–198. Springer, Berlin, 2003.
 - [12] J.T. Betts, S.K. Eldersveld, P.D. Frank, and J.G. Lewis. An interior-point algorithm for large scale optimization. In *Large-scale PDE-constrained optimization (Santa Fe, NM, 2001)*, volume 30 of *Lect. Notes Comput. Sci. Eng.*, pages 184–198. Springer, Berlin, 2003.
 - [13] J.T. Betts and W.P. Huffman. Application of sparse nonlinear programming to trajectory optimization. *AIAA J. of Guidance, Control and Dynamics*, 15:198–206, 1992.
 - [14] R.H. Bishop and D.M. Azimov. Analytical space trajectories for extremal motion with low-thrust exhaust-modulated propulsion. *J. of Spacecraft and Rockets*, 38:897–903, 2001.
 - [15] J.F. Bonnans and Th. Guilbaud. Using logarithmic penalties in the shooting algorithm for optimal control problems. *Optimal Control, Applications and Methods*, 24:257–278, 2003.
 - [16] J.F. Bonnans and G. Launay. Large scale direct optimal control applied to a re-entry problem. *AIAA J. of Guidance, Control and Dynamics*, 21:996–1000, 1998.
 - [17] J.F. Bonnans and J. Laurent-Varin. Computation of order conditions for symplectic partitioned Runge-Kutta schemes with application to optimal control. Technical Report 5398, INRIA, 2004. <http://www.inria.fr/rrrt/rr-5398.html> to appear in *Numerische Mathematik*.
 - [18] J.F. Bonnans and P. Rouchon. *Commande et optimisation de systèmes dynamiques*. Editions de l'École Polytechnique, 2005. À paraître.

- [19] B. Bonnard, L. Faubourg, G. Launay, and E. Trélat. Optimal control of the atmospheric arc of a space shuttle and numerical simulations by multiple-shooting technique. *Prepublications LAA0 Univ. de Bourgogne*, 2002.
- [20] B. Bonnard, L. Faubourg, G. Launay, and E. Trélat. Optimal control with state constraints and the space shuttle re-entry problem. *Journal of Dynamical and Control Systems*, 2002.
- [21] B. Bonnard and E. Trélat. Une approche géométrique du contrôle optimal de l'arc atmosphérique de la navette spatiale. *ESAIM Control Optim. Calc. Var.*, 7:179–222 (electronic), 2002.
- [22] A. E. Bryson and Y.-C. Ho. *Applied optimal control*. Hemisphere Publishing, New-York, 1975.
- [23] R. Bulirsch, E. Nerz, and H.J. Pesch and O. von Stryk. *Combining direct and indirect methods in optimal control: range maximization of a hang glider*, pages 273–288. Birkhäuser, Basel, 1993.
- [24] J. C. Butcher. *Numerical methods for ordinary differential equations*. A Wiley-Interscience Publication. John Wiley & Sons Ltd., Chichester, 2003.
- [25] J.-P. Carrou, editor. *Mécanique spatiale, Tomes I et II*. Cépaduès-Editions-CNES, 1995.
- [26] D. Castillo. Optimisation de trajectoires de véhicules spatiaux par une méthode de points intérieurs. Rapport de fin d'études EPF, ONERA, 2005.
- [27] R. Cominetti and J.P. Dussault. Stable exponential-penalty algorithm with superlinear convergence. *J. Optim. Theory Appl.*, 83:285–309, 1994.
- [28] M.G. Crandall and P.-L. Lions. Viscosity solutions of Hamilton Jacobi equations. *Bull. American Mathematical Society*, 277:1–42, 1983.
- [29] G.B. Dantzig. *Linear Programming and extensions*. Princeton University Press, Princeton, 1963.
- [30] A. L. Dontchev and William W. Hager. The Euler approximation in state constrained optimal control. *Math. Comp.*, 70(233):173–203, 2001.

- [31] A. L. Dontchev, William W. Hager, and Vladimir M. Veliov. Second-order Runge-Kutta approximations in control constrained optimal control. *SIAM J. Numer. Anal.*, 38(1):202–226 (electronic), 2000.
- [32] A.L. Dontchev and W.W. Hager. The Euler approximation in state constrained optimal control. *Mathematics of Computation*, 70:173–203, 2001.
- [33] A.L. Dontchev, W.W. Hager, and V.M. Veliov. Second-order Runge-Kutta approximations in control constrained optimal control. *SIAM Journal on Numerical Analysis*, 38:202–226 (electronic), 2000.
- [34] ESA. Festip system concepts description. Technical report, ESA, August 1998.
- [35] M. Falcone, T. Giorgi, and P. Loreti. Level sets of viscosity solutions: some applications to fronts and rendezvous problems. *SIAM J. Applied Mathematics*, 54:1335–1354, 1994.
- [36] J. Fave. Trajectoires de rentrée dans l’atmosphère de véhicules spatiaux, 1981. Ecole Nationale Supérieure de l’Aéronautique et de l’Espace.
- [37] A.V. Fiacco and G.P. McCormick. *Nonlinear programming*. John Wiley and Sons, Inc., New York-London-Sydney, 1968.
- [38] K.R. Frisch. The logarithmic potentiel method of convex programming, 1955. Memorandum of May 13.
- [39] A.T. Fuller. Relay control systems optimized for various performance criteria. In *Proc. IFAC Congress, Moscow*, pages 510–519. Butterworth, London, 1961.
- [40] A.T. Fuller. Study of an optimum non-linear control system. *J. of Electronics and Control*, 15:63–71, 1963.
- [41] D.M. Gay, Overton M.L., and Wright M.H. A primal-dual interior method for nonconvex nonlinear programming. *Advances in Nonlinear Programming, Kluwer Academic Publishers*, 1998.
- [42] M. Gertz, J. Nocedal, and A. Sartenaer. A starting-point strategy for nonlinear interior methods. *Applied Mathematics Letters*, 17:945–952, 2004.
- [43] P.E. Gill, W. Murray, and M.H. Wright. *Practical optimization*. Academic Press, London, 1981.

- [44] L. G. Hačijan. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk SSSR*, 244(5):1093–1096, 1979.
- [45] W. Hager. Runge-Kutta methods in optimal control and the transformed adjoint system. *Numerische Mathematik*, 87(2):247–282, 2000.
- [46] E. Hairer, Ch. Lubich, and G. Wanner. *Geometric numerical integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2002.
- [47] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I*. Springer-Verlag, Berlin, second edition, 1993.
- [48] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1993.
- [49] E. Hairer and G. Wanner. *Solving ordinary differential equations. II*. Springer-Verlag, Berlin, second edition, 1996.
- [50] S.P. Han. A globally convergent method for nonlinear programming. *J. Optimization theory and applications*, 22:297–309, 1977.
- [51] J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages, and computation*. 1979. Addison-Wesley Series in Computer Science.
- [52] T. Jockenhövel, L. T. Biegler, and A. Wächter. Dynamic optimization of the tennessee eastman process using the optcontrolcentre. *Computers and Chemical Engineering*, 27:1513–1531, 2003.
- [53] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.
- [54] V. Klee and G. J. Minty. How good is the simplex algorithm? In *Inequalities, III (Proc. Third Sympos., Univ. California, Los Angeles, Calif., 1969; dedicated to the memory of Theodore S. Motzkin)*, pages 159–175. Academic Press, New York, 1972.
- [55] D. Kraft. Finite-difference gradients versus error-quadrature gradients in the solution of parameterized optimal control problems. *Optimal Control, Appl. and Methods*, 2:191–199, 1981.

- [56] F. Leibfritz and E.W. Sachs. Inexact SQP interior point methods and large scale optimal control problems. *SIAM Journal on Control and Optimization*, 38:272–293 (electronic), 1999.
- [57] G. Li and F. Ruskey. The advantages of forward thinking in generating rooted and free trees. In *10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. <http://www.theory.csc.UVic.CA/fruskey/> ou <http://www.theory.csc.UVic.CA/cos/>.
- [58] H Maurer. Numerical solution of singular control problems using multiple shooting techniques. *JOTA*, 18:235–257, 1976.
- [59] H. Maurer. On the minimum principle for optimal control problems with state constraints. Schriftenreihe des Rechenzentrum 41, Universität Münster, 1979.
- [60] F. Messine. *Méthodes d’Optimisation Globale basées sur l’Analyse d’Intervalle pour la Résolution des Problèmes avec Contraintes*. PhD thesis, LIMA-IRIT-ENSEEIH-ENPT, Toulouse, 1997.
- [61] S.K. Mitter. Successive approximation methods for the solution of optimal control problems. *Automatica*, 3:135–149, 1966.
- [62] J.J. Moré and D.C. Sorensen. Computing a trust region step. *SIAM J. Scientific and Statistical Computing*, 4:553–572, 1983.
- [63] A. Murua. On order conditions for partitioned symplectic methods. *SIAM J. Numer. Anal.*, 34(6):2204–2211, 1997.
- [64] M.J.D. Powell. A method for nonlinear constraints in minimization problems. In *Optimization (Sympos., Univ. Keele, Keele, 1968)*, pages 283–298. Academic Press, London, 1969.
- [65] M.J.D. Powell. Algorithms for nonlinear constraints that use Lagrangian functions. *Mathematical Programming*, 14:224–248, 1978.
- [66] I.M. Ross and F. Fahroo. Pseudospectral knotting methods for solving optimal control problems. *AIAA Journal of Guidance, Control and Dynamics*, 27:397–508, 2004.
- [67] J. M. Sanz-Serna and L. Abia. Order conditions for canonical Runge-Kutta schemes. *SIAM Journal on Numerical Analysis*, 28(4):1081–1096, 1991.

- [68] D. F. Shanno and R. J. Vanderbei. Interior-point methods for nonconvex nonlinear programming: orderings and higher-order methods. *Math. Program.*, 87(2, Ser. B):303–316, 2000. Studies in algorithmic optimization.
- [69] M. Sofroniou and W. Oevel. Symplectic Runge-Kutta schemes. I. Order conditions. *SIAM J. Numer. Anal.*, 34(5):2063–2086, 1997.
- [70] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer-Verlag, New-York, 1993.
- [71] P. Tsiotras and H.J. Kelley. Drag-law effects in the Goddard problem. *Automatica*, 27-3:481–490, 1991.
- [72] M. Ulbrich, S. Ulbrich, and Luís N. Vicente. A globally convergent primal-dual interior-point filter method for nonlinear programming. *Math. Program.*, 100(2, Ser. A):379–410, 2004.
- [73] L.N. Vicente. On interior-point Newton algorithms for discretized optimal control problems with state constraints. *Optimization Methods and Software*, 8:249–275, 1998.
- [74] A. Wächter and L.T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*. To appear.
- [75] M. Weiser. *Function Space Complementarity, Methods for Optimal Control Problems*. PhD thesis, Fachbereich Mathematik und Informatik der Freien Universität Berlin, Februar 2001.
- [76] S.J. Wright. Interior point methods for optimal control of discrete time systems. *Journal of Optimization Theory and Applications*, 77:161–187, 1993.