Université Catholique de Louvain
Faculté des Sciences Appliquées

Génération et Parallélisation des Équations du Mouvement de Systèmes Multicorps par l'Approche Symbolique.

Thèse présentée par

Tony Postiau
Ingénieur Civil Mécanicien

en vue de l'obtention du grade de

Docteur en Sciences Appliquées

Prof. Paul Fisette (promoteur)	Université Catholique de Louvain
Prof. Jean-Didier Legat (promoteur)	Université Catholique de Louvain
Prof. Jean-Claude Samin (encadrement)	Université Catholique de Louvain
Prof. Vincent Legat	Université Catholique de Louvain
Prof. Jean-Pierre David	Université de Montréal (Québec)
Prof. Manfred Hiller	Université de Duisburg (Allemagne)

Louvain-la-Neuve
Septembre 2004

Remerciements

Au terme de cette thèse de doctorat, je tiens à exprimer ma gratitude aux personnes qui m'ont soutenu durant ces quelques années.

Je remercie naturellement mes deux promoteurs Paul Fisette et Jean-Didier Legat qui, grâce à leurs conseils éclairés, m'ont régulièrement aidé à orienter mon travail de recherche dans la bonne direction.

Je suis tout spécialement reconnaissant à Jean-Claude Samin pour l'intérêt particulier qu'il a accordé à ma thèse. C'est lui qui m'a transmis une grande partie de mes connaissances en modélisation et simulation de robots et de systèmes articulés, ainsi que la passion de ce domaine de la mécanique.

Que les membres du jury soient également remerciés pour l'attention qu'ils ont porté à mon texte et les commentaires constructifs qu'ils m'ont transmis suite à sa lecture et lors de ma défense privée. Merci à Jean-Pierre David pour sa collaboration et sa contribution au développement de la vectorisation des équations et les simulations sur FPGA. Merci à Vincent Legat pour l'intérêt porté à la modélisation des systèmes multicorps et le partage de son point de vue sur certaines de nos conventions. Merci à Manfred Hiller pour avoir accepté de faire partie de ce jury et pour l'attention accordée à mon travail.

Je remercie l'équipe du centre de calcul intensif, et en particulier Bernard Van Renterghem, pour sa disponibilité, ses conseils et sa collaboration pour l'évaluation des gains en temps calcul obtenus sur ordinateurs parallèles.

Je remercie à nouveau tout particulièrement Paul Fisette et Jean-Claude Samin de l'aide qu'ils m'ont apportée pour l'élaboration de ce document.

Outre un encadrement scientifique de qualité, j'ai bénéficié au sein de l'unité PRM de conditions de travail excellentes et d'une atmosphère des plus agréables. J'en remercie tous mes collègues, professeurs, chercheurs et techniciens : l'esprit de l'unité, c'est vous qui le créez, merci de votre sympathie.

Paul et Jean-Claude, vous avez été pour moi bien plus que de bons professeurs, vous avez été des collègues passionnés, confiants en moi et impliqués dans mon travail de recherche. Votre attitude et vos qualités humaines contribuent de façon importante à la motivation et au plaisir de la recherche en PRM.

Nous passons tous une partie importante de notre vie au travail parmi nos collègues. C'est une chance de pouvoir passer tout ce temps dans une bonne ambiance avec un ami : Merci Jean-François.

Être entouré de ses amis et de sa famille constitue un élément important qui contribue à l'équilibre nécessaire pour mener à bien un projet tel qu'une thèse de doctorat. Mes amis, mes parents : votre amitié, votre affection et vos encouragements m'ont apporté un soutien essentiel. Merci spécialement à Nancy, ma soeur, pour l'accueil particulier que j'ai toujours reçu au sein de sa famille. J'en ai très souvent profité pour me distraire et m'amuser en allant jouer avec "les Monstres". Et ce n'est pas fini !

Au cours de cette année, nous sommes quelques collègues et amis proches à avoir terminé notre doctorat : Laurent, Cécile, Bruno et bientôt Olivier. Quel aboutissement, les amis !

Finalement, je tiens à soutenir et à encourager à mon tour ceux qui sont encore en pleine recherche ou qui vont bientôt se lancer dans l'aventure : soyez motivés et confiants, vous en retirerez de la fierté et du plaisir !

Cher lecteur, merci pour l'intérêt porté à ce texte.

Ce travail de recherche a bénéficié du support financier et scientifique des pôles d'Attraction Inter-universitaire IUAP-IV-24 et IAP-V-06.

Table des matières

Introduction

Qu'est-ce qu'un *système multicorps* ? Nous en avons tous déjà vus autour de nous, nous en avons même utilisés. En réalité, nous en sommes nous-mêmes, un bel exemple.

Un vélo, une voiture, un camion, un hélicoptère, une grue, un moteur, un robot, un satellite, le corps humain : tous sont des exemples de systèmes multicorps. On les appelle également des systèmes mécaniques articulés.

Un système multicorps[1] est un système composé de plusieurs parties rigides ou flexibles, appelées des *corps*, qui sont reliés entre eux par des *articulations*.

On distingue souvent ces systèmes selon leur structure. Celle-ci peut être de type *arborescente* ou ouverte, lorsque sa topologie peut être représentée par un arbre, ou bien de type *bouclée* ou fermée, lorsque le graphe topologique contient des cycles. La figure 1 illustre les deux types de topologie, ouverte ou fermée.

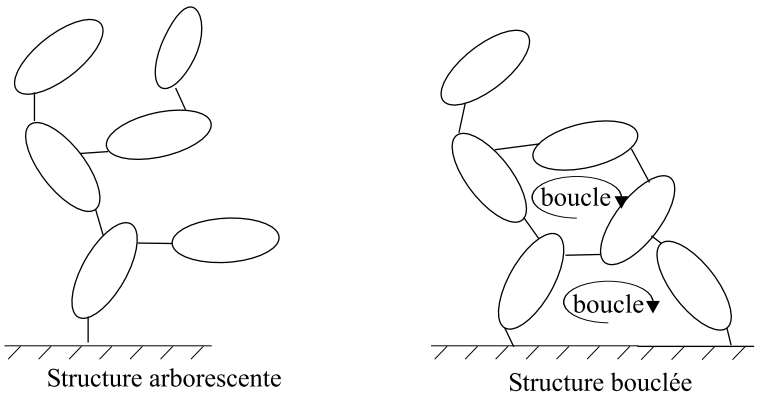


FIG. 1 – Structures arborescentes ou bouclées

A l'époque actuelle, de nombreuses applications impliquent l'étude de tels

[1] en anglais : *MultiBody System* (MBS)

systèmes. Ceux-ci sont modélisés et analysés à l'aide d'ordinateurs pour en comprendre et pour en améliorer le fonctionnement ou même pour mieux les concevoir.

Dans le domaine du sport et de la compétition, on étudie les mouvements des athlètes. Dans le domaine de l'automobile et des transports, on recherche continuellement à améliorer les performances, le confort et la sécurité des voitures, des camions et des trains. Dans le domaine aéronautique, les dispositifs de contrôle des avions et des hélicoptères sont constitués de mécanismes complexes dont le comportement dynamique est étudié pour en garantir le fonctionnement correct et performant.

Dans l'espace, des bras robotiques sont utilisés par les astronautes pour extraire les satellites de la navette ou pour l'assemblage et la construction de la station spatiale internationale. A quelques centaines de millions de kilomètres de la terre, les robots mobiles d'exploration Sojourner, Spirit et Opportunity, sont tour à tour partis à la découverte du sol martien.

Quel que soit le domaine d'application, l'étude de ces systèmes repose entre autres choses sur l'utilisation d'*équations* qui en décrivent le comportement.

Nous avons tous été confrontés à des équations au cours de nos études : $y = a\ x + b$, $F = m\ a$, $U = R\ I$, $E = m\ c^2$, etc. Quels que soient les domaines desquels ces équations proviennent, il s'agit toujours d'expressions écrites à l'aide de symboles qui représentent des opérations arithmétiques $+, \times, -, /$ ou autres, ainsi que des variables $x, y, \ldots$ ou des paramètres $a, b, m, \ldots$. Nous nous souvenons tous avoir du écrire et même parfois résoudre de telles équations, à la main, avec un stylo et du papier.

Nous nous souvenons également que cela devenait fastidieux et difficile lorsque le problème était compliqué et qu'il n'y avait plus, une seule mais, plusieurs équations à traiter. Heureusement, pour ces situations difficiles qui correspondent bien souvent aux applications réelles, nous disposons actuellement des ordinateurs. Ces machines à calculer extraordinaires sont bien utiles aux ingénieurs pour les aider à évaluer et à résoudre les équations lorsqu'elles sont nombreuses et compliquées.

Mais pour qu'ils exécutent un travail de calcul, les ordinateurs ont besoin d'instructions, de programmes. Naturellement, il y a plusieurs possibilités pour donner ces instructions à un ordinateur. L'une de ces possibilités est précisément d'écrire ces instructions sous la même forme que les équations que nous écrivons à la main. Dans ce cas, il est donc nécessaire d'écrire ces équations, ce qui constitue un travail fastidieux, comme nous l'avons déjà dit. L'idée intéressante est alors d'utiliser l'ordinateur pour *écrire* les équations. Cette première phase d'écriture des équations par l'ordinateur, sous la même forme symbolique que nous connaissons, s'appelle la *génération symbolique*.

Une fois ces équations générées, l'ordinateur va ensuite pouvoir les utiliser pour effectuer les calculs requis, par exemple pour l'étude du mouvement d'un

système multicorps, puisque c'est le domaine qui nous intéresse. Lorsque le système devient complexe ou de grande taille, le nombre d'équations augmente, et le temps nécessaire pour exécuter les calculs également. Une solution pour réduire le temps de calcul consiste alors à utiliser plusieurs ordinateurs ou bien, un ordinateur qui contient plusieurs processeurs.

Un nouveau problème surgit alors : celui de la répartition du travail. Certaines règles classiques du travail en groupe sont d'application ici : l'efficacité de l'équipe sera meilleure si chacun peut effectuer sa partie du travail sans avoir besoin des autres, ou alors avec un minimum d'interactions. En ce qui nous concerne, il s'agit donc de séparer l'ensemble des équations à calculer et de constituer des sous ensembles suffisamment indépendants afin qu'ils puissent être exécutés efficacement par les différents processeurs utilisés. Cette phase de division et de répartition des équations s'appelle la *parallélisation*.

Après ces quelques explications qui nous permettent de mieux comprendre le titre de ce travail, nous vous proposons de découvrir le domaine dans lequel ce travail s'inscrit, à savoir, la modélisation et l'étude de la dynamique des systèmes multicorps.

Etat de l'art

L'étude des systèmes multicorps n'est pas une discipline nouvelle. Elle date en fait du 17^{e}siècle, époque à laquelle vivait Isaac Newton. Euler, d'Alembert et Lagrange y ont grandement contribué au 18^{e}siècle. Mais, c'est avec l'avènement de la robotique et de l'informatique à la fin du 20^{e}siècle que l'étude de la dynamique des systèmes multicorps a réellement pris son essor et est devenue une discipline à part entière dans le domaine de la mécanique. Un aperçu historique complet de cette discipline est dressé par W. Schiehlen dans [1].

Nous présentons ici les éléments relatifs à l'étude des systèmes articulés qui nous semblent les plus importants pour expliquer ensuite les objectifs de notre travail.

Modélisation des systèmes multicorps

Dans toute démarche de modélisation de système, l'objectif de l'ingénieur est d'obtenir des équations qui représentent le comportement du système étudié. Afin d'écrire ces équations, il est tout d'abord nécessaire de se donner un jeu de variables qui permet de décrire la configuration, ou l'état, du système : ce sont *les coordonnées généralisées*. Différents choix de coordonnées sont possibles. Selon les coordonnées choisies, l'ingénieur peut alors utiliser différentes méthodes, ou formalismes pour écrire les équations permettant d'étudier le comportement du système. Dans le cas des systèmes multicorps, ces équations concernent les mouvements des corps qui sont soumis aux lois de la mécanique rationnelle.

On distingue deux problèmes classiques dans l'analyse dynamique des systèmes multicorps :

- Le problème *dynamique inverse* consiste à déterminer les efforts Q à transmettre par les articulations du système afin que celui-ci évolue selon une trajectoire prescrite par les valeurs q des coordonnées généralisées ainsi que par les valeurs de leurs dérivées premières $\dot{q}$ et secondes $\ddot{q}$.
  On peut l'exprimer par l'équation globale suivante :

$$Q = Q(q, \dot{q}, \ddot{q}) \tag{1}$$

- Le problème *dynamique direct* consiste à déterminer les accélérations généralisées $\ddot{q}$ lorsque le système est soumis aux forces articulaires Q dans une configuration connue en terme de valeurs des coordonnées généralisées q et de leur dérivée première $\dot{q}$.
  On peut l'exprimer par l'équation globale suivante :

$$\ddot{q} = \ddot{q}(q, \dot{q}, Q) \tag{2}$$

Typiquement, la dynamique inverse concerne le contrôle de robot et la dynamique directe concerne la simulation des systèmes multicorps, quels qu'ils soient.

Les accélérations généralisées $\ddot{q}$ et les forces articulaires Q dépendent linéairement les unes des autres. Cette dépendance est exprimée par la relation suivante qui correspond aux *équations du mouvement* du système

$$M(q)\,\ddot{q} + c(q, \dot{q}, F_{ext}, L_{ext}, g) = Q \tag{3}$$

où

- $M(q)$ représente la matrice de masse généralisée du système,
- $c(q, \dot{q}, F_{ext}, L_{ext}, g)$ regroupe les termes gyroscopique, de Coriolis, de gravité, et les forces et couples extérieures appliquées sur les corps, ainsi que la gravité g,

Dans la plupart des cas, les équations du mouvement sont accompagnées d'un ensemble d'équations supplémentaires qui expriment des relations entre les coordonnées généralisées. Ce sont des *équations de contraintes*. Le nombre et la nature des équations de contraintes dépendent essentiellement du choix des coordonnées, de la structure du système et des conditions auxquelles le système est soumis. Ces équations de contraintes[2] sont algébriques et généralement nonlinéaires. On les écrit souvent sous la forme implicite suivante :

$$h(q) = 0 \tag{4}$$

[2]On considère dans ce travail que les contraintes sont scléronomes, c'est à dire qu'elles ne dépendent pas explicitement du temps. On considère simplement que $h(q,t) \equiv h(q, q(t))$

L'ensemble des équations nécessaires à l'étude du mouvement d'un système multicorps soumis à des contraintes est alors le suivant :

$$M(q)\,\ddot{q} + c(q, \dot{q}, F_{ext}, L_{ext}, g) = Q + J(q)^T \lambda \quad (5)$$

$$h(q) = 0 \quad (6)$$

où $J(q) = \partial \dot{h}(q)/\partial \dot{q}$ est la matrice Jacobienne associée aux contraintes et λ est le vecteur des multiplicateurs de Lagrange.

La présence simultanée d'équations différentielles et algébriques pose une difficulté. En effet leur intégration numérique nécessite l'utilisation de méthodes particulières. Il est toutefois possible de les transformer en équations purement différentielles. En pratique les dérivées premières et secondes des équations de contraintes

$$\dot{h}(q, \dot{q}) = J(q)\dot{q} \quad (7)$$

$$\ddot{h}(q, \dot{q}, \ddot{q}) = J(q)\ddot{q} + \dot{J}\dot{q} \quad (8)$$

jouent un rôle important pour l'élaboration de ces formulations purement différentielles.

Nous présentons dans les paragraphes qui suivent les différentes options les plus classiques en ce qui concerne :

1. le choix des *coordonnées* pour représenter la configuration d'un système multicorps,
2. les *formalismes* utilisés pour obtenir les équations du mouvement du système,
3. les méthodes de *traitements des équations de contraintes*,
4. les différentes *formulations* des équations du mouvement en vue de leur utilisation pour l'analyse numérique du comportement du système.

Coordonnées

Le choix des coordonnées a une influence directe sur le type de formalisme utilisable et le nombre d'équations obtenues pour la modélisation mathématique d'un système multicorps. Il existe de différents types de coordonnées dont les plus connues sont les suivantes.

Coordonnées relatives Également appelées coordonnées articulaires, elle représentent les débattements angulaires ou linéaires dans les articulations qui relient les corps du système (ex. :[2]).

L'utilisation des coordonnées relatives permet d'exploiter explicitement et avantageusement la topologie du système pour un calcul récursif des grandeurs cinématiques et dynamiques.

Dans le cas des systèmes à topologie ouverte, c'est-à-dire les systèmes ayant une structure linéaire ou arborescente, le jeu de coordonnées est minimal. Le nombre de coordonnées correspond exactement au nombre de degrés de liberté du système. Aucune équation de contrainte ne doit être ajoutée aux équations du mouvement du système et toutes les coordonnées sont donc *indépendantes*. Elles conduisent à un jeu d'équations purement différentielles.

Cet avantage est toutefois limité aux systèmes simples car en pratique la plupart des systèmes réels sont complexes et ont une topologie fermée, c'est-à-dire que leur structure contient des boucles cinématiques qui vont nécessiter l'écriture d'équations de contraintes algébriques supplémentaires pour exprimer des conditions de fermeture de ces boucles. On obtient alors des équations différentielles et algébriques, qui pourront être traitées telles quelles, ou transformées en équations différentielles ordinaires avant leur utilisation.

Toutefois, le choix des coordonnées relatives assure toujours un jeu de coordonnées et d'équations minimal par rapport aux autres type de coordonnées.

Coordonnées indirectes Ces coordonnées ont été introduites dans [3] et peuvent être considérées comme un sous-groupe des coordonnées relatives. Pour des systèmes qui présentent certaines séquences particulières d'articulations telles que :

1. une succession[3] d'articulations rotoïdes dont les axes sont parallèles,
2. une succession[4] d'articulations prismatiques dont les axes sont parallèles,
3. une articulation sphérique,

il est possible d'obtenir un jeu d'équations significativement plus compact en utilisant des coordonnées indirectes au lieu des coordonnées relatives. Les coordonnées indirectes correspondent alors à des combinaisons des coordonnées relatives dans les deux premiers cas. L'utilisation de coordonnées indirectes nécessite l'utilisation d'une double arborescence pour traiter séparément le calcul récursif des translations et des rotations des corps. Elles ont été étudiées dans [4] et [5] et sont utilisées dans le logiciel DYNAFLEX [6].

Coordonnées absolues Les coordonnées absolues [7, 8] expriment les positions et orientations des corps par rapport à un repère de référence unique. Six coordonnées, trois pour les translations et trois[5] pour les rotations, sont utilisées pour décrire la configuration individuelle de chaque corps du système. Le nombre de coordonnées est alors bien souvent très supérieur au nombre de degrés de liberté du système.

[3] éventuellement entre coupée d'articulations prismatiques d'axes quelconques

[4] éventuellement entre coupée d'articulations prismatiques d'axes quelconques ou d'articulations rotoïdes de même axe

[5] ou quatre dans le cas des paramètres d'Euler ou Quaternions

Elles ont l'avantage de rendre l'écriture des équations du mouvement des corps très directe et très facile. De plus, la matrice de masse globale du système est constante et (bloc) diagonale, et les équations du mouvement des corps sont très simples et peuvent être calculées indépendamment pour chaque corps.

En revanche, les restrictions imposées par les articulations au niveau des mouvements relatifs des corps les uns par rapport aux autres, nécessitent l'écriture d'équations de contraintes relativement complexes. De une à cinq équations de contraintes sont nécessaires selon le type d'articulation, ce qui compense largement la simplicité des équations du mouvement.

Ainsi, que le système soit arborescent ou contienne des boucles cinématiques, on obtient systématiquement un jeu d'équations différentielles et algébriques. Le nombre d'équations est alors bien plus grand que dans le cas de l'utilisation de coordonnées relatives.

Coordonnées naturelles L'utilisation des coordonnées naturelles [9, 10] implique la représentation des corps du système par un ensemble de *masses ponctuelles* reproduisant les caractéristiques dynamiques du corps. Ceci implique une disposition particulière de ces points et une distribution adéquate de la masse du corps original sur chaque point.

Un avantage essentiel de cette méthode est qu'elle ne fait intervenir que des coordonnées cartésiennes absolues. Ceci implique une matrice de masse diagonale et constante, et donc, des équations du mouvement extrêmement simples.

Les positions relatives des points qui représentent un corps rigide sont imposées à l'aide d'équations de contraintes. Ces équations expriment essentiellement la conservation des distances entre les différentes masses ponctuelles qui correspondent à un même corps rigide.

Les mouvements relatifs des corps, imposés par les articulations, sont également exprimés à l'aide d'équations de contraintes. Toutefois, il n'est pas toujours nécessaire d'écrire autant d'équations de contraintes que dans le cas de l'utilisation des coordonnées absolues, et ceci en plaçant judicieusement certains points sur les axes des articulations. Par exemple, dans le cas de deux corps reliés par une articulation sphérique, en utilisant pour les deux corps, un seul et même point situé au centre de l'articulation, il ne sera pas nécessaire d'écrire une seule équation de contrainte pour cette articulation.

Un second avantage de cette méthode est que les équations de contraintes sont quadratiques en les coordonnées et donc la Jacobienne des contraintes est linéaire, ce qui favorise le traitement numérique de ces contraintes.

L'utilisation des coordonnées naturelles permet d'obtenir des modèles qui offrent d'excellentes performances en simulation, parfois supérieures à celles des modèles qui utilisent des coordonnées relatives, dans le cas de systèmes de petite taille [10, 11]. Elles offrent également une plus grande robustesse

pour faire face à certaines configurations singulières que peuvent adopter des mécanismes particuliers [10].

Toutefois, comme pour le choix des coordonnées absolues, on doit gérer un grand nombre de variables et d'équations différentielles et algébriques, un nombre largement supérieur au nombre de degrés de liberté du système. Ce qui amène les adeptes de ces coordonnées à se poser la question de leur intérêt pour la modélisation de systèmes réels de grande taille tels que ceux rencontrés dans les applications industrielles [10].

Remarque En pratique, il est assez difficile de défendre un choix de coordonnées face à tous les types d'applications envisageables. Étant donnés les avantages et les inconvénients de chaque type de coordonnées, il est souvent plus efficace d'utiliser un ensemble mixte, combinant à la fois des coordonnées relatives et des coordonnées absolues. Ceci permet alors de bénéficier des avantages de chacune tout en compensant les inconvénients respectifs.

Formalismes

Dans le cas de l'utilisation de coordonnées absolues, l'écriture des équations du mouvement du système multicorps ne repose pas à proprement parler sur l'utilisation d'un formalisme particulier. Les équations de Newton (9) et d'Euler (10) peuvent être écrites directement pour chaque corps.

$$m\,\ddot{\mathbf{R}}^G = \mathbf{F} \tag{9}$$

$$\mathbf{I}^G\,.\,\dot{\boldsymbol{\omega}} + \tilde{\boldsymbol{\omega}}\,.\,\mathbf{I}^G\,.\,\boldsymbol{\omega} = \mathbf{L}^G \tag{10}$$

avec

- $\ddot{\mathbf{R}}^G$ est l'accélération absolue du centre de masse du corps,
- m est la masse du corps,
- $\mathbf{F}$ est la résultante des forces appliquées sur le corps,
- $\mathbf{I}^G$ est le tenseur d'inertie du corps, par rapport à son centre de masse,
- $\boldsymbol{\omega}$ est le vecteur vitesse angulaire du corps,
- $\mathbf{L}^G$ est la résultante des moments des forces appliquées sur le corps, par rapport à son centre de masse.

L'utilisation des coordonnées naturelles simplifie encore plus cette étape car il est évidemment inutile d'écrire les équations d'Euler pour un système de masses ponctuelles.

Lorsque les coordonnées relatives sont utilisées, diverses approches sont envisageables pour obtenir les équations du mouvement. Les plus connues sont

- la méthode de Lagrange,
- le Principe des Travaux Virtuels ou le Principe des Puissances Potentielles
- et les méthodes dites "récursives".

La méthode de Lagrange est connue pour être sous optimale, en terme de nombre d'opérations, pour la génération des équations du mouvement des systèmes que l'on rencontre en réalité, hors du cadre des exemples académiques. Les méthodes variationnelles telles que les Travaux Virtuels ou les Puissances Potentielles ne sont efficaces que si elles se basent sur une approche récursive. Dans le cas contraire, elles conduisent à un nombre d'opérations bien supérieur à ce que l'on peut obtenir par les méthodes récursives [12, 13].

Depuis plus de vingt ans, les formalismes récursifs basés sur les équations de Newton-Euler ont fait l'objet de diverses recherches visant à réduire leur complexité algorithmique [14]. Initialement développés dans le domaine de la robotique [15], ils ont progressivement été étendus de manière à être utilisables pour l'étude des systèmes multicorps à topologie arborescente et puis bouclée.

Actuellement, trois types d'algorithmes de génération des équations du mouvement se distinguent. Nous les présentons brièvement dans les paragraphes qui suivent. Pour de plus amples détails nous suggérons au lecteur de consulter les quelques références proposées.

L'algorithme récursif de Newton-Euler Cette méthode[6] a été développée à l'origine pour obtenir la dynamique inverse (1) de robots manipulateurs séries [15], et ceci avec une complexité algorithmique $O(N)$.

On montre dans [16] qu'il est possible d'obtenir la dynamique directe en appliquant l'algorithme récursif plusieurs fois en bloquant toutes les articulations du système, sauf une. Le procédure est ainsi répétée pour chaque articulation et permet d'obtenir les coefficients de la matrice de masse du système (3). La complexité de la procédure est alors de $O(N^2)$. Cette matrice de masse exprime la dépendance linéaire des forces généralisées articulaires par rapport aux accélérations articulaires.

L'algorithme récursif de Newton Euler est basé sur un double parcours récursif de la topologie du système[7].

1. Lors d'un premier parcours direct, partant de la base vers les extrémités, on exprime de façon récursive la cinématique de chaque corps, c'est-à-dire sa position, son orientation, ses vitesses linéaire et angulaire etc.,
2. Lors d'un second parcours, inverse, repartant des extrémités vers la base, on calcule successivement les forces généralisées présentes dans les articulations ainsi que les termes de la matrice de masse du système.

Cet algorithme est présenté en détail dans [17, 12, 13] et est également reproduit dans ce travail à la section 1.3.1.

Si l'extraction de la matrice de masse est d'une complexité algorithmique en $O(N^2)$, le calcul explicite des accélérations articulaires par résolution directe

[6] en anglais : *Composite Inertia Method* ou *Composite Rigid Body Algorithm* CRBA
[7] on considère ici que la topologie est arborescente

du système d'équations du mouvement, linéaires en les accélérations, est d'une complexité algorithmique en $O(N^3)$ [18].

Toutefois, cette méthode est toujours plus efficace que la méthode de Lagrange [16, 12, 13].

L'utilisation d'un jeu minimal de paramètres barycentriques [19, 20] permet de réduire le nombre d'opérations nécessaires pour le calcul des forces articulaires ou des termes de la matrice de masse, sans toutefois diminuer la complexité algorithmique en $O(N^2)$. Cette formulation Newton-Euler avec paramètres barycentriques a été utilisée avantageusement pour des applications où l'identification dynamique est nécessaire pour l'évaluation des paramètres dynamiques du système [21].

Algorithmes récursifs O(N) La complexité de l'algorithme récursif de Newton-Euler pose un problème pour la modélisation des systèmes qui contiennent un grand nombre de corps en série. Cet algorithme a été étudié et modifié afin de réduire sa complexité. Des algorithmes[8] de calcul des accélérations dont la complexité est $O(N)$ ont été développés [22, 23] initialement pour les systèmes à topologie linéaire, puis pour les systèmes à topologie arborescente [24, 25, 26] et finalement pour les systèmes contenant des boucles cinématiques [27, 28][9].

Ils permettent d'éviter de calculer explicitement la matrice de masse du système et de résoudre un système d'équations linéaires pour obtenir les valeurs des accélérations généralisées.

Le calcul des accélérations avec une complexité en $O(N)$ implique un triple parcours récursif de la topologie du système :

1. un parcours direct de la base vers les extrémités pour le calcul récursif de la cinématique absolue des corps,
2. un parcours inverse des extrémités vers la base pour le calcul de forces et de matrices d'inerties partielles,
3. un dernier parcours direct pour le calcul des accélérations généralisées associées aux articulations.

On trouvera un présentation du principe dans [14] et le développement détaillé de l'algorithme appliqué aux structures arborescentes et bouclées dans [12] et [28].

Lorsque le système contient plus d'une dizaine de corps en série, ce formalisme dont la complexité est linéaire, devient plus attractif que le formalisme récursif classique en $O(N^2)$. Toutefois le nombre total de corps du système à partir duquel le formalisme $O(N)$ devient plus intéressant dépend de l'efficacité de l'implémentation du formalisme et de la topologie du système. On montre

[8] en anglais : *Articulated Inertia Method* ou *Articulated Body Algorithm* ABA

[9] On notera toutefois que le traitement des équations de contraintes entraîne généralement un surcoût $O(m^3)$, où m est le nombre de contraintes

dans [17] et [12] que le nombre pivot se situe entre 12 et 30, selon la topologie linéaire ou arborescente du système. L'utilisation des paramètres barycentriques dans le formalisme Newton Euler récursif repousse également ce nombre pivot vers des valeurs supérieures [21].

Formalismes O(N) adaptés au calcul parallèle Bien que les algorithmes $O(N)$ classiques offrent les meilleures performances pour la génération et le calcul du modèle dynamique direct utilisé pour la simulation des systèmes multicorps de grande taille, ils présentent l'inconvénient majeur de ne pas être parallélisables. En effet, l'algorithme récursif est essentiellement séquentiel, ce qui ne permet pas de profiter de plusieurs processeurs pour en accélérer l'exécution. Ceci constitue un obstacle à l'étude de systèmes contenant des centaines voire des milliers de corps en série. Si ce genre de système n'est pas commun dans le domaine des mécanismes, c'est en revanche habituel dans un domaine tel que l'étude de la dynamique moléculaire [29].

Diverses méthodes ont été élaborées depuis une quinzaine d'années. Elles sont présentées et utilisées sous le nom anglais de *Velocity Transformation Method* [30, 31, 9, 10, 11] ou encore de *Constraint Force Algorithm* [32, 33, 34, 29, 35].

Elles sont utilisées avantageusement pour la simulation de systèmes multicorps classiques dans [10, 11, 33, 34] et pour l'étude de la dynamique de molécules de polymères de polyéthylène dans [29].

Le principe général à la base de l'élaboration de ces méthodes est présenté dans [14] sous le nom de *Global Analysis Technique*.

Ces méthodes combinent les avantages de l'utilisation des coordonnées relatives et absolues. Elles tirent avantage d'une représentation topologique arborescente

- pour obtenir un jeu minimal de coordonnées relatives indépendantes qui représentent la configuration du système de façon univoque,
- pour éviter l'écriture et surtout le traitement de contraintes exprimant les restrictions de mouvements relatifs imposées aux corps par les articulations.

Paradoxalement, le calcul explicite des forces de contraintes constitue un élément clé de ces méthodes [34], alors qu'on évite soigneusement de les calculer explicitement lorsqu'elles ne sont pas nécessaires[10], dans les approches récursives classiques.

Les vitesses et accélérations absolues des corps sont exprimées en fonction des dérivées premières et secondes des coordonnées relatives en utilisant des matrices de transformation. Les forces appliquées sur les corps sont décomposées en forces articulaires et forces de contraintes. Les équations du mouvement sont alors écrites pour chaque corps en bénéficiant des avantages de la formula-

[10] par exemple, si on ne désire pas étudier la résistance de la structure du système

tion basée sur les coordonnées absolues, à savoir le caractère creux et constant de la matrice de masse.

Le caractère creux des matrices de masse et de transformation des vitesses, ainsi qu'une factorisation originale de l'inverse de la matrice de masse sur base des matrices de transformation, permettent de résoudre facilement les équations du mouvement et confèrent à ces méthodes une complexité $O(N)$ et un haut degré de parallélisme, ce qui permet également de réduire sa complexité à $O(logN)$ en utilisant $O(N)$ processeurs [14, 34, 29].

Ces caractéristiques en font des méthodes idéales pour la modélisation et l'analyse de systèmes de (très) grande taille lorsqu'on dispose d'un environnement de calcul (massivement) parallèle [34, 36].

On notera toutefois que ces méthodes sont moins performantes que l'algorithme $O(N)$ récursif, dans un contexte de calcul séquentiel [34].

Traitement des contraintes

Les équations de contraintes qui accompagnent les équations du mouvement peuvent avoir différentes origines, dont les plus communes sont les suivantes :

1. Le mouvement de certaines articulations est imposé. L'évolution temporelle des valeurs des coordonnées associées à ces articulations, ainsi que les valeurs de leurs dérivées premières et secondes, sont alors prescrites par des fonctions du temps. Ces équations de contraintes sont généralement explicites et indépendantes ce qui rend leur résolution triviale.
2. Les coordonnées choisies ne prennent pas en compte (toutes) les limitations des mouvements relatifs, imposées aux corps par les articulations qui les relient. C'est typiquement le cas lors de l'utilisation de coordonnées absolues ou bien lorsque la structure du système contient des boucles cinématiques. Dans ce cas, les équations de contraintes sont la plupart du temps formulées de façon implicite et font intervenir plusieurs coordonnées dans une relation non-linéaire qui exprime une condition géométrique de fermeture.
3. Le comportement d'une partie du système est prescrite par des relations particulières. Ce dernier cas se produit lorsque l'utilisateur ne désire pas - ou ne peut pas - modéliser une partie du système à l'aide de corps et d'articulations, mais préfère fournir directement des équations décrivant le comportement de cette partie. Cette procédure est parfois utilisée pour la modélisation du contact roue-rail en dynamique des véhicules ferroviaires, par exemple [12]. Le cas des contraintes non-holonomes en est un cas particulier auquel nous ne nous sommes pas intéressé dans ce travail.

Les équations de contraintes constituent donc généralement un système d'équations algébriques non-linéaires qui restreint l'espace des valeurs possibles pour les coordonnées généralisées.

Diverses solutions peuvent être utilisées pour prendre en compte les conditions imposées par ces équations de contraintes.

1. La plus directe consiste à les résoudre explicitement, ainsi que leurs dérivées premières et secondes. Ainsi on détermine dans une première étape, les valeurs correctes des coordonnées généralisées à utiliser pour l'évaluation des équations du mouvement du système, dans une seconde étape. Cette procédure est la plus rigoureuse mais n'est toutefois pas la plus utilisée en raison du nombre d'opérations supplémentaires qu'elle implique. Cette approche peut conduire à reformuler les équations du mouvement sous une forme réduite, en se basant sur un jeu de coordonnées indépendantes, choisies en utilisant par exemple la technique du partitionnement des coordonnées [37].
2. La résolution explicite des seules dérivées secondes des équations de contraintes, conjointement aux équations du mouvement, permet de calculer les valeurs des dérivées secondes des coordonnées généralisées. Une double intégration des accélérations généralisées fournit alors des valeurs des coordonnées et des vitesses généralisées. Toutefois, l'accumulation possible d'erreur au niveau de l'intégration numérique peut entraîner une divergence des valeurs telles que les conditions imposées par les équations de contraintes ne soient plus satisfaites. Il convient alors de surveiller l'évolution des valeurs obtenues et d'éventuellement mettre en oeuvre une procédure de stabilisation [38] pour éviter la divergence. Cette approche peut conduire à une formulation réduite ou augmentée des équations du mouvement.
3. Les équations de contraintes peuvent également être intégrées directement et conjointement aux équations du mouvement. Elles ne sont alors pas résolues explicitement au niveau du modèle, mais implicitement par la méthode d'intégration. Le traitement direct de l'ensemble des équations différentielles du mouvement et algébriques de contraintes nécessite l'utilisation de méthode d'intégration particulières et adaptées au problème [39, 40, 41, 42, 43, 44]. Cette approche nécessite de reformuler l'ensemble des équations sous une forme implicite.

Hormis le cas des contraintes prescrivant la cinématique de certaines articulations, la résolution explicite des équations de contraintes n'est pas triviale.

Bien entendu, leurs dérivées premières (7) et secondes (8) sont linéaires en fonction, respectivement, des vitesses et des accélérations articulaires. Elles peuvent donc être résolues en utilisant, par exemple, des méthodes algébriques directes si leur nombre n'est pas trop important.

Cependant, ce n'est pas le cas des équations de contraintes (6) qui ne sont pas linéaires en fonction des coordonnées généralisées. De telles équations non-linéaires n'ont pas toujours de solution analytique. Leur résolution implique donc généralement de recourir à une méthode numérique. La plus utilisée est la

méthode itérative de Newton-Raphson. Cette méthode nécessite que certaines conditions soient remplies pour assurer sa bonne convergence.

Il faut que :

- les valeurs initiales des coordonnées soient assez proches de la solution,
- la Jacobienne des contraintes soit bien conditionnée.

Un bon partitionnement des coordonnées et l'intégration de l'ensemble des dérivées des coordonnées généralisées constitue une bonne assurance que ces conditions soient remplies lors de la simulation d'un système multicorps.

Néanmoins, pour les études purement cinématiques, il est parfois assez difficile d'obtenir un jeu de valeurs initiales satisfaisantes, ce qui constitue alors un handicap pour l'utilisation de la méthode itérative de Newton-Raphson.

Toutefois, très souvent, lorsque des coordonnées relatives sont utilisées pour décrire la configuration du système, les équations de contraintes proviennent essentiellement de la présence de boucles cinématiques. Ces contraintes expriment alors des conditions géométriques de fermeture de ces boucles. Des considérations géométriques, basées sur la nature et la disposition des articulations au sein des boucles, peuvent être faites de manière à obtenir des solutions analytiques à ces équations de contraintes relatives aux boucles cinématiques [45, 46, 47]. Cette méthode constitue la solution la plus performante pour le traitement et la résolution des équations de contraintes provenant des boucles cinématiques.

Formulations

Nous avons vu ci-dessus que le comportement d'un système multicorps soumis à des contraintes est régi par les n équations du mouvement (5) et les m équations de contraintes (6). Dans ces équations, on dénombre $n+m$ inconnues à savoir les n accélérations généralisées $\ddot{q}$ et les m multiplicateurs de Lagrange λ.

On peut alors formuler un système de $n+m$ équations permettant de calculer les $n+m$ inconnues. Cette première approche conduit

- soit à *intégrer directement* les équations (5) et (6) à l'aide d'une méthode d'intégration numérique appropriée au traitement d'un système d'équations différentielles et algébriques,
- soit à générer une *formulation augmentée*[11], qui permet l'intégration numérique à l'aide de méthodes classiques pour équations différentielles ordinaires.

Toutefois, nous savons que les m équations de contraintes constituent des relations entre les n coordonnées généralisées. En supposant que ces m relations sont indépendantes, on peut en déduire que le nombre f de degrés de liberté du système est $f = n - m$.

[11] en anglais : "Descriptor form"

Il est donc possible d'utiliser un jeu minimal de $n-m$ coordonnées, pour décrire la configuration du système et de formuler un système de $n-m$ équations pour calculer ces $n-m$ coordonnées indépendantes. Cette seconde approche conduit à générer une *formulation réduite*[12].

Ces deux approches génériques peuvent conduire à un grand nombre de formulations particulières. On trouve dans [9] plus de dix manières différentes de formuler les équations du mouvement en vue de leur intégration numérique.

Les variations concernent en autre

- l'utilisation d'un jeu de variables dépendantes ou indépendantes,
- le traitement des équations de contraintes,
- le calcul ou l'élimination des forces de contraintes,
- le caractère purement différentiel ou différentiel/algébrique du système d'équations,
- le nombre d'équations présentes dans la formulation finale.

Hormis le cas de l'intégration directe du système différentiel - algébrique défini par les équations (5) et (6), on peut classer les différentes formulations qui produisent un système d'équations purement différentielles, en deux catégories.

Formulations Augmentées Afin de former un système de $n+m$ équations différentielles en les $n+m$ inconnues que constitue l'ensemble des $\ddot{q}$ et des λ, on utilise les n équations du mouvement (5). Pour les m équations complémentaires, on n'utilise pas les équations de contraintes (6) elles-mêmes, mais leurs dérivées secondes (8) où interviennent les inconnues $\ddot{q}$.

En posant $g(q,\dot{q}) = \dot{J}\dot{q}$ on peut écrire les équations (8) comme ceci

$$\ddot{h}(q,\dot{q},\ddot{q},t) = J(q)\ddot{q} + g(q,\dot{q})$$

Le système de $n+m$ équations différentielles peut alors être formulé sous la forme matricielle suivante :

$$\begin{pmatrix} M(q) & -J^T(q) \\ J(q) & 0 \end{pmatrix} \begin{pmatrix} \ddot{q} \\ \lambda \end{pmatrix} = \begin{pmatrix} Q - c(q,q,F_{ext},L_{ext},g) \\ -g(q,\dot{q}) \end{pmatrix} \qquad (11)$$

La résolution de ce système d'équations permet de calculer $\ddot{q}$ et λ. Cette résolution[13] peut devenir assez coûteuse lorsque les nombres de coordonnées utilisées et de contraintes sont élevés, en raison de la complexité $O(N^3)$ des méthodes de résolution directe.

En posant $y = \binom{q}{\dot{q}}$ on peut alors exprimer le calcul des accélérations sous la forme explicite $\dot{y} = f(y,t)$ et procéder à son intégration numérique à l'aide d'une méthode classique.

Toutefois cette méthode conduit généralement à une accumulation d'erreurs due au caractère naturellement instable des équations (8).

[12] en anglais : "State Space form"

[13] Celle-ci peut être délicate car la matrice $\left(\begin{smallmatrix} M & -J^T \\ J & 0 \end{smallmatrix}\right)$ peut ne pas être régulière

Il est alors nécessaire de recourir à une méthode de *stabilisation* [38, 48] ou de *pénalité* [49, 9, 11] pour éviter la divergence du processus d'intégration numérique.

Formulations Réduites La formulation réduite consiste à constituer un système de $n - m$ équations qui permet de calculer $n - m$ coordonnées indépendantes correspondant au nombre de degrés de liberté du système.

Les $n - m$ coordonnées indépendantes peuvent être

1. un ensemble distinct de coordonnées z,
2. un sous ensemble des coordonnées généralisées.

Le premier cas peut se présenter lorsqu'on utilise des coordonnées absolues ou naturelles pour la modélisation du système. Le jeu de coordonnées indépendantes pourrait alors être un ensemble de coordonnées articulaires. C'est souvent le cas lors de l'utilisation des formalismes $O(N)$ adaptés au calcul parallèle.

Dans le second cas, la technique du partitionnement des coordonnées [37, 50, 51] consiste à répartir les coordonnées généralisées q en deux sous ensembles : d'une part $n-m$ coordonnées indépendantes q_u et d'autres part m coordonnées dépendantes q_v.

Le partitionnement peut être effectué sur base de l'analyse numérique de la matrice Jacobienne des contraintes, au moyen

- d'une triangularisation de Gauss avec pivotement partiel ou total,
- d'une factorisation QR,
- d'une décomposition en valeurs singulières (SVD).

La première solution est la plus souvent utilisée en raison de son coût calcul réduit.

La procédure de réduction associée au partitionnement des coordonnées consiste alors à exprimer les m accélérations dépendantes $\ddot{q}_v$ en fonction des accélérations indépendantes $\ddot{q}_u$, en utilisant les m équations (8) dérivées secondes des équations de contraintes.

On peut ensuite exprimer les m multiplicateurs de Lagrange λ à l'aide des m accélérations dépendantes $\ddot{q}_v$, et les éliminer des équations du mouvement (5). Lors de cette opération, on transforme les n équations du mouvement (5) en un système de $n - m$ équations réduites que nous pouvons écrire comme ceci :

$$\mathcal{M}_r \ \ddot{q}_u + \mathcal{F}_r = 0 \tag{12}$$

D'autres techniques de réduction équivalentes utilisent une matrice de projection qui est le complément orthogonal de la matrice Jacobienne des contraintes [52, 53, 9]. Ce complément orthogonal peut être calculé par exemple, en utilisant les méthodes de factorisation QR ou de décomposition en valeurs singulières de la matrice Jacobienne des contraintes.

Les $n - m$ inconnues, que sont les accélérations indépendantes $\ddot{q}_u$, peuvent alors être obtenues par résolution directe du système d'équations réduites .

La résolution du système d'équations (12) implique un nombre d'opérations nettement moins élevé que dans le cas de la formulation augmentée (11) lorsque le nombre m de contraintes est important. En effet la résolution d'un système de N équations linéaires par une méthode directe est d'une complexité $O(N^3)$. La différence entre le coût de résolution de la forme augmentée et de la forme réduite est alors de $O(m^3 + n^2)$ en faveur de la forme réduite.

En posant $y = \binom{q_u}{\dot{q}_u}$, on peut à nouveau exprimer le calcul des accélérations indépendantes $\ddot{q}_u$ sous la forme explicite $\dot{y} = f(y, t)$ et les intégrer à l'aide d'une méthode numérique classique. Le problème d'instabilité lié à l'intégration de l'équation (8) ne se pose plus ici.

Les valeurs des coordonnées et des vitesses dépendantes peuvent être obtenues par intégration des vitesses et des accélérations dépendantes, respectivement. Toutefois, pour une meilleure précision, leurs valeurs peuvent être obtenues par résolution des équations de contraintes (6) ainsi que leur dérivées premières (7), ce qui implique alors un supplément de calculs d'autant plus important que le nombre de contraintes est élevé.

Génération des modèles

Parmi les logiciels d'analyse de système multicorps, on peut distinguer deux groupes selon l'approche *symbolique* ou *numérique* utilisée pour la génération et le calcul des équations du mouvement. La figure 2 illustre le déroulement des opérations d'analyse selon chaque approche. Les cadres en pointillés correspondent à un processus d'analyse numérique typique, à savoir l'intégration numérique des équations du mouvement du système.

Les programmes *symboliques* n'effectuent, a priori, aucun calculs numériques. Ils ne manipulent pas des nombres, mais des symboles qui correspondent aux coordonnées et aux paramètres du systèmes, pour produire les expressions analytiques des équations. Ces équations sont généralement exportées sous forme de fichiers qui contiennent des routines de calcul écrites en un langage de programmation standard tel que FORTRAN, C, etc. Ces routines peuvent alors être importées dans un environnement de calcul numérique pour être évaluées dans le cadre de l'analyse numérique du système. Les logiciels symboliques n'offrent pas nécessairement des possibilités de calcul numérique.

Les avantages principaux de l'approche symbolique sont :

- optimisation des expressions analytiques des équations grâce aux manipulations symboliques,
- grande vitesse d'évaluation des équations générées sous forme analytique.
- facilité d'utilisation des équations dans la plupart des environnements de calcul numérique,

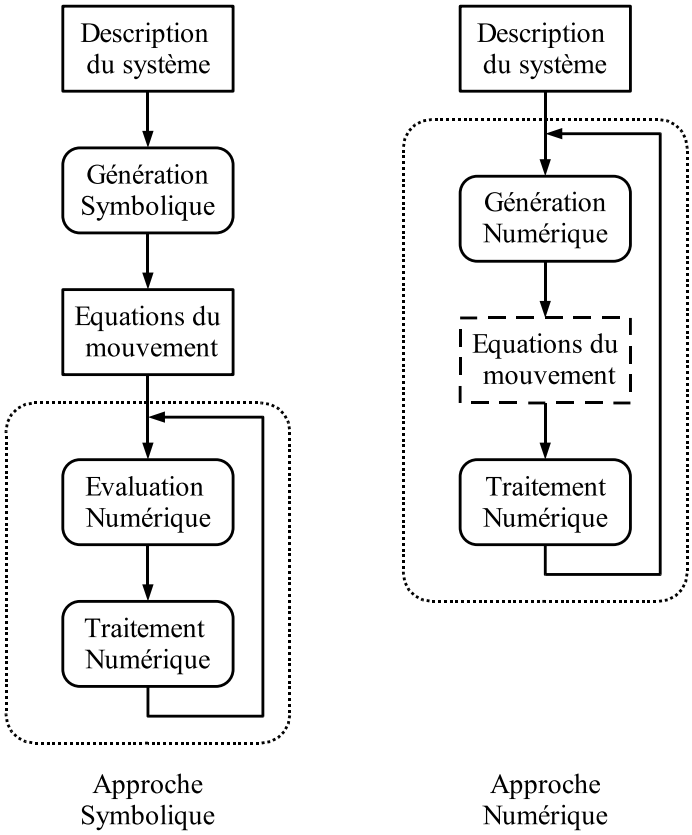


FIG. 2 – Comparaison des approches symbolique et numérique

Les programmes *numériques* ne génèrent pas les expressions analytiques des équations. Les algorithmes des formalismes sont directement appliqués aux valeurs des coordonnées et des paramètres du système pour calculer numériquement les résultats des équations du mouvement. Les expressions analytiques des équations n'existent donc pas au sein du programme numérique, ce que nous illustrons par le cadre en trait discontinu sur la figure 2. En revanche, les programmes numériques offrent toujours la possibilité d'effectuer directement les diverses analyses numériques cinématiques et dynamiques applicables aux systèmes multicorps. Toutefois, on notera que les équations sont régénérées à chaque évaluation, ce qui prend plus de temps que l'exécution d'une sous-routine qui implémente la forme analytique des équations.

Les avantages principaux de l'approche numérique sont :

- adaptation aux changements de configuration qui peuvent survenir dans le système,
- utilisation d'un seul logiciel pour la modélisation et l'analyse numérique du système.

Exécution des modèles en parallèle

Depuis plus de quinze années, des chercheurs ont proposé des solutions pour permettre de réduire le temps d'exécution des modèles de systèmes multicorps. Les objectifs sont récurrents et reflètent la volonté :

- d'étudier des systèmes de plus en plus grands et complexes,
- d'atteindre des performances de simulation temps réel pour
  - le développement de systèmes de contrôle avancé,
  - la simulation en connexion avec le système réel ou l'être humain,
- d'atteindre de performances de simulation meilleures que le temps réel pour des traitements numériques gourmands en nombre de simulations tels que l'optimisation.

Parmi les solutions habituelles, on envisage

- l'utilisation de formalismes plus efficaces tels que les formalismes O(N),
- l'utilisation d'architectures de calcul parallèle,
- une combinaison optimale des deux premières solutions par le biais du développement de formalismes modifiés, voire dédicacés aux architectures parallèles utilisées.

Architectures Au début des années 1990, divers travaux [54, 55] relatent l'utilisation de plusieurs *transputers* pour obtenir une architecture de calcul parallèle. Un *transputer* est un microprocesseur particulier contenant une unité de calcul en virgule flottante, de la mémoire et des dispositifs de communication, permettant de communiquer avec d'autres microprocesseurs semblables inter connectés en réseau.

Certains chercheurs [56, 33] ont conçu des architectures dédicacées à partir d'éléments de base, tels que des processeurs, des mémoires et des dispositifs d'interconnexion.

L'avènement, à la fin des années 1990, des composants reconfigurables de grande capacité tels que les FPGA ouvre des perspectives intéressantes au développement de prototypes d'architectures parallèles [57, 58, 59].

D'autres développements [60, 61] se basent sur l'utilisation d'ordinateurs parallèles classiques, souvent à mémoire partagée et équipés d'un nombre limité de processeurs, pour l'exécution de modèles parallèles.

Les modèles parallèles sont alors obtenus en utilisant des formalismes récursifs classiques modifiés pour obtenir une formulation parallèle adaptée à l'architecture utilisée. Toutefois, avant [34], aucun formalisme n'était réellement

dédicacé au calcul parallèle, ce qui n'empêchait pas d'obtenir une réduction du temps de calcul des modèles en les distribuant sur plusieurs processeurs.

Le recours au calcul sur ordinateur parallèle est donc une solution évidente [62, 63] pour la simulation des grands systèmes. Dans la mesure où les meilleurs formalismes ont une complexité $O(N)$, c'est la seule solution qui permette de limiter l'augmentation du temps calcul nécessaire pour étudier des systèmes de plus en plus grands. Les plus grands systèmes simulés actuellement sont les molécules dont on simule la dynamique en utilisant un algorithme $O(N)$ dédicacé au calcul parallèle [29]. Cet algorithme permet d'atteindre une complexité $O(logN)$ en utilisant $O(N)$ processeurs. Les ordinateurs massivement parallèles contiennent actuellement plusieurs milliers de processeurs.

Parallélisme Afin de pouvoir exploiter une architecture de calcul parallèle pour l'analyse d'un système, il est nécessaire de pouvoir mettre en évidence un certain parallélisme au niveau du modèle de celui-ci. Ce parallélisme peut avoir différentes origines :

- le parallélisme algébrique,
- le parallélisme algorithmique,
- le parallélisme spatial ou topologique.

Le parallélisme algébrique correspond à la possibilité de décomposer une opération algébrique en sous-groupes d'opérations arithmétiques. Par exemple, le résultat du produit de deux matrices peut être calculé élément par élément, ou ligne par ligne. Ce type de parallélisme est généralement qualifié de parallélisme à *grain fin*, lorsque la décomposition est poussée jusqu'à obtenir un grand nombre de petits groupes d'opérations arithmétiques.

Le parallélisme algorithmique provient de la possibilité de séparer des opérations indépendantes au niveau de l'implémentation de l'algorithme. Tous les algorithmes $O(N)$ présentent un certain taux de parallélisme, mais seuls les formalismes parallèles [34] présentent un taux de parallélisme croissant linéairement avec la complexité du système. Les autres formalismes $O(N)$ ne présentent qu'un taux de parallélisme constant, ce qui ne permet pas d'exploiter un nombre croissant de processeurs lorsque la taille du système augmente.

Le parallélisme spatial ou topologique, provient de la structure du système. Si le système présente différentes parties indépendantes, comme dans le cas des branches d'une structure arborescente ou dans le cas d'un système composé de sous-systèmes, les calculs relatifs à ces différentes parties peuvent être exécutées indépendamment les unes des autres [64]. Dans la plupart des systèmes multicorps classiques, l'augmentation de la taille est souvent due à une augmentation des ramifications de la structure et donc on bénéficie d'une augmentation du parallélisme spatial.

Dans certains cas, on exploite même le parallélisme au niveau de la méthode d'intégration. Par exemple, dans [65], une méthode d'intégration implicite est préconisée. Une telle méthode nécessite l'évaluation d'une matrice Jacobienne,

qui est évaluée numériquement, colonne par colonne, en appelant autant de fois que nécessaire le modèle du système, obtenu sous forme implicite. Chaque colonnes de la Jacobienne est calculée en parallèle. Cette pratique contribue grandement au rendement du calcul parallèle, bien que ce type de parallélisme soit externe au modèle.

Parallélisation Afin de tirer profit des architecture parallèles pour la simulation des systèmes multicorps, il est nécessaire de distribuer le calcul du modèle du système. Dans la littérature, on constate que dans la grande majorité des applications où le calcul parallèle est envisagé, le modèle du système est obtenu par l'approche numérique. C'est donc l'implémentation de l'algorithme de calcul du modèle qui fait l'objet de la parallélisation. L'implémentation parallèle de l'algorithme de calcul est alors la plus souvent dépendante de l'architecture visée et parfois même du système lui-même [60], ce qui permet alors d'obtenir de très bon résultats en terme d'efficacité parallèle. Dans [64], c'est le programme dédicacé à la simulation d'un véhicule qui est parallélisé manuellement sur base de modèles générés symboliquement pour différents sous systèmes, correspondant aux quatre quart d'une Jeep Iltis, en l'occurence.

Avant le développement de l'algorithme $O(N)$ dédicacé au calcul parallèle [34, 35] pour lequel une méthode systématique de distribution des calculs a été proposée [36], il n'y avait aucune procédure de parallélisation générique.

Les procédures de parallélisation des algorithmes qui ne sont pas dédicacés au calcul parallèle, exploitent toujours plus ou moins le parallélisme algorithmique et le parallélisme spatial, mais sont toujours relativement limitées à une classe de systèmes.

Nous pouvons toutefois mentionner un exemple de parallélisation de modèles générés par l'approche symbolique, dans lequel la procédure de parallélisation est plus générique. Dans [54] les opérations arithmétiques du modèle d'un robot manipulateur série sont analysées et classées selon leurs interdépendances. Elles sont ensuite ordonnancées en vue d'être regroupées et distribuées entre quelques processeurs et une efficacité parallèle quasi maximale est atteinte en utilisant un réseau de quatre *transputers* avec une interconnexion complète.

Les logiciels d'analyse de systèmes multicorps

Il existe à l'heure actuelle une multitude d'outils qui permettent d'étudier les systèmes multicorps. Nous n'envisageons pas d'en établir ici une liste exhaustive mais nous citons les plus connus et quelques-unes de leurs caractéristiques, celles-ci n'étant pas toujours très accessibles.

Les logiciels commerciaux

Les logiciels commerciaux ont tous en commun un environnement graphique intégré permettant à la fois la modélisation graphique et la paramétrisation des modèles, l'analyse numérique et l'exploitation des résultats sous forme de graphes et d'animations 3D.

En général, ces logiciels se basent sur une approche numérique. Divers modules optionnels sont généralement disponibles afin d'offrir des fonctionnalités complémentaires telles que

- la modélisation de corps flexibles,
- l'importation de géométries provenant de logiciels de CAO,
- l'interconnection avec MATLAB/SIMULINK,
- l'utilisation de routines écrites par utilisateur,
- etc.

Ces logiciels permettent en principe de modéliser tous les types de système multicorps et proposent généralement des librairies de composants et de sous-systèmes utilisés plus particulièrement dans certains domaines tels que l'automobile, le ferroviaire ou encore l'aérospatiale.

On rencontre généralemetn ces outils dans l'industrie, et dans certaines écoles d'ingénieurs. Il leur est parfois reproché leur performance en terme de temps calcul, leur coût important et l'investissement en temps nécessaire à leur utilisation et à leur maîtrise.

Voici les principaux logiciels de ce type :

- **SIMPACK** [66] : développé initialement par le centre de recherche en aérospaciale Allemand (DLR), il est commercialisé depuis près de dix ans par la société Intec. Ce logiciel utilise des coordonnées relatives et implémente un formalisme récursif d'une complexité d'ordre N. Il semble également capable de produire des modèles sous forme symbolique, toutefois les formulations proposées ne sont ni les plus performantes, ni les plus pratiques à utiliser. En outre, cette fonctionnalité n'est disponible que moyennant l'achat d'un module complémentaire.
- **ADAMS** [67] : commercialisé par MSC Software, c'est un des logiciels les plus répandus dans l'industrie automobile. Il utilise des coordonnées absolues et est particulièrement orienté sur l'utilisation de composants et de sous-systèmes pour la création de nouveaux modèles non disponibles en librairie. Selon ses utilisateurs, son principal inconvénient semble être une certaine lenteur d'exécution.
- **Virtual.lab MOTION** [68] : évolution du logiciel DADS initialement développé par le Pr. E. J. Haug, il est actuellement commercialisé par la société LMS. Celui-ci utilise également des coordonnées absolues. Il n'est pas non plus particulièrement rapide en simulation, mais il a l'avantage d'être intégré dans une suite logicielle offrant des possibilités d'analyses très diverses telles que de l'analyse de structure, des analyses acoustiques,

etc., ce qui en fait un outil très intéressant et très productif pour l'étude complète d'un système, et ce plus particulièrement dans le domaine de l'automobile.

- **CARSIM** [69] : Ce dernier logiciel est dédicacé à l'étude des voitures et camions. Il s'agit en fait d'un environnement de simulation et d'analyse, plus que d'un outil de modélisation à part entière. En effet, il utilise des modèles paramétrables générés préalablement par un programme de génération symbolique appelé *AutoSim*. Cette caractéristique lui permet d'obtenir des performances de simulation largement suffisantes pour viser les applications temps réel. Toutefois, le générateur ne semble pas être disponible pour l'utilisateur, qui ne peut apparemment pas simuler d'autres systèmes que ceux qui lui sont proposés.

Le logiciel de calcul numérique MATLAB contient depuis quelques années un outil de modélisation de systèmes mécaniques appelé SIMMECHANICS. [70, 71]. Cet outil utilise une approche numérique. Le système est décrit à l'aide de coordonnées articulaires et un formalisme d'une complexité d'ordre N est utilisé pour générer les équations. Les contraintes sont traitées à l'aide d'une méthode de projection. Dans le cas de l'utilisation des modèles pour des applications temps réel, les contraintes ne sont plus résolues explicitement, mais une méthode de stabilisation est alors employée.

Les outils logiciels de recherche

Divers centres de recherches ou universités développent et utilisent encore actuellement des codes propres de modélisation et de simulation dédicacés aux systèmes multicorps.

Certains de ces outils utilisent l'approche symbolique pour générer les équations requises à l'analyse des systèmes multicorps.

Logiciels basés sur un outil de calcul symbolique

Les deux logiciels suivants se basent sur le logiciel de calcul symbolique Maple [72] pour générer les équations, et sont tous deux développés au Canada.

- **DYNAFLEX** [53, 6, 73] Ce logiciel est développé à l'Université de Waterloo (Ontario) sous la supervision du Pr. John McPhee. Il se base sur la théorie des graphes linéaires pour la description des systèmes et utilise la méthode des travaux virtuels et des coordonnées relatives ou indirectes pour la génération des équations. Il peut générer une formulation réduite des équations pour les systèmes contraints, mais les équations de contraintes ne sont pas résolues en position et en vitesses à l'heure de la rédaction de ce texte. Il permet de traiter aisément des systèmes électro-mécaniques et d'utiliser des sous-systèmes pour composer un système de complexité moyenne. Il s'agit essentiellement d'un générateur

symbolique d'équations. Celles-ci peuvent ensuite être utilisées directement dans Maple ou exportées en langage C ou vers MATLAB.

- **SYMOFROS** [74, 75] Ce logiciel est développé sous la supervision de Jean-Claude Piedboeuf à l'Agence Spatiale Canadienne. Il se base sur le principe des puissances potentielles et les coordonnées relatives pour la description des systèmes et la génération des équations. Celles-ci sont automatiquement générées en C pour être utilisées dans l'environnement SIMULINK. SYMOFROS est un outil complet de modélisation et de simulation destinés principalement au développement de systèmes robotiques de complexité moyenne.
- **SYMKIN** : Il s'agit d'un outil destiné à la génération des équations cinématiques et dynamiques de systèmes contenant des boucles cinématiques. Il permet d'obtenir la solution des équations de contraintes relatives aux boucles sous forme analytique. Il utilise pour cela les fonctionnalités du logiciel de calcul symbolique Mathematica. Il a été développé par la même équipe que le logiciel numérique MOBILE décrit ci-dessous.

Logiciels symboliques indépendants

Les outils décrits ci-dessous sont généralement plus anciens. Leur développement a commencé dans les années 80 et ils offraient au début des années 90 des performances en terme de génération qu'il était impossible d'atteindre avec les logiciels de calcul symbolique commerciaux, ce qui est encore vrai actuellement. Il permettaient déjà d'obtenir des simulations en temps réel, pour des modèles très simples, ce qui était également impossible en utilisant les approches numériques.

Il n'ont cependant pas tous été mis à jour récemment, sauf ROBOTRAN.

- **AUTOSIM** [76] : Ce logiciel a été développé en LISP au début des années 80 pour la génération symbolique d'équations du mouvement. Il utilise la formulation de Kane[77], permet de traiter les systèmes avec boucles et les contraintes. Il semble capable de calculer des coordonnées dépendantes en résolvant les contraintes. Il permet également de déterminer les conditions d'équilibres d'un système. Il a été utilisé pour le développement du logiciel numérique Carsim présenté ci-dessus.
- **SD/FAST** [78] : Ce logiciel a été développé par Michael A. Sherman et Dan E. Rosenthal de Symbolic Dynamics Inc. Il génère les équations de la dynamique directe en se basant également sur la formulation de Kane et également sur une formulation [79] de complexité en ordre N. Il peut produire les équations en langage C et FORTRAN. Il ne permet pas de modéliser des corps flexibles ni de résoudre des contraintes. Une formulation augmentée avec méthode de stabilisation des contraintes [38] est proposée pour les systèmes contenant des boucles cinématiques.
- **NEWEUL** [80, 81] : Ce logiciel est développé à l'université de Stuttgart

par l'équipe du Pr. W. Schiehlen. Ce logiciel se base sur le formalisme de Newton-Euler et permet de décrire le système en utilisant des coordonnées relatives. Il permet de traiter les contraintes et les corps élastiques. Il est accompagné d'un module de simulation qui permet d'exploiter les équations générées symboliquement pour effectuer des analyses numériques.

- **ROBOTRAN** : Ce logiciel est développé dans notre laboratoire. Il est présenté plus en détail ci-après. Il implémente des formalismes récursifs de type Newton-Euler ou de complexité en ordre N.

Caractéristiques des générateurs symboliques

Les programmes de génération symbolique basés sur des logiciels de calcul symbolique commerciaux, tels que Maple, bénéficient évidemment de toute la puissance et des nombreuses possibilités de manipulation symbolique de ces logiciels. Cela facilite énormément le développement et le test de nouveaux formalismes ou de nouvelles techniques de modélisation. Ces outils sont donc particulièrement appropriés à la recherche dans le contexte académique.

Toutefois, il faut réaliser que ces logiciels ne sont pas développés pour exploiter le caractère récursif généralement présent dans les équations obtenues lors de la modélisation des systèmes multicorps. En pratique, les développeurs et les utilisateurs de ces programmes de générations symboliques dédicacés aux systèmes multicorps sont souvent confrontés aux limitations de ces logiciels symboliques en ce qui concerne :

- d'une part, leur capacité de traitement[14] des nombreuses d'équations obtenues lors de la modélisation des systèmes de grande taille[15]. Des approches alternatives sont alors envisagées pour pallier cet inconvénient comme par exemple, la modélisation du système par parties et l'assemblage des modèles symboliques des sous-systèmes [73].
- d'autre part, la possibilité d'accéder librement à la structure de données utilisée pour stocker les équations, ce qui peut constituer un source de difficultés pour implémenter certaines analyses ou traitements qui ne seraient pas disponibles au niveau de l'interface de programmation du logiciel. Par exemple, il n'est pas évident de pouvoir associer des attributs supplémentaires aux expressions ou aux équations ou de pouvoir effectuer des manipulations directement sur les équations ou sur les expressions symboliques.

Le développement de générateurs symboliques indépendants implique l'élaboration de librairies de fonctions pour la manipulation et la gestion des expressions et des équations symboliques. Toutefois cela permet d'obtenir un outil dédicacé à la manipulation symbolique des équations relatives à la modélisation des systèmes multicorps, et ainsi d'atteindre de meilleures performances

[14] en un temps raisonnable et sans dépasser la quantité de mémoire vive disponible...
[15] qui contient plus d'une cinquantaine d'articulations

en terme de :

- quantité de mémoire nécessaire,
- vitesse de génération,
- parfois même d'optimisation des équations,

que dans le cas de l'utilisation de logiciels symboliques standards. Ceci explique la capacité des générateurs symboliques indépendants à traiter facilement des systèmes de plus grande taille[16], à savoir des systèmes réalistes tels que des véhicules complexes par exemple, alors que les outils tels que DYNAFLEX et SYMOFROS peuvent nécessiter un temps et une quantité de mémoire considérable pour générer les modèles de ces systèmes, lorsqu'ils y arrivent.

Modèles symboliques Tous ces outils ont en commun la génération des expressions analytiques des équations du mouvement par manipulation de symboles qui représentent les coordonnées et les paramètres du système. Ces expressions analytiques doivent être neutres, c'est-à-dire que leur évaluation ne doit pas nécessiter l'utilisation de librairies spécifiques. Ce type d'expressions symboliques est à distinguer d'un programme informatique qui contiendrait un ensemble d'appel à des sous routines de calcul [82, 8].

En pratique, il est toutefois assez rare de pouvoir obtenir un modèle composé uniquement d'expressions analytiques. On trouvera souvent parmi les équations du mouvement quelques fonctions dont les expressions analytiques ne sont pas explicitées, ce qui donne au modèle symbolique un caractère plus générique. Ce sera par exemple le cas des fonctions de calcul des interactions entre le système multicorps et son environnement. Toutefois, dans le cas des programmes basés sur l'utilisation d'un logiciel symbolique générique, les expressions analytiques de ces fonctions peuvent être introduites dans le logiciel et substituées aux appels de fonction dans le modèle du système multicorps pour obtenir un modèle purement analytique.

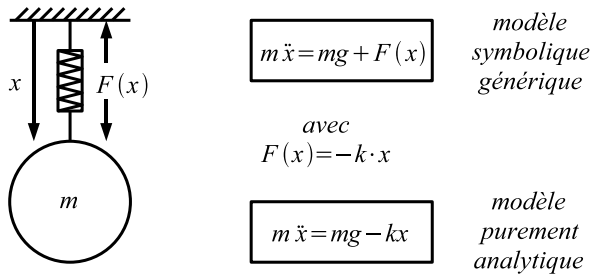


FIG. 3 – Appel de fonction dans les modèles symboliques

[16] plusieurs dizaines voire plus d'une centaine de coordonnées généralisées

La figure 3 illustre cette différence à l'aide d'un système simple. Nous considérons une masse suspendue par un élément qui exerce une force fonction de la distance entre la masse et un support. En considérant la force comme une fonction de la distance, on obtient un modèle symbolique générique. En particularisant au cas où la fonction correspond au modèle d'un ressort linéaire, on obtient un modèle complètement analytique mais qui perd son caractère générique.

Autres logiciels multicorps

- **MOBILE** [83, 84] développé par A. Kecskeméthy, M. Hiller et Ch. Woernle, à l'université de technologie de Gratz, en Autriche et à l'université de Duisburg en Allemagne.
- **FASIM** [85] développé au sein du laboratoire de mécatronique du Pr. M. Hiller en collaboration avec la société BOSCH.
- **DynaMechs** [86] développé initialement par Scott McMillan depuis 1991 à la *Ohio State University, Department of Electrical Engineering.*
- **ODE** [87] développé par Russell Smith[17] depuis 2000.

Ces logiciels numériques sont en fait des librairies de routines écrites en langage C/C++ et qui permettent la description, la modélisation, la simulation, le contrôle et l'animation graphique de systèmes multicorps. Ils se basent tous sur la représentation des systèmes à l'aide de coordonnées articulaires, et sont capables de gérer des boucles cinématiques.

Ils semblent tous offrir des performances de simulation temps réel pour des systèmes robotiques ou des véhicules simples, contrairement aux logiciels numériques commerciaux. Toutefois, ces performances sont généralement encore inférieures à celles obtenues en utilisant des modèles générés symboliquement.

L'inconvénient de ces outils est qu'ils nécessitent un travail de programmation relativement important pour modéliser les systèmes, ce qui peut constituer une limite en terme de productivité. Leur utilisation et leur maîtrise nécessite un apprentissage fastidieux de leur interface de programmation et une bonne connaissance du langage C++. En outre, ils ne permettent pas d'exporter les modèles sous formes de routines indépendantes pour une utilisation dans d'autres environnements de calcul.

Commentaires

A notre connaissance, aucun de ces outils ne propose de fonctionnalités relatives à la parallélisation génériques des modèles. Dans les descriptifs de ces divers logiciels, on ne trouve pas d'informations relatives à leur utilisation sur des systèmes de calcul parallèle.

[17] Ph.D. in July 1998, in the department of Electrical and Electronic Engineering at the University of Auckland, New Zealand

Les seules références à l'utilisation du calcul parallèle se trouvent dans la littérature scientifique relative à certains développements spécifiques effectués au sein de centres de recherches ou d'universités.

Le logiciel de génération symbolique ROBOTRAN

Le logiciel ROBOTRAN [88, 89, 90] est un programme indépendant dédicacé à la génération symbolique de modèles de systèmes multicorps. Il est le fruit de plusieurs années de recherches et de développements menées au sein du Département de Mécanique de l'Université catholique de Louvain. Différents travaux ont bénéficié de l'utilisation de cet outil [91, 92, 93, 64] et certains ont contribué à son développement [20, 12, 17, 21].

Une description détaillée de ce logiciel est fournie dans [12] qui constitue la contribution principale au développement de ROBOTRAN.

Formalismes et formulations disponibles

ROBOTRAN utilise essentiellement les coordonnées relatives. Les formalismes récursifs Newton-Euler semi-explicite [17] et explicite $O(N)$ [26] sont disponibles pour la génération de modèles permettant de résoudre le problème dynamique direct pour des systèmes à topologie arborescentes et composés de corps rigides. Des formulations semi explicites et implicites, basées sur l'utilisation du formalisme Newton-Euler récursif, sont proposées pour l'étude de systèmes contenant des poutres flexibles.

Des formalismes Newton-Euler basés sur l'utilisation de paramètres barycentriques sont également proposés pour générer des modèles adéquats pour la résolution des problèmes dynamiques directs et inverses ainsi que pour l'identification des paramètres dynamiques pour des systèmes arborescents [21].

Divers modèles cinématiques directs peuvent également être générés symboliquement, tels que les équations de calcul de la position, la vitesse, l'orientation, etc. relatives à un repère associé à n'importe quel corps du système.

En ce qui concerne les systèmes contenant des boucles cinématiques, il est possible de générer symboliquement les expressions des contraintes de fermeture des boucles, selon la technique de coupure présentée dans [12], ainsi que les dérivées premières et secondes de ces contraintes.

Il n'était toutefois pas possible d'obtenir des modèles symboliques *complets* pour l'étude de la dynamique de systèmes contenant des boucles cinématiques, avant la réalisation de ce travail. L'approche utilisée pour l'étude de la dynamique de tels systèmes consistait alors généralement à assembler des parties de modèles générés symboliquement et à les compléter par une approche numérique au sein du logiciel d'analyse MBSOFT [94] basé sur l'environnement de calcul MATLAB.

Génération symbolique

ROBOTRAN dispose d'une librairie mathématique symbolique pour la création d'expressions complexes sur base de symboles. Cette librairie contient les opérateurs arithmétiques de base, *addition*, *multiplication*, etc. ainsi que des fonctions trigonométriques telles que *sinus* et *cosinus*.

On peut donc créer aisément une expression telle que $L_1\,cos(\alpha) + L_2\,cos(\gamma)$ en combinant les symboles L_1, L_2, α et γ à l'aide de ces opérations.

Lorsqu'on compose une expression symbolique dans ROBOTRAN, des tests sont effectués de façon récursive sur les opérandes lors de chaque utilisation d'un opérateur, qu'il soit arithmétique ou trigonométrique. Ainsi de nombreuses simplifications, telles que par exemple,

- les additions avec 0 : $a + 0 \mapsto a$,
- les multiplications par 0 : $a \cdot 0 \mapsto 0$ ou par 1 : $a \cdot 1 \mapsto a$,
- les simplifications arithmétiques telles que $(a - b) + (a + b) \mapsto 2 \cdot a$,
- les simplifications trigonométriques telles que
  $cos(\alpha)\,sin(\beta) + cos(\beta)\,sin(\alpha) \mapsto sin(\alpha + \beta)$

sont effectuées directement lors de la création des expressions.

Chaque expression peut également être affectée à une nouvelle variable, dite "auxiliaire", ce qui facilite la programmation des algorithmes de génération.

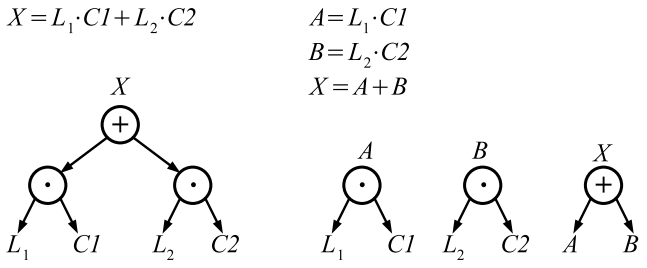


FIG. 4 – Expressions symboliques

Une différence essentielle entre ROBOTRAN et les logiciels génériques de calcul symbolique réside dans la manière de gérer les expressions symboliques. En effet, l'affectation d'une expression à une nouvelle variable peut se faire par le biais de la création d'une équation[18] qui est alors stockée dans une liste. La nouvelle variable ainsi définie n'est désormais plus considérée comme une expression complexe, mais comme un symbole élémentaire. Il s'agit là d'une clé permettant d'implémenter de manière performante des formalismes récursifs dans un environnement de calcul symbolique. Ceci permet de segmenter

[18] au sens *assignation* du membre de droite au membre de gauche

les expressions complexes en expressions intermédiaires plus simples comme le montre la figure 4, ce qui a pour effet d'alléger considérablement leur gestion.

Dans le cas de l'utilisation des fonctions trigonométriques, des équations intermédiaires sont systématiquement créées pour éviter de recalculer ces fonctions avec le même argument, à chaque fois qu'elles sont utilisées dans les équations. La formation des expressions symboliques A et B suivantes :

$$\left.\begin{array}{l} A = L_1\, cos(\alpha) \\ B = L_2\, cos(\gamma) \end{array}\right\} \quad \longmapsto \quad \left\{\begin{array}{l} c\alpha = cos(\alpha) \\ c\gamma = cos(\gamma) \\ A = L_1\, c\alpha \\ B = L_2\, c\gamma \end{array}\right.$$

provoqueront automatiquement la création des équations qui définissent des variables *trigonométriques* intermédiaires $c\alpha$ et $c\gamma$. Ainsi, les occurrences ultérieures de $cos(\alpha)$ et $cos(\gamma)$ seront automatiquement remplacées par $c\alpha$ et $c\gamma$ respectivement. Ces variables trigonométriques ont une *nature* particulière qui permet à ROBOTRAN de les identifier au sein d'une expression afin de permettre la détection et la simplification éventuelle de formules trigonométriques particulières.

La création d'équations intermédiaires permet également de ne calculer qu'une seule fois des parties communes à plusieurs expressions, ce qui résulte en une plus grande performance lors de l'évaluation du modèle généré.

Gestion des expressions et des équations

Outre des fonctions de génération d'expressions, ROBOTRAN est pourvu de fonctions de gestion des listes d'expressions et d'équations symboliques contenues en mémoire durant la phase de génération.

En effet, durant le processus de génération des expressions, des routines de simplification sont appelées automatiquement. Celles-ci génèrent un nombre plus ou moins important d'expressions intermédiaires lors de leur travail de simplification. Il est crucial de les effacer de la mémoire dès qu'elles ne sont plus nécessaires, afin de limiter l'espace occupé.

Lors de la création d'une équation, on définit une nouvelle variable à l'aide d'une expression symbolique complexe. Cette nouvelle variable est représentée par un nouveau symbole et l'expression complexe qui le définit est désormais masquée et inaccessible aux routines de simplifications. Un processus récursif de sélection permet alors d'identifier les expressions intermédiaires qui ne contribuent pas à la définition de la nouvelle variable et de les effacer de la mémoire.

Ce mécanisme permet de contrôler et de limiter de façon drastique la quantité de mémoire nécessaire à la génération des équations, ce qui constitue un des atouts de ROBOTRAN par rapport aux programmes basés sur des logiciels symboliques commerciaux.

Dans le cas de l'identification de formules trigonométriques particulières, une procédure de vérification est exécutée afin d'éviter la création de variables trigonométriques issues de formules identiques. Si une simplification identique a déjà eu lieu, la variable trigonométrique créée à cette occasion est alors réutilisée.

A la fin de la phase de génération symbolique, lorsque toutes les équations produites par l'exécution de l'algorithme récursif ont été créées, une procédure récursive de sélection est appliquée à la liste des équations pour déterminer celles qui sont réellement utiles pour le modèle. En effet on montre dans [12] qu'environ un tiers des équations générées par un formalisme récursif, ne contribuent pas au calcul de la solution finale. Ces équations sont alors repérées et éliminées avant d'exporter le modèle.

Capacités et performances

Grâce à ses librairies symboliques dédicacées au traitement des équations relatives à la modélisation des systèmes multicorps, ROBOTRAN est un outil simple, rapide et efficace.

Simple, car il fonctionne comme un compilateur, qui produit un fichier contenant les équations du modèle désiré à partir d'un fichier contenant la description du système multicorps.

Rapide, car il permet de modéliser des systèmes composés de dizaines voire de centaines d'articulations et de générer des milliers d'équations contenant plusieurs dizaines de milliers d'opérations, en quelques secondes seulement sur un ordinateur personnel standard.

Efficace, car il est pourvu de fonctions de manipulation symbolique puissantes qui effectuent presque toutes les simplifications possibles afin de produire des modèles contenant un minimum d'opérations arithmétiques.

Enfin, les modèles générés peuvent être exportés en utilisant la syntaxe des langages de programmation les plus répandus tels que FORTRAN, C, MATLAB, JAVA, ce qui garantit la possibilité d'utiliser les modèles dans des environnements les plus divers.

Objectifs poursuivis

A travers ce bref aperçu du domaine de la dynamique des systèmes multicorps, nous constatons qu'il y a une multitude d'approches et de méthodes adaptées à la modélisation et à l'analyse de ces systèmes. Chaque approche, chaque méthode possède des avantages et des inconvénients de telle sorte qu'il est difficile d'affirmer la supériorité universelle de l'une d'entre elles. Les publications qui quantifient cela de façon claire et objective sont d'ailleurs assez rares ! Il est nécessaire de tenir compte de l'application considérée et du contexte de l'analyse afin de choisir l'approche ou la méthode la plus appropriée.

Dans ce travail, nous allons nous intéresser à la modélisation des systèmes essentiellement mécaniques, de petite ou de grande taille, c'est-à-dire contenant jusqu'à près d'une centaine d'articulations tels que

- les robots manipulateurs à structure sérielle ou parallèle,
- les véhicules terrestres évoluant sur route ou sur rails,
- les engins tout-terrains ou volants,
- les mécanismes complexes rencontrés dans divers types de machines.

A l'heure actuelle, ces systèmes sont généralement analysés dans un contexte *mécatronique*. En effet, le système multicorps étudié se trouve très souvent en interaction avec divers composants ou systèmes de contrôle ou d'actionnement hydrauliques, électriques ou électroniques, qui sont également modélisés à l'aide d'outils spécifiques.

C'est donc avec la perspective de l'intégration avec des modèles issus d'autres domaines de l'ingénierie que nous poursuivons ce travail de génération de modèles de systèmes mécaniques multicorps.

Cette perspective entraîne naturellement une série de conditions à satisfaire par les modèles que nous proposons de générer :

- la *portabilité* : ils doivent être facile à intégrer et à utiliser dans des environnements de calcul les plus divers,
- la *performance* : ils doivent constituer un coût calcul minimum tout en permettant une modélisation fidèle de la partie mécanique du système,
- la *fiabilité* : ils doivent être précis et robustes de manière à représenter, le plus exactement possible, le comportement du système mécanique quelles que soient les conditions auxquelles il est soumis et la longueur de l'intervalle de temps simulé.

Dans ce contexte, nous avons choisi :

- l'*approche symbolique* pour générer les modèles, car c'est la seule qui leur confère une portabilité maximale,
- les *coordonnées relatives*, car elles constituent le plus sûr moyen de limiter le nombre d'équations et ainsi d'opérations à effectuer, ce qui constitue une bonne garantie de performance pour les modèles dynamiques de systèmes relativement complexes,
- une démarche de *résolution explicite* des équations de contraintes, basée sur la technique du partitionnement des coordonnées, car c'est la technique qui offre la meilleure garantie de précision et la possibilité de résoudre tous les types de contraintes,
- le *formalisme récursif Newton-Euler*, car, bien que sa complexité soit a priori la moins favorable, il se révèle le plus économique sur la plupart des systèmes multicorps envisagés, ceux-ci possédant rarement une topologie linéaire et un nombre maximal de degrés de liberté,
- une *formulation explicite réduite* des équations du mouvement, car c'est celle qui offre la plus grande simplicité d'utilisation,
- le logiciel ROBOTRAN, car c'est le seul outil de génération symbolique

capable de gérer efficacement le traitement des systèmes de grande taille et qui offre les possibilités de développement nécessaires à la réalisation de ce travail.

Nous avons également choisi d'explorer deux pistes de *parallélisation* des modèles de systèmes multicorps générés symboliquement. En effet le recours au calcul parallèle est la seule solution qui permet de diminuer les temps d'exécution des modèles au delà des meilleurs résultats possibles avec un seul processeur.

Dans ce travail, nous avons apporté diverses contributions au domaine de la dynamique des systèmes multicorps. Nous montrons que

- la technique du partitionnement des coordonnées et la génération symbolique constituent une excellente combinaison pour obtenir des *modèles d'état* de systèmes articulés complexes contenant des boucles cinématiques ou plus généralement soumis à des contraintes algébriques non-linéaires,
- la résolution explicite des équations de contraintes non-linéaires, et pas seulement celle de leurs dérivées premières et secondes, peut être réalisée symboliquement et insérée dans un modèle dynamique,
- la génération symbolique de la formulation réduite des équations du mouvement permet d'économiser un grand nombre d'opérations et de compenser ainsi un inconvénient majeur de la technique du partitionnement des coordonnées,
- les équations récursives des systèmes multicorps, quelle que soit leur structure, possèdent un taux de parallélisme à grain fin très élevé au niveau des opérations arithmétiques et l'utilisation d'architectures parallèles dédicacées permet d'exploiter ce parallélisme,
- des modèles dynamiques générées symboliquement en utilisant des formalismes qui ne sont pas intrinsèquement parallèles, peuvent être parallélisés automatiquement après leur génération afin de les distribuer sur plusieurs processeurs d'un ordinateur parallèle classique.

Au delà de ces objectifs scientifiques, nous apportons également notre contribution au développement du logiciel ROBOTRAN par l'amélioration de ses librairies symboliques et par l'augmentation de ses fonctionnalités.

Structure du document Ce travail comporte deux parties dont le dénominateur commun est l'utilisation de l'approche symbolique.

La première partie est consacrée à la génération symbolique de modèles de systèmes articulés, en vue de leur utilisation sur un ordinateur classique.

Dans le chapitre 1, nous présentons les conventions utilisées et les formalismes récursifs mis en oeuvre pour la modélisation des systèmes à topologie arborescentes. Une section de ce chapitre est consacrée aux extensions apportées au logiciel ROBOTRAN, qui permettent de générer des modèles complètement symboliques et plus efficaces qu'auparavant pour ces systèmes. Deux exemples

de systèmes arborescents sont présentés pour illustrer les différents concepts développés dans le chapitre.

Le chapitre 2 présente les différentes techniques mises en oeuvre pour la modélisation symbolique des systèmes dont la structure contient des boucles cinématiques. Le traitement des équations de contraintes et la réduction des équations du mouvement sur base de l'utilisation conjointe de l'approche symbolique et de la technique du partitionnement des coordonnées y sont développés. Divers exemples de complexités différentes sont analysés pour mettre en évidence les avantages de l'approche symbolique.

La seconde partie est consacrée à la parallélisation des modèles générés par l'approche symbolique.

Dans le chapitre 3, nous présentons une méthode de parallélisation à grain fin. Cette méthode met en évidence un taux de parallélisme jamais exploité, à notre connaissance, dans le domaine de la modélisation des systèmes articulés. Nous proposons de distribuer les opérations arithmétiques sur une architecture parallèle spécifique. Une méthode d'ordonnancement et des résultats de simulation sont présentés. Nous évaluons également le taux de parallélisme à grain fin pour une demi douzaine de systèmes multicorps classiques.

Le chapitre 4 concerne le découpage des modèles dynamiques directs en vue de leur exécution sur des ordinateurs parallèles conventionnels, à mémoire partagée ou en grappe. Grâce à une méthode de marquage des équations, les modèles générés symboliquement peuvent être découpés a posteriori sans nécessiter une implémentation parallèle particulière de l'algorithme de génération. La clé de découpage repose sur une analyse de la topologie du système et de la structure de la matrice Jacobienne des contraintes. Un bogie ferroviaire articulé est utilisé pour illustrer la méthode. Des résultats relatifs au découpage et au calcul parallèle d'un modèle de voiture sont également présentés.

Publications de l'auteur

1. T. Postiau, P. Fisette, and J.D. Legat. Fine grain parallelization of multibody system equations of motion. In *Proc. of Parallel Computing, pp 193-200*, Delft (The Netherlands), August 1999.
2. T. Postiau, P. Fisette, and J.-D. Legat. Generation and parallelization of multibody system equations of motion by a symbolic approach. In *Proc. 5th National Congress on Theoretical and Applied Mechanics*, UCL Louvain-la-Neuve, Belgium, May 23-24, 2000.
3. T. Postiau, P. Fisette, and J.-D. Legat. Generation and parallelization of multibody system equations of motion by a symbolic approach. In *Proc. ICTAM 2000*, Chicago, Aug. 27 - Sep. 2, 2000.
4. T. Postiau, P. Fisette, L. Sass, and J.-Cl. Samin. Real time simulation of parallelized symbolic multibody models of modern road vehicles. In *Proc. 17th IAVSD Symposium*, DTU - Copenhague, Danemark, Aug. 20 - 24, 2001.
5. T. Postiau, L. Sass, P. Fisette, and J.-C. Samin. High-performance multibody models of road vehicles : Fully symbolic implementation and parallel computation. *Supplement to Vehicle System Dynamics*, 2001.
6. J.P. David, J.D. Legat, P. Fisette, and T. Postiau. Implementation of very large dataflow graphs on a reconfigurable architecture for robotic application. In *Proc. of the 15th International Parallel & Distributed Processing Symposium, IPDPS'2001*, San Francisco, 23-27 April 2001.
7. P. Fisette, T. Postiau, L. Sass, and J.-C. Samin. Fully symbolic generation of complex multibody models. *Mechanics of Structures and Machines*, 30(1) :31–82, 2002.
8. T. Postiau, L. Sass, and P. Fisette. Fully symbolic generation of complex multibody models. In *Proc. 6th National Congress on Theoretical and Applied Mechanics*, RUG - Ghent, Belgium, May 26-27, 2003.

Première partie

Génération symbolique de modèles de systèmes multicorps

Chapitre 1

Modélisation des systèmes arborescents

Dans ce chapitre, nous nous intéressons à des systèmes articulés dont la structure est relativement simple, c'est-à-dire, dont la topologie est ouverte, de type linéaire ou arborescente. La modélisation des systèmes contenant des boucles cinématiques fait l'objet du chapitre suivant.

Après une brève présentation des conventions et notations utilisées, nous étudions quelques procédures récursives mises en oeuvre pour la génération symbolique de grandeurs cinématiques. Nous présentons également les formalismes récursifs, basés sur les équations de Newton-Euler, que nous utilisons pour la génération symbolique des équations du mouvement sous forme implicite ou explicite.

Ensuite, nous proposons trois interfaces permettant d'introduire des forces extérieures appliquées sur les corps d'un système lors d'interactions avec son environnement. L'association d'équations cinématiques générées symboliquement et du recours à une fonction externe au modèle, offre à l'utilisateur un moyen souple et efficace pour introduire les forces d'interaction.

Une partie est consacrée aux développements de nouvelles capacités de traitement symbolique du logiciel ROBOTRAN. Celles-ci nous permettent d'obtenir des modèles encore plus compact (10 à 15%) que ceux obtenus avec la version précédente [12].

Finalement, deux systèmes articulés sont analysés afin d'illustrer les différents concepts présentés dans ce chapitre.

1.1 Concepts et conventions

Notre approche de la modélisation des systèmes articulés repose sur une série de conventions et de concepts qui sont présentés dans les sections suivantes.

1.1.1 Topologie : définitions

Chaîne cinématique et filiation

- le nombre de corps (hormis la base) et d'articulations est identique. On représente une articulation physique à plusieurs degrés de liberté par une séquence équivalente d'articulations élémentaires à un degré de liberté et de corps fictifs.
- on reliera toujours au moins un corps du système réel à une base inertielle qui sert de référence, généralement le sol, pour la plupart des applications terrestres.

Les articulations sont des éléments physiques tels que des charnières, des glissières, des rotules. Elles ne sont pas considérées comme des corps dans la modélisation du système, mais seulement comme des éléments de connexion entre deux corps. La figure 1.1 illustre les *points d'ancrage* des articulations sur les corps. Ces points sont utilisés pour calculer la résultante de toutes les forces agissant au sein de l'articulation.

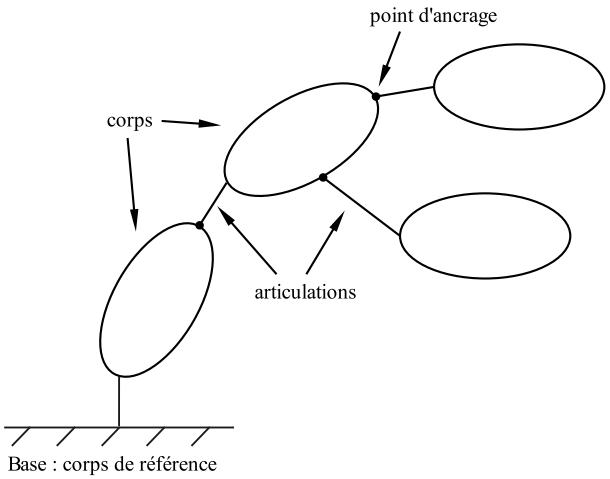


FIG. 1.1 – Schéma d'un système multicorps

- une *chaîne cinématique* ou un *chemin* est un ensemble de plusieurs corps consécutifs rencontrés lorsqu'on parcourt la structure de la base vers une

extrémité. Sur la figure 1.2, $\{i, j, k\}$, $\{l, k, j, m, n\}$ sont des chaînes cinématiques mais pas $\{j, n, o\}$, ni $\{i, k, j\}$.

- il n'y a qu'un seul chemin qui relie deux corps d'un système arborescent.
- la *base* est le corps fixe correspondant à la référence inertielle.
- un corps i est un *ancêtre* du corps j s'il appartient à la chaîne cinématique qui commence par la base et se termine juste avant le corps j. La base est l'ancêtre de tous les corps.
- un corps j est un *descendant* du corps i si celui-ci est un ancêtre du corps j.
- le corps i est le *parent* du corps j si il est l'ancêtre direct du corps j, c'est-à-dire si le corps j est le descendant attaché au corps i. Un corps n'a qu'un seul parent dans une structure arborescente.
- le corps j est un *enfant* du corps i si celui-ci est le parent du corps j. Un corps peut avoir plusieurs enfants.
- Nous noterons $\overline{i}$ l'ensemble des enfants d'un corps i.
- un corps *terminal* est un corps sans enfant(s).
- une *branche* est une chaîne cinématique qui a les propriétés suivantes :
  - le dernier corps est un corps terminal ou un corps qui a plusieurs enfants,
  - l'ancêtre du premier corps est la base ou un corps qui a plusieurs enfants.
- les concepts de filiation et de parenté s'appliquent aussi aux branches,
- deux branches sont parallèles si aucun corps d'une branche n'est l'ancêtre d'un corps de l'autre.

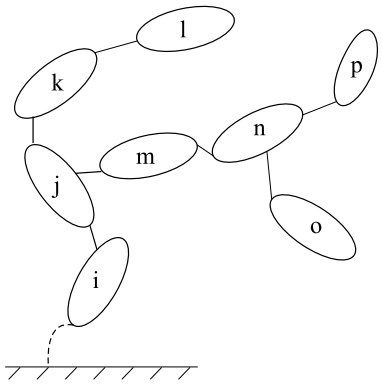


FIG. 1.2 – Chaîne arborescente de corps

On peut illustrer ces concepts en se référant à la figure 1.2 :

- le corps j est le parent des corps k et m,

- le corps j est l'ancêtre des corps k, l, m, n, o, p,
- le corps n est le parent des corps o et p,
- le corps k est un descendant des corps i, j, et l'enfant du corps j,
- l'ensemble $\bar{j}$ des enfants du corps j est $\{k, m\}$,
- les corps l, p, o sont des corps terminaux,
- $\{i, j\}$, $\{k, l\}$, $\{m, n\}$, $\{o\}$, et $\{p\}$ sont des branches.

Numérotation

- les corps sont numérotés par ordre croissant en partant de la base (qui porte le numéro 0) vers les corps terminaux, comme illustrés sur la figure 1.3.
- l'articulation qui précède un corps porte le même numéro que celui-ci.
- le vecteur *inbody* est défini de telle sorte que son élément i contient le numéro du parent du corps i.

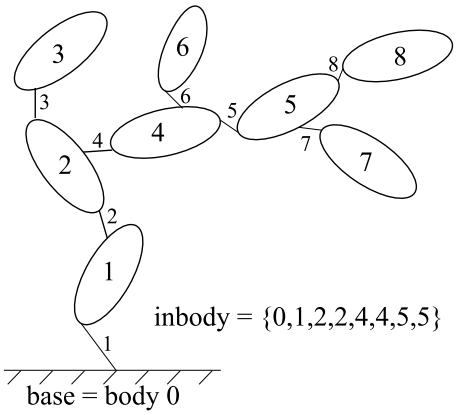


FIG. 1.3 – Numérotation des articulations et des corps

Notations

Lors du développement des équations cinématiques et dynamiques, nous allons utiliser les écritures [1] suivantes :

- $\sum_i a^i$: somme sur tous les corps ($i = 1, ..., N^{body}$) de la structure,
- $\sum_{i:\, i<j} a^i$: somme sur tous les ancêtres du corps j,

[1] Les inégalités utilisées dans ces notations ne correspondent pas à un ordre mathématique : $i < j$ signifie que i est un ancêtre de j et donc, $i \not< j$ n'implique pas $i \geq j$. En effet, i peut se situer sur une branche parallèle.

- $\sum_{j:\, i<j} a^j$: somme sur tous les descendants du corps i,
- $\sum_{i:\, i\leq j} a^i$, $\sum_{j:\, i\leq j} a^j$: respectivement, somme sur le corps j et tous ses ancêtres, et somme sur le corps i et tous ses descendants,
- $\sum_{i:\, h\leq i<j} a^i$: somme sur le corps h et les corps i qui sont à la fois les descendants de h et les ancêtres de j,
- $\sum_{j\,\in\,\bar{\imath}}$: somme sur tous les enfants du corps i,
- $ch^{h:i<h\leq j}$ la chaîne cinématique des corps s'étendant du corps i, non inclus, au corps j inclus, $ch^{h:i<h\leq j}(n)$ étant le n$^{\text{ième}}$ corps de la chaîne.

1.1.2 Cinématique : définitions

Notations

Nous utilisons la figure 1.4 pour illustrer les définitions des éléments relatifs au corps rigide i, à son corps parent h et à ses enfants j et k.

- O^i est le point d'ancrage de l'articulation du corps i sur son corps parent h,
- O'^i est l'origine du repère du corps i et donc le point de référence du corps i, le point d'ancrage de l'articulation i sur le corps i. Le point O'^i est confondu avec le point O^i si l'articulation est rotoïde.
- $\mathbf{z}^i$ est le vecteur $\overrightarrow{O^i\,O'^i}$, qui représente le déplacement relatif au sein de l'articulation i,
- $\hat{\mathbf{e}}^i$ est le vecteur unitaire aligné avec l'axe de l'articulation i,
- $\mathbf{d}^{ik}$ est le vecteur $\overrightarrow{O'^i\,O^k}$ qui représente la position du point d'ancrage sur le corps i de l'articulation du corps k par rapport au point de référence du corps i(qui est le parent de k). Les composantes de ce vecteurs sont constantes dans le repère du corps i.
- $\mathbf{p}^i$, le vecteur position absolue du point de référence O'^i. Attention, cette définition correspond à une convention relative à l'implémentation des formalismes dans ROBOTRAN. Elle diffère de la définition faite dans [13]
- $[\hat{\mathbf{X}}]$ représente le triplet de vecteurs unitaires $\begin{bmatrix}\hat{\mathbf{X}}_1\\ \hat{\mathbf{X}}_2\\ \hat{\mathbf{X}}_3\end{bmatrix}$ de la base orthonormée $\{\hat{\mathbf{X}}\}$.

Pour une écriture plus compacte des développements cinématiques ultérieurs, on définit des vecteurs position augmentés. Ceux-ci sont illustrés sur la figure 1.5.

- $\mathbf{d}_z^{ij} \triangleq \mathbf{d}^{ij} + \mathbf{z}^j$, $\mathbf{d}_z^{ik} \triangleq \mathbf{d}^{ik} + \mathbf{z}^k$, les *vecteurs position augmentés* des articulations j et k respectivement.

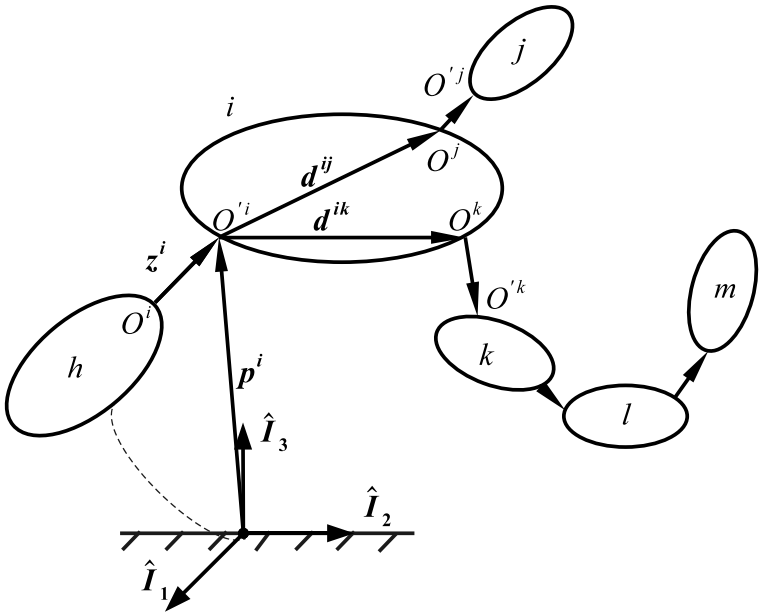


FIG. 1.4 – Notations cinématiques : points et vecteurs position

Sur la figure 1.6 sont illustrés des éléments utilisés pour la description de l'orientation relative des corps.

- $\{O, \{\hat{\mathbf{I}}\}\}$, le repère inertiel de référence solidaire du corps de base 0,
- $\{\hat{\mathbf{X}}^i\}$, la base solidaire du corps i. Dans la base $\{\hat{\mathbf{X}}^i\}$, les composantes d^{ij} et d^{ik} des vecteurs $\mathbf{d}^{ij} = [\hat{\mathbf{X}}^i]^T\, d^{ij}$ et $\mathbf{d}^{ik} = [\hat{\mathbf{X}}^i]^T\, d^{ik}$ sont constantes pour un corps rigide,
- $R^{i,h}$, la matrice de rotation permettant de passer de la base $\{\hat{\mathbf{X}}^h\}$ à la base $\{\hat{\mathbf{X}}^i\}$: $[\hat{\mathbf{X}}^i] = R^{i,h}[\hat{\mathbf{X}}^h]$,
- $\mathbf{\Omega}^i$, le vecteur vitesse angulaire relative du corps i par rapport à son parent h,
- $\boldsymbol{\omega}^i = [\hat{\mathbf{X}}^i]^T\, \omega^i$, le vecteur vitesse angulaire absolue du corps i, donc de la base $\{\hat{\mathbf{X}}^i\}$ par rapport à la base inertielle $\{\hat{\mathbf{I}}\}$.

1.1.3 Hypothèses de modélisation des articulations

Les corps sont généralement reliés entre eux par des éléments mécaniques tels que des glissières, des rotules, des paliers, des charnières, etc. Les mouvements relatifs des corps ainsi reliés seront différents selon la nature de ces élé-

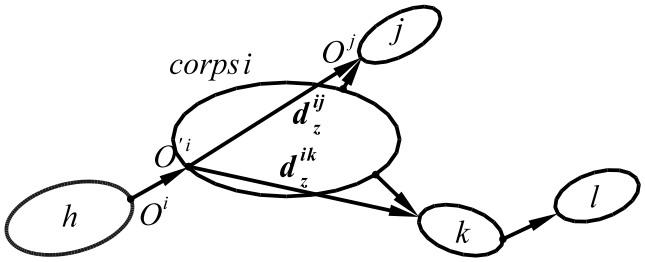


FIG. 1.5 – Vecteurs position augmentés

ments mécaniques, chacun offrant un nombre de degrés de liberté propre. Plutôt que de proposer une librairie contenant tous les types d'articulations physiques existantes ou imaginables, nous proposons de modéliser les articulations complexes en combinant successivement plusieurs articulations élémentaires. Deux articulations à un degré de liberté sont proposées ; l'une *prismatique*, l'autre *rotoïde*. Elles sont illustrées sur la figure 1.7 et correspondent respectivement à des mouvements relatifs en translation ou en rotation. Les variables représentant les mouvement relatifs sont utilisées comme *coordonnées généralisées*, notées q pour l'écriture des équations décrivant le mouvement des corps.

Pour chaque articulation i, la coordonnée généralisée q^i représente :

- la valeur du déplacement relatif $\overrightarrow{O^i\,O'^i}$ selon le vecteur articulaire $\hat{\mathbf{e}}^i$, si l'articulation i est prismatique,
- l'angle de rotation relative ϑ^i du corps i par rapport à son parent selon le vecteur articulaire $\hat{\mathbf{e}}^i$ si l'articulation i est rotoïde.

Les position et orientation relatives du corps i par rapport à son parent h sont données par les expressions suivantes :

- pour une articulation prismatique i :
  - l'orientation relative de la base $\{\hat{\mathbf{X}}^i\}$ par rapport à la base $\{\hat{\mathbf{X}}^h\}$ est exprimée par une matrice de rotation élémentaire constante :

$$[\hat{\mathbf{X}}^i] = R^{i,h}[\hat{\mathbf{X}}^h] \tag{1.1}$$

    Selon les conventions de modélisation utilisées dans ROBOTRAN, les deux bases $\{\hat{\mathbf{X}}^i\}$ et $\{\hat{\mathbf{X}}^h\}$ ont toujours un vecteur en commun, lequel correspond à l'axe de l'articulation i.
  - la position relative du point de référence O'^i, du corps i, par rapport au point d'ancrage O^i de l'articulation i sur le corps parent h, est donnée par :

$$\mathbf{z}^i = \overrightarrow{O^i\,O'^i} = q^i\hat{\mathbf{e}}^i$$

    ce qui signifie que lorsque $q^i = 0$, O'^i coïncide avec O^i.
- pour une articulation rotoïde i :

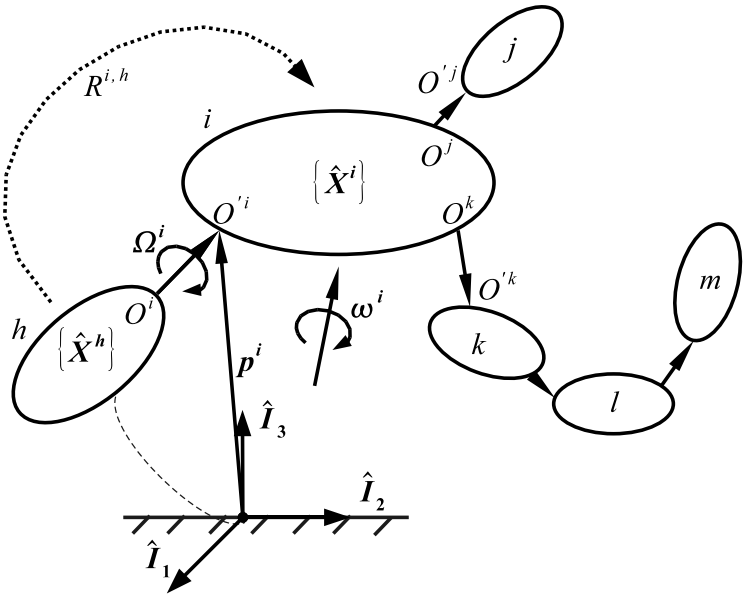


FIG. 1.6 – Notations cinématiques : repères et vecteurs

– l'orientation relative du repère $\{\hat{\mathbf{X}}^i\}$ par rapport au repère $\{\hat{\mathbf{X}}^h\}$ est exprimée dans l'équation :

$$[\hat{\mathbf{X}}^i] = R^{i,h}[\hat{\mathbf{X}}^h] \tag{1.2}$$

où l'angle q^i est la seule variable présente dans l'expression de la matrice $R^{i,h}$.

– les points de référence O'^i et d'ancrage O^i, sont situés sur l'axe de l'articulation et coïncident :

$$\mathbf{z}^i = \overrightarrow{O^i O'^i} = 0$$

Pour les deux types d'articulation, le vecteur articulaire $\hat{\mathbf{e}}^i$ a des composantes constantes dans les deux bases $\{\hat{\mathbf{X}}^i\}$ et $\{\hat{\mathbf{X}}^h\}$. De plus selon les conventions de modélisation utilisées, deux de ses trois composantes sont toujours nulles et la troisième vaut l'unité.

On définit les vecteurs vitesse *relative* linéaire $\overset{\circ}{\mathbf{z}}{}^i$ et angulaire $\mathbf{\Omega}^i$ du corps i par rapport à son parent h :

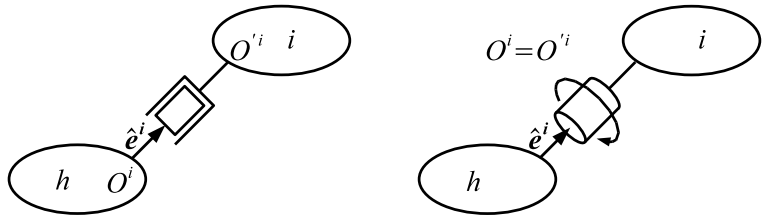


FIG. 1.7 – Articulations élémentaires prismatique ou rotoïde

$$\overset{\circ}{\mathbf{z}}{}^i \triangleq \dot{q}^i\,\hat{\mathbf{e}}^i\ ,\ \boldsymbol{\Omega}^i \triangleq 0 \quad \text{si } i \text{ est prismatique,}$$
$$\boldsymbol{\Omega}^i \triangleq \dot{q}^i\,\hat{\mathbf{e}}^i\ ,\ \overset{\circ}{\mathbf{z}}{}^i \triangleq 0 \quad \text{si } i \text{ est rotoïde.}$$

Pour chaque articulation i, on définit les vecteurs $\boldsymbol{\varphi}^i$ et $\boldsymbol{\psi}^i$, comme suit :

$$\boldsymbol{\psi}^i \triangleq \hat{\mathbf{e}}^i \ \text{ et } \ \boldsymbol{\varphi}^i \triangleq 0 \quad \text{si } i \text{ est prismatique,} \tag{1.3}$$
$$\boldsymbol{\varphi}^i \triangleq \hat{\mathbf{e}}^i \ \text{ et } \ \boldsymbol{\psi}^i \triangleq 0 \quad \text{si } i \text{ est rotoïde.} \tag{1.4}$$

Cette définition nous permet d'écrire les relations suivantes pour les deux types d'articulations :

$$\overset{\circ}{\mathbf{z}}{}^i \triangleq \dot{q}^i\,\boldsymbol{\psi}^i \text{ et } \boldsymbol{\Omega}^i \triangleq \dot{q}^i\,\boldsymbol{\varphi}^i\ , \tag{1.5}$$
$$\overset{\circ\circ}{\mathbf{z}}{}^i \triangleq \ddot{q}^i\,\boldsymbol{\psi}^i \text{ et } \overset{\circ}{\boldsymbol{\Omega}}{}^i \triangleq \ddot{q}^i\,\boldsymbol{\varphi}^i\ .$$

1.1.4 Dynamique : définitions

Nous utilisons la figure 1.8 pour illustrer les définitions des éléments relatifs au corps rigide i :

- m^i et $\mathbf{I}^i = [\hat{\mathbf{X}}^i]^T\, I^i\, [\hat{\mathbf{X}}^i]$, la masse et le tenseur d'inertie, par rapport au centre de masse G^i, du corps i,
- $\mathbf{d}^{ii} = [\hat{\mathbf{X}}^i]^T\, d^{ii}$, le vecteur position du centre de masse G^i par rapport au point de référence O'^i. Les composantes d^{ii} et I^i sont constantes dans le repère $\{\hat{\mathbf{X}}^i\}$,
- $\mathbf{x}^i = [\hat{\mathbf{I}}]^T\, x^i$, le vecteur position absolue du centre de masse G^i,
- $\mathbf{g} = [\hat{\mathbf{I}}]^T\, g$, le vecteur gravité[2] ,

[2]On suppose que le champ gravitationnel est constant. La résultante des forces de gravité s'applique au centre de masse G^i et est donnée par $m^i\mathbf{g}$.

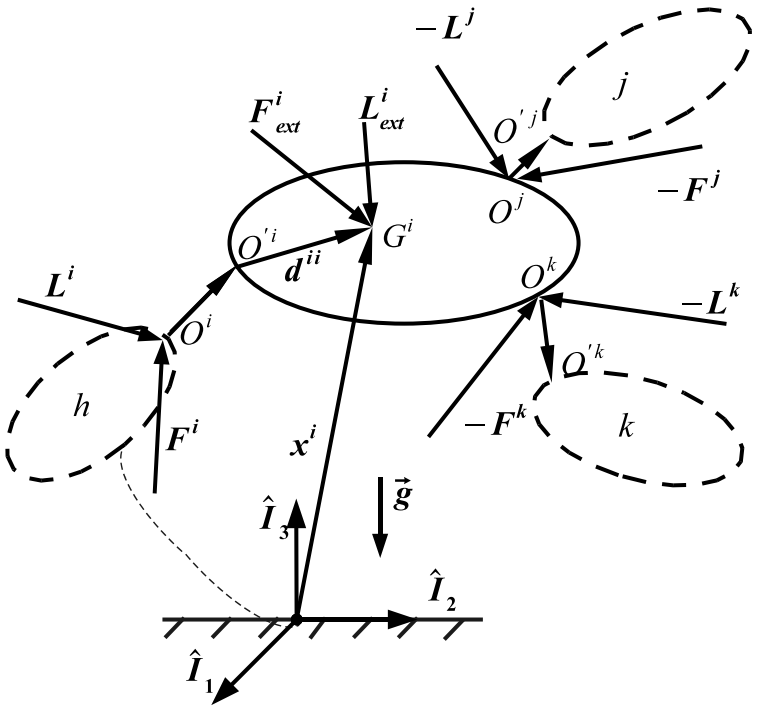


FIG. 1.8 – Notations principales pour la dynamique

- $\mathbf{F}^i$, $\mathbf{L}^i$, les résultantes de forces et de couples appliqués sur le corps i au point O'^i par son parent h via l'articulation i.
  Des forces et couples de réactions opposées,$-\mathbf{F}^i$ et $-\mathbf{L}^i$, sont appliqués au corps h selon la 3^{ime} loi de Newton.
  Le corps i est lui aussi soumis aux forces et couples de réactions $-\mathbf{F}^j$, $-\mathbf{F}^k$, $-\mathbf{L}^j$, et $-\mathbf{L}^k$ de ses enfants j et k,
- $\mathbf{F}^i_{ext}$, $\mathbf{L}^i_{ext}$, les résultantes des forces et couples externes (hormis la gravité et les forces généralisées internes provenant des articulations) appliqués sur le corps i en son centre de masse.

1.2 Génération de grandeurs cinématiques

Nous développons dans cette section des méthodes de calcul permettant de générer les expressions des vecteurs position, vitesses et accélérations d'un point P attaché à un corps du système. Ce point peut être le point outil d'un robot

manipulateur, un point correspondant à l'emplacement d'un accéléromètre dans un véhicule, etc. D'autres grandeurs sont également utiles, telles les matrices de rotation permettant de recalculer les coordonnées des vecteurs, afin de changer la base dans laquelle elles sont exprimées.

Une autre entité intéressante est la matrice Jacobienne à laquelle nous consacrons la section 1.2.2. Cette matrice a une grande utilité en robotique. Elle sera également utilisée pour le traitement des boucles cinématiques que nous abordons dans le chapitre 2.

Nous exprimerons ici les grandeurs (position, vitesses, etc.) par rapport au repère inertiel. Plus loin, nous verrons qu'il est également utile de pouvoir générer les grandeurs relativement à un autre repère, pour la génération des équations de calcul des forces d'interactions d'un système avec son environnement ou pour traiter les boucles cinématiques.

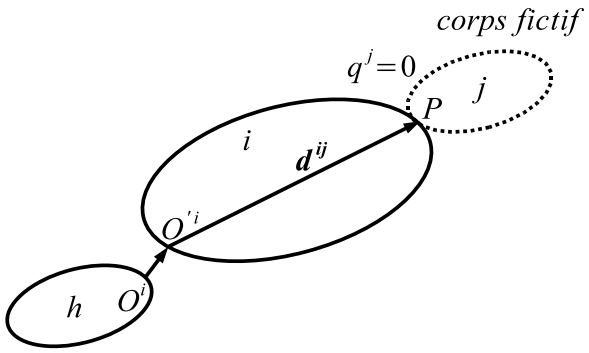


FIG. 1.9 – Ajout d'un corps fictif dont l'origine est P

Afin de simplifier la procédure de calcul, nous ferons l'hypothèse que le point P est l'origine du repère d'un corps, que nous appellerons le corps final, comme illustré sur la figure 1.9. En pratique, afin de satisfaire cette hypothèse en toute circonstance, il suffit de considérer qu'un corps *fictif* supplémentaire est relié au corps portant le point par une articulation fictive dont la coordonnée relative associée est constamment nulle. Les repères du corps fictif ajouté et du corps porteur du point P ont donc la même orientation.

Nous allons analyser trois approches récursives. Tout d'abord en effectuant un parcours *progressif*, partant du corps de référence vers le corps final. Ensuite en effectuant un parcours *régressif*, partant du corps final et remontant vers le corps de référence. Finalement nous envisagerons une approche mixte dans laquelle certaines grandeurs sont calculées en effectuant un parcours progressif et d'autres selon un parcours régressif.

Nous comparerons les différents approches en terme de nombre d'opérations

et discuterons l'intérêt de chaque méthode. L'objectif de cette démarche est de déterminer quelle est la méthode à utiliser pour générer des équations qui nécessitent le minimum d'opérations pour le calcul des grandeurs cinématiques.

1.2.1 Position, vitesses et accélérations

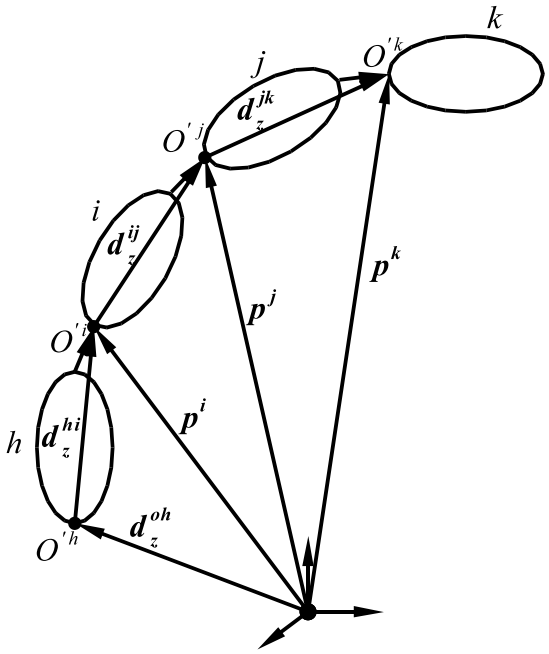


FIG. 1.10 – Chaîne de corps

En se basant sur la figure 1.10 et sur les définitions de vecteurs faites dans la section 1.1.2, on peut écrire la relation suivante :

$$\mathbf{p}^k = \sum_{i:\, i<k} \mathbf{d}_z^{hi} = \ldots + \mathbf{d}_z^{hi} + \mathbf{d}_z^{ij} + \mathbf{d}_z^{jk} \tag{1.6}$$

où $\mathbf{p}^k$ est le vecteur position absolue du point de référence[3] O'^k du corps k, qui peut être calculée en additionnant les vecteurs augmentés situés sur le chemin menant de l'origine au corps k . Il y a plusieurs façon d'effectuer la somme de ces vecteurs augmentés. Nous en examinons deux.

[3]Le point de référence du corps k est l'origine du repère solidaire du corps qui est située au point d'ancrage de l'articulation k sur le corps k

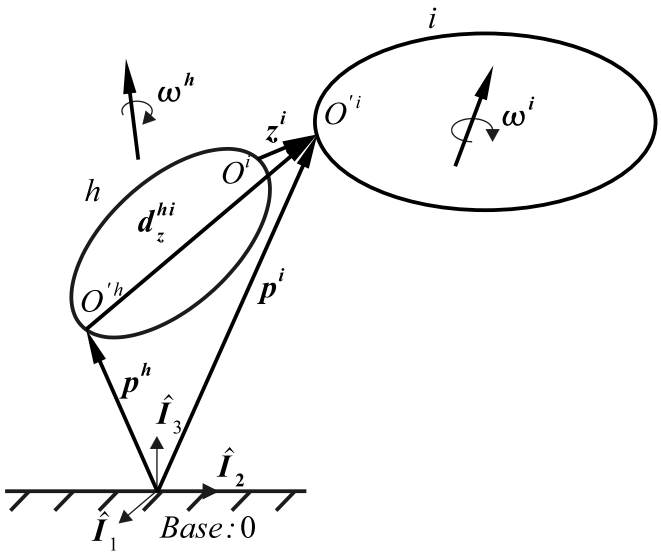


FIG. 1.11 – Cinématique : parcours progressif

On peut additionner progressivement les vecteurs augmentés en partant de l'origine et en parcourant le chemin qui conduit au corps final k. La somme partielle correspond alors à chaque étape à la position absolue du point de référence d'un des ancêtres du corps final k.

Une autre façon d'additionner les vecteurs est de parcourir le chemin à l'envers, en remontant du corps k vers le corps de base. Les sommes intermédiaires ne sont alors plus des positions absolues, mais bien les positions relatives du point de référence du corps final k par rapport à chacun de ses ancêtres de plus en plus éloignés. L'ancêtre le plus éloigné est le corps de base et on obtient finalement la position absolue du point de référence O'^k du corps k.

Dans les sections suivantes, nous allons développer les expressions cinématiques récursives, non seulement de la position, mais aussi des vitesses et accélérations, angulaire et linéaire, en utilisant trois méthodes différentes.

Méthode progressive

La première méthode consiste donc à parcourir le chemin $ch^{h:0<h\leq f}$ partant de la base pour atteindre le corps final f dans le sens 'naturel'. Il faut cependant noter que ce chemin est recherché a priori et construit en reculant à partir du corps final et en notant les numéros des parents successifs contenus dans le vecteur *inbody*.

Les équations récursives suivantes sont développées sur base de la figure 1.11, qui représente un corps rigide i attaché à un corps parent h via l'articulation i. Les positions, vitesses, etc. définies ci-dessous sont absolues, c'est-à-dire qu'elles se rapportent au repère de base.

- Vecteur position :

$$\mathbf{p}^i = \mathbf{p}^h + \mathbf{d}_z^{hi}, \tag{1.7}$$

- Vecteurs vitesse :
  - angulaire :

$$\boldsymbol{\omega}^i = \boldsymbol{\omega}^h + \boldsymbol{\Omega}^i = \boldsymbol{\omega}^h + \boldsymbol{\varphi}^i \dot{q}^i \tag{1.8}$$

  - linéaire :

$$\dot{\mathbf{p}}^i = \dot{\mathbf{p}}^h + \widetilde{\boldsymbol{\omega}}^h \, \mathbf{d}_z^{hi} + \boldsymbol{\psi}^i \dot{q}^i \tag{1.9}$$
$$\tag{1.10}$$

- Vecteurs accélération :
  - angulaire :

$$\dot{\boldsymbol{\omega}}^i = \dot{\boldsymbol{\omega}}^h + \widetilde{\boldsymbol{\omega}}^i \, \boldsymbol{\varphi}^i \dot{q}^i + \boldsymbol{\varphi}^i \ddot{q}^i \tag{1.11}$$

  - linéaire :

$$\ddot{\mathbf{p}}^i = \ddot{\mathbf{p}}^h + (\widetilde{\dot{\boldsymbol{\omega}}}^h + \widetilde{\boldsymbol{\omega}}^h \, \widetilde{\boldsymbol{\omega}}^h) \, \mathbf{d}_z^{hi} + 2 \, \widetilde{\boldsymbol{\omega}}^i \, \boldsymbol{\psi}^i \dot{q}^i + \boldsymbol{\psi}^i \ddot{q}^i \tag{1.12}$$

Sur base de ces équations, nous pouvons écrire le schéma de calcul *progressif*. Toutes les grandeurs définies dans ce schéma sont exprimées dans le repère de base $\{O, \{\hat{\mathbf{I}}\}\}$.

Initialisation :

$$p^0 = \dot{p}^0 = \omega^0 = \dot{\omega}^0 = 0$$
$$R^{0,0} = E, \text{ la matrice identité}$$

Recursion :

For $j = 1 : n$

$$
\begin{aligned}
i &= ch(j) \\
h &= inbody(i); \\
R^{0,i} &= R^{0,h} R^{h,i} \\
rl^h &= R^{0,h} d_z^{hi} \\
p^i &= p^h + rl \\
&- - - \\
\omega^i &= \omega^h + R^{0,i} \varphi^i \dot{q}^i \\
orl &= \tilde{\omega}^h \; rl^h \\
\dot{p}^i &= \dot{p}^h + orl + R^{0,i} \psi^i \dot{q}^i \\
T^h &= \tilde{\omega}^h R^{0,h} \\
&- - - \\
\dot{\omega}^i &= \dot{\omega}^h + T^h \varphi^i \dot{q}^i + R^{0,i} \varphi^i \ddot{q}^i \\
\ddot{p}^i &= \ddot{p}^h + \tilde{\dot{\omega}}^h rl + \tilde{\omega}^h orl + 2T^h \psi^h \dot{q}^h + R^{0,i} \psi^i \ddot{q}^i
\end{aligned}
$$

end.

On notera que :

- des variables intermédiaires rl et orl sont définies et utilisées à chaque étape. Leur utilité est évidente : elles permettent d'éviter de recalculer plusieurs fois les mêmes expressions,
- il n'est pas nécessaire de définir de variable intermédiaire pour $R^{0,i}\varphi^i$ car φ^i n'a toujours qu'une seule composante non nulle,
- les composantes des vecteurs $\mathbf{d}_z^{hi}$, sont exprimées dans le repère du corps h parent du corps i et sont donc projetées dans le repère de base en utilisant la matrice $R^{0,h}$,
- cette matrice de rotation $R^{0,h}$ est, en fait, calculée à l'étape $j-1$.

Méthode régressive

La seconde méthode consiste à parcourir le chemin $ch^{0<h\leq f}$ en partant du corps final f en en remontant vers la base. Le parcours se fait naturellement en utilisant le vecteur *inbody* à chaque étape.

Les équations récursives suivantes sont développées sur base de la figure 1.12, qui représente un corps rigide i attaché à son corps parent h via l'articulation i. Les positions, vitesses, etc. définies ci-dessous sont les valeurs correspondant au corps final f et relatives à ses différents ancêtres. Il ne s'agit plus ici des grandeurs absolues ! Ainsi $\mathbf{p}^{if}$ est le vecteur position relative du point

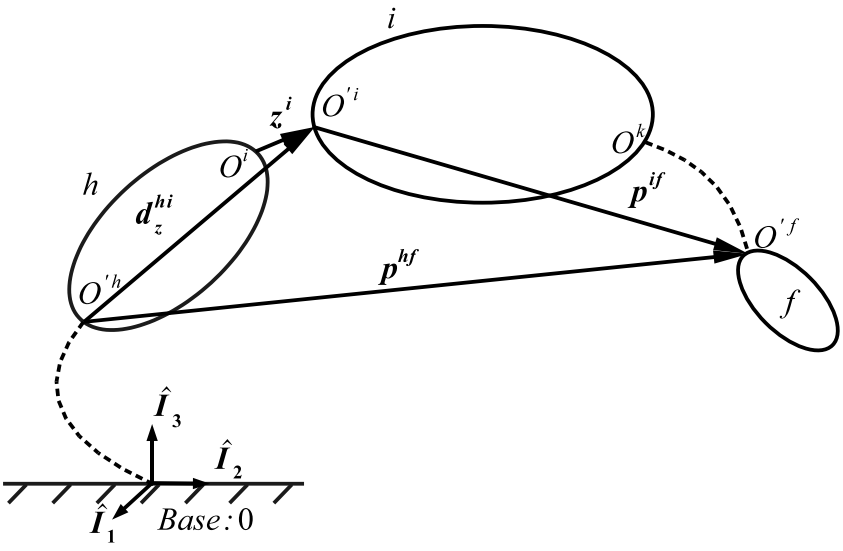


FIG. 1.12 – Cinématique : parcours régressif

O'^f par rapport au point O'^i. Le vecteur vitesse de rotation $\boldsymbol{\omega}^{if}$ est le vecteur vitesse de rotation du repère du corps f relativement au repère du corps i.

- Vecteur position :

$$\mathbf{p}^{hf} = \mathbf{p}^{if} + \mathbf{d}_z^{hi}, \tag{1.13}$$

- Vecteurs vitesse :
  - angulaire :

$$\boldsymbol{\omega}^{hf} = \boldsymbol{\omega}^{if} + \boldsymbol{\Omega}^i = \boldsymbol{\omega}^{if} + \boldsymbol{\varphi}^i \dot{q}^i \tag{1.14}$$

  - linéaire :

$$\overset{\circ}{\mathbf{p}}{}^{hf} = \overset{\circ}{\mathbf{p}}{}^{if} + \widetilde{\boldsymbol{\Omega}}^i \, \mathbf{p}^{if} + \boldsymbol{\psi}^i \dot{q}^i \tag{1.15}$$

- Vecteurs accélération :
  - angulaire :

$$\overset{\circ}{\boldsymbol{\omega}}{}^{hf} = \overset{\circ}{\boldsymbol{\omega}}{}^{if} + \widetilde{\boldsymbol{\Omega}}^i \, \boldsymbol{\omega}^{if} + \boldsymbol{\varphi}^i \ddot{q}^i \tag{1.16}$$

  - linéaire :

$$\overset{\circ\circ}{\mathbf{p}}{}^{hf} = \overset{\circ\circ}{\mathbf{p}}{}^{if} + (\overset{\circ}{\widetilde{\boldsymbol{\Omega}}}{}^i + \widetilde{\boldsymbol{\Omega}}^i \, \widetilde{\boldsymbol{\Omega}}^i) \, \mathbf{p}^{if} + 2 \, \widetilde{\boldsymbol{\Omega}}^i \, \overset{\circ}{\mathbf{p}}{}^{if} + \boldsymbol{\psi}^i \ddot{q}^i \tag{1.17}$$

$$\tag{1.18}$$

Sur base de ces équations récursives, nous pouvons écrire le schéma de calcul *régressif*. Les grandeurs définies dans ce schéma sont exprimées dans un repère différent à chaque étape; les vecteurs relatifs au corps i et exprimés dans le repère du corps i, sont projetés dans le repère du corps parent h par multiplication avec la matrice $R^{h,i}$, avant d'être utilisés pour calculer le nouveau vecteur relatif au corps parent h.

Initialisation :

$$p^{f,f} = \dot{p}^{f,f} = \ddot{p}^{f,f} = \omega^{f,f} = \dot{\omega}^{f,f} = 0$$
$$R^{f,f} = E, \text{ la matrice identité}$$

Récursion :

For $j = n : 1$

$$\begin{aligned}
i &= ch(j) \\
h &= inbody(i); \\
R^{f,h} &= R^{f,i} R^{i,h} \\
rl^h &= R^{h,i} p^{i,f} \\
p^{h,f} &= rl^h + d_z^{hi} \\
&--- \\
\omega^{h,i} &= \varphi^i \dot{q}^i \\
\omega^{h,f} &= R^{h,i} \omega^{i,f} + \omega^{h,i} \\
orl &= \tilde{\omega}^{h,i}\ rl^h\ L^j \\
Rv^h &= R^{h,i} \dot{p}^{i,f} \\
\dot{p}^{h,f} &= Rv^h + orl + \psi^i \dot{q}^i \\
&--- \\
\dot{\omega}^{h,f} &= R^{h,i} \dot{\omega}^{i,f} + \tilde{\Omega}^i \omega^{h,f} + \varphi^i \ddot{q}^i \\
\ddot{p}^{h,f} &= R^{h,i} \ddot{p}^{i,f} + \dot{\tilde{\omega}}^{h,i} rl^h + 2\tilde{\omega}^{h,i} Rv^h + \tilde{\omega}^{h,i} orl + \psi^i \ddot{q}^i
\end{aligned}$$

end.

On notera que :

- on définit aussi des variables intermédiaires rl et orl, utilisées à chaque étape, mais elles ne sont pas identiques aux variables du schéma progressif. Nous ne leur avons pas donné un nom différent car leur portée est très locale et le risque de confusion est très limité.
- la matrice de rotation $R^{f,h}$ n'est pas utilisée dans les étapes du schéma régressif car, les projections se font dans le repère du corps parent h au moyen de la matrice de rotation $R^{h,i}$ de l'articulation i,

- ce ne sont plus les vecteurs $\mathbf{d}_z^{hi}$, $\boldsymbol{\varphi}^i$ et $\boldsymbol{\psi}^i$, mais les sommes partielles de ces vecteurs qui sont projetées dans le repère du corps parent.

Discussion

Afin de comparer les démarches *progressive* et *régressive*, nous avons généré symboliquement les équations nécessaires au calcul de différentes grandeurs cinématiques relatives à un repère situé à l'extrémité d'un bras manipulateur de type série à cinq degrés de liberté. Les nombres d'opérations arithmétiques et trigonométriques en virgule flottante nécessaires pour l'évaluation de ces équations sont repris dans la table 1.1. Les opérations trigonométriques correspondent aux calcul des *sinus* et *cosinus* des coordonnées relatives des articulations rotoïdes. Le nombre d'opérations trigonométriques est le même pour les deux méthodes.

		Progressive	Régressive
1	p	69	43
2	p, R	73	75
3	p, $\dot{p}$	118	87
4	p, R, $\dot{p}$, ω,	122	130
5	p, R, $\dot{p}$, ω, $\ddot{p}$, $\dot{\omega}$	247	259

TAB. 1.1 – Comparaison des méthodes en terme de nombre d'opérations arithmétiques et trigonométriques

Si les équations vectorielles récursives relatives à chaque méthode semblent contenir un nombre d'opérations vectorielles tout à fait identiques dans les deux cas, on constate, au vu de la table 1.1, que ce n'est pas le cas pour les équations générées.

Afin de comprendre les différences, il faut examiner les équations des schémas récursifs. On peut y constater que dans le premier schéma, la matrice de projection $R^{0,h}$ est nécessaire pour le calcul de toutes les grandeurs intermédiaires, ce qui n'est pas le cas dans le second schéma. Ceci explique l'avantage de la seconde méthode pour les cas 1 et 3 de la table 1.1 où la matrice de rotation n'est pas demandée.

On remarque que dans les cas 4 et 5, la seconde méthode est légèrement moins performante que la première. L'explication se situe au niveau des projections. Dans la première méthode, les vecteurs projetés sont principalement les vecteurs articulaires $\boldsymbol{\varphi}^i$ et $\boldsymbol{\psi}^i$ qui sont (par convention de modélisation avec ROBOTRAN) alignés avec un des axe du repère du corps i et n'ont par conséquent qu'une seule composante non nulle. Ce n'est pas le cas dans la deuxième méthode où les vecteurs projetés ne sont généralement pas alignés avec un axe du repère du corps i, et donc leur projection requiert plus d'opérations. Cette

différence peut devenir prépondérante pour le calcul des vitesses et accélérations linéaires sur de longues chaînes cinématiques.

Il semble donc que le principal intérêt de la seconde méthode concerne le calcul de la position et éventuellement de la vitesse linéaire de l'extrémité d'un chaîne cinématique courte.

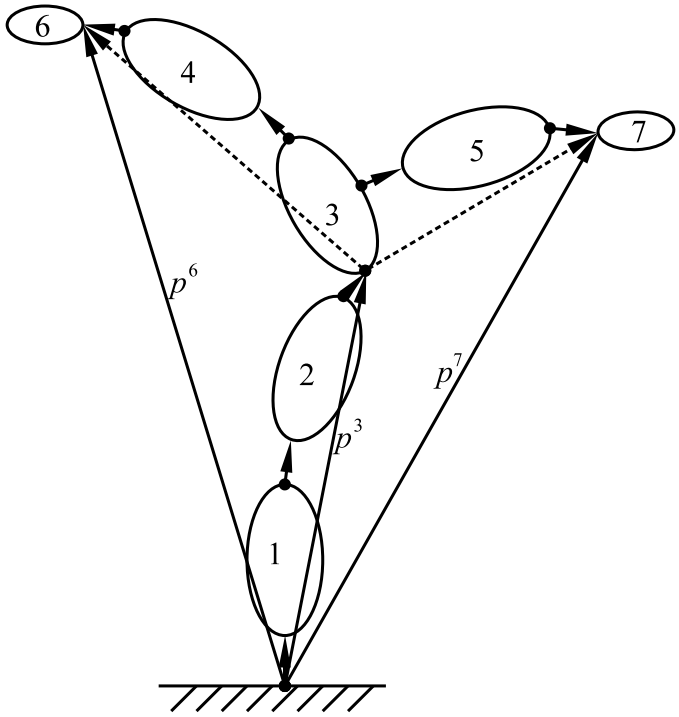


FIG. 1.13 – Structure arborescente

Il y a également une considération topologique à prendre en compte pour la comparaison des deux démarches. Il s'agit des cas, relativement fréquents[4], où on désire calculer les grandeurs cinématiques correspondant à plusieurs points situés sur une structure arborescente, comme illustré sur la figure 1.13.

Dans ce cas, en utilisant la première méthode, toutes les informations relatives au corps 3 ne doivent être calculées qu'une seule fois pour les deux branches menant aux corps 6 et 7, étant donné qu'elles sont exprimées par rapport à la base. Cette économie n'est pas possible dans la deuxième méthode, car les grandeurs relatives au corps 3 ne sont pas calculées explicitement.

[4] particulièrement dans le domaine des véhicules où on peut trouver sur une même chaîne cinématique des points d'ancrage d'amortisseur, des roues, etc.

Cet avantage de la première méthode est décisif pour l'application à des systèmes plus complexes dans lesquels ce genre de situation est classique.

1.2.2 Jacobienne

La matrice Jacobienne est un élément très utilisé dans l'étude des systèmes articulés. Il faut donc être attentif à la complexité des équations générées pour le calcul des éléments de cette matrice. Nous allons à nouveau examiner deux méthode de génération et les comparer en fonction de la complexité des équations obtenues.

La Jacobienne correspond aux dérivées partielles des vitesses linéaire et angulaire d'un repère par rapport aux vitesses généralisées. Les vitesses linéaires $\dot{\mathbf{p}}$ et angulaires $\dot{\boldsymbol{\omega}}$ étant des vecteurs, On parlera de Jacobienne vectorielle de translation, notée $\mathbf{J}_t$ et d'orientation notée $\mathbf{J}_o$. La matrice Jacobienne est alors la projection de ces Jacobiennes vectorielles dans un repère donné. Elle joue un rôle essentiel pour le traitement des équations de contraintes provenant des boucles cinématiques présentes dans beaucoup de systèmes.

Les Jacobiennes vectorielles de translation et d'orientation du repère solidaire du corps i sont définie par la relation suivante :

$$\mathbf{J}_t^{ij} \triangleq \frac{\partial \dot{\mathbf{p}}^i}{\partial \dot{q}^j}$$
$$\mathbf{J}_o^{ij} \triangleq \frac{\partial \boldsymbol{\omega}^i}{\partial \dot{q}^j}$$

ainsi :

$$\begin{pmatrix} \dot{\mathbf{p}}^i \\ \boldsymbol{\omega}^i \end{pmatrix} = \sum_j \begin{pmatrix} \mathbf{J}_t^{ij} \\ \mathbf{J}_o^{ij} \end{pmatrix} \dot{q}^j$$

Le développement des expressions des Jacobiennes vectorielles n'est pas trivial. Écrivons les expressions des vitesses linéaire et angulaire absolues du corps i :

$$\dot{\mathbf{p}}^f = \sum_{i:i\leq f} (\tilde{\boldsymbol{\omega}}^h \mathbf{d}_z^{hi} + \boldsymbol{\psi}^i \dot{q}^i) \tag{1.19}$$

$$\boldsymbol{\omega}^f = \sum_{i:i\leq f} \boldsymbol{\varphi}^i \dot{q}^i \tag{1.20}$$

L'expression de $\dot{\mathbf{p}}^f$ nécessite quelques manipulations de manière à mettre en évidence les vitesses généralisées et ainsi, faire apparaître les termes de la Jacobienne :

$$\begin{aligned}
\dot{\mathbf{p}}^f &= \sum_{i:i\leq f}\Big(\sum_{j:j<i}\tilde{\boldsymbol{\Omega}}^j.\mathbf{d}_z^{hi}\Big)+\sum_{i\leq f}(\boldsymbol{\psi}^i\dot{q}^i)\\
&= \sum_{j:j<f}(\tilde{\boldsymbol{\Omega}}^j\cdot\sum_{i:j<i\leq f}\mathbf{d}_z^{hi})+\sum_{i\leq f}(\boldsymbol{\psi}^i\dot{q}^i)\\
&= \sum_{j:j<f}(\tilde{\boldsymbol{\Omega}}^j\cdot\sum_{i:j<i\leq f}\mathbf{d}_z^{hi}+\boldsymbol{\psi}^j\dot{q}^j)+\boldsymbol{\psi}^f\dot{q}^f\\
&= \sum_{j:j<f}((\tilde{\boldsymbol{\varphi}}^j\cdot\sum_{i:j<i\leq f}\mathbf{d}_z^{hi}+\boldsymbol{\psi}^j)\,\dot{q}^j)+\boldsymbol{\psi}^f\dot{q}^f
\end{aligned}$$

avec : $\sum_{i:j<i\leq f}\mathbf{d}_z^{hi}=\mathbf{p}^{jf}$ et $h=inbody(i)$

Il est alors aisé d'écrire l'expression des Jacobiennes vectorielles de translation et d'orientation relative au repère du corps f :

$$\mathbf{J}_t^{fj}=\tilde{\boldsymbol{\varphi}}^j\sum_{i:j<i\leq f}\mathbf{d}_z^{hi}+\boldsymbol{\psi}^j=\tilde{\boldsymbol{\varphi}}^j.\mathbf{p}^{jf}+\boldsymbol{\psi}^j \tag{1.21}$$

$$\mathbf{J}_o^{fj}=\boldsymbol{\varphi}^j \tag{1.22}$$

Si l'expression de $\mathbf{J}_o$ n'appelle aucun commentaire particulier vu sa simplicité, il n'en est pas de même de l'expression de $\mathbf{J}_t$. En effet, lors des manipulations destinées à mettre en évidence les vitesses généralisées $\dot{q}^j$, on voit apparaître le terme $\sum_{i:j<i\leq f}\mathbf{d}_z^{hi}$ qui n'est autre que $\mathbf{p}^{jf}$ le vecteur position relative de l'origine du repère du corps final f par rapport à l'origine du repère du corps j, un vecteur qui apparaît naturellement dans l'approche *régressive* développée précédemment. Dans [95], ce résultat est obtenu par application du principe de superposition des vitesses.

Néanmoins, deux caractéristiques de cette méthode la rendent inutilisable telle quelle pour le calcul de la Jacobienne :

- considérant d'une part que la méthode s'applique aux composantes de la Jacobienne vectorielle projetée dans un repère donné, le fait que les composantes des vecteurs $\mathbf{p}^{jf}$ ne sont pas toutes évaluées dans le même repère nécessite que ceux-ci soient projetés dans un repère commun et donc que la méthode soit modifiée en conséquence. On dispose en fait des matrices de rotations permettant de recalculer les composantes des vecteurs connues dans les repères locaux et de les exprimer dans le repère du corps final. A la dernière étape, il suffirait alors de projeter à nouveau toute la Jacobienne dans le repère de base, qui est généralement le repère de travail pour la plupart des applications.

- d'autre part, dans le cas des systèmes à structure arborescente et pas simplement linéaire, la méthode régressive n'est pas la plus indiquée si on désire connaître les Jacobiennes de plusieurs repères sur la structure.

Nous allons donc d'une part, extraire des développements précédents une expression récursive permettant de calculer la Jacobienne de translation par le première méthode progressive, et d'autre part présenter une méthode hybride, proposée dans [13], semi-progressive et semi-régressive éliminant quelques uns des désavantages des deux méthodes précédentes.

Méthode progressive

L'extraction d'une formulation récursive de la Jacobienne vectorielle $\mathbf{J}_t$ est assez directe sur base de (1.21) :

$$\mathbf{J}_t^{ij} = \begin{cases} \mathbf{J}_t^{hj} + \tilde{\boldsymbol{\varphi}}^h \mathbf{d}_z^{hi} & \text{si } j < i \\ \boldsymbol{\psi}^i & \text{si } j = i \end{cases} \tag{1.23}$$

Cette formulation récursive est celle qui est proposée dans [12].

Ainsi, l'évaluation de la matrice Jacobienne par la méthode progressive exprimée dans le repère solidaire du corps de base est développée dans le schéma suivant :

Initialisation :

$R^{0,0} = E$, la matrice identité
$J_o^{00} = J_t^{00} = 0$, un triplet nul

Récursion :

For $j = 1 : n$

$i = ch(j)$
$h = inbody(i)$
$R^{0,i} = R^{0,h} R^{h,i}$
$rl^h = R^{0,h} d_z^{hi}$

For $k = 1 : j - 1$

$J_o^{ik} = J_o^{hk}$

end

$J_o^{ij} = R^{0,h} \varphi^i$

For $k = 1 : j - 1$

$$J_t^{ik} = J_t^{hk} + \tilde{J_o^{ik}}\, rl^h$$

end

$$J_t^{ij} = R^{0,h}\psi^i$$

end.

On notera que :

- à chaque étape on calcule les composantes de la Jacobienne du repère du corps courant, lesquelles sont réutilisées à l'étape suivante,
- dans le cas de la matrice Jacobienne d'orientation J_o, il n'y a en fait pas de calcul à faire si on considère que la matrice de rotation $R^{0,h}$ et que la multiplication $R^{0,h}\varphi^i$ n'est en fait qu'une sélection d'une ligne [5] de cette matrice,
- dans le cas de la matrice Jacobienne de translation J_t, ce calcul requiert plusieurs opérations $(\tilde{J_o^{ik}} rl)_{k<j}$ à chaque étape, ce qui conduit à une complexité globale de l'ordre de $O(n^2/2)$ [12] pour le calcul de la Jacobienne d'un repère situé à l'extrémité d'une chaîne de longueur n.

Méthode hybride

En observant l'équation (1.21), on réalise qu'on peut calculer $\mathbf{J}_t$ en utilisant une formule non récursive quasi aussi simple que pour $\mathbf{J}_o$, ce qui est naturellement plutôt intéressant du point de vue du nombre d'opérations requises. Pour cela, il faut toutefois disposer du vecteur position relative $\mathbf{p}^{jf}$ pour le calcul de chaque colonne j de la matrice Jacobienne. Ce vecteur peut être facilement obtenu récursivement par une méthode basée sur un parcours régressif comme nous l'avons montré dans la section précédente. Toutefois, dans la méthode purement régressive, les composantes p^{jf} du vecteur qui nous intéresse sont exprimées dans le repère local à chaque étape, comme nous l'avons fait remarquer ci-dessus.

De manière à obtenir ces composantes dans le repère de base à chaque étape, nous allons procéder en deux phases. Tout d'abord nous effectuons un parcours progressif de la chaîne cinématique de manière à générer les matrices de rotations nécessaire pour la projection des vecteurs articulaires locaux dans le repère de base, comme c'est le cas dans la méthode progressive. Le parcours régressif est alors fait dans la seconde phase, lors de laquelle sont calculées les composantes p^{jf} exprimées dans le repère de base, ainsi que les colonnes de la matrice Jacobienne de translation J_t.

[5] car le triplet φ^i ne contient qu'une seule valeur non nulle, laquelle vaut l'unité, selon les conventions de modélisation de ROBOTRAN

Le schéma de calcul hybride proposé est alors le suivant :

Initialisation :

$$R^{0,0} = E, \text{ la matrice identité}$$
$$p^0 = \dot{p}^0 = \omega^0 = \dot{\omega}^0 = 0$$

Parcours progressif :

For $j = 1 : n$

$$i = ch(j)$$
$$h = inbody(i)$$
$$R^{0,i} = R^{0,h} R^{h,i}$$
$$J_o^i = R^{0,i} \varphi^i$$

end.

Parcours régressif :

For $j = n : 1$

$$k = n - j + 1$$
$$i = ch(k)$$
$$h = inbody(i)$$
$$rl^h = R^{0,h} d_z^{hi}$$
$$p^{h,f} = p^{i,f} + rl^h$$
$$J_t^i = R^{0,h} \psi^i - \tilde{p}^{i,f} (R^{0,h} \varphi^i)$$

end.

On notera que :
- la complexité du calcul de J_t est maintenant d'ordre $O(n)$!
- les colonnes de la matrice J_o peuvent indifféremment être calculées dans la première ou la seconde étape, tout comme les variables auxiliaires rl^h.

Discussion

Ici aussi, nous avons généré symboliquement les équations nécessaires au calcul des Jacobiennes relatives à un repère situé à l'extrémité du même bras manipulateur série à cinq degrés de liberté. Les nombres d'opérations en virgule flottante nécessaires à l'évaluation de ces équations sont repris dans la table 1.2.

On constate que la méthode hybride d'ordre $O(n)$ est plus économique, comme on l'avait déjà observé au niveau des équations vectorielles. Dans la cas

		Progressif	Mixte
1	$p,\ J,$	117	100
2	$p,\ \dot{p},\ \omega,\ J,$	164	149

TAB. 1.2 – Comparaison des méthodes en terme de nombre d'opérations

de l'exemple considéré, le gain est de l'ordre de 15%. La différence sera d'autant plus importante que la chaîne cinématique considérée sera plus longue.

Toutefois, la remarque, déjà faite dans la discussion précédente et concernant le cas de l'évaluation de plusieurs Jacobiennes relatives à différents repères d'une grande structure linéaire ou arborescentes reste d'application ici. La caractéristique de la première méthode qui produit naturellement toutes les Jacobiennes relatives aux repères intermédiaires (et toutes exprimées dans le même repère de base) constitue un avantage indéniable.

Ainsi, on préconisera l'utilisation de la première méthode basée sur un parcours progressif dans le cas de structures complexes arborescentes.

La seconde méthode reste néanmoins avantageuse dans les applications impliquant un système ayant une topologie linéaire ou arborescente avec une branche commune courte ou si on désire calculer la matrice Jacobienne d'un seul repère, ce qui est typiquement le cas pour les modèles dynamiques de contrôle de robots.

1.3 Génération des équations du mouvement

1.3.1 Méthode récursive de Newton-Euler

Pour obtenir les équations du mouvement sous forme récursive, nous allons nous baser sur les équations de Newton-Euler (9) et (10).

Pour cela nous devons connaître les accélérations linéaires et angulaires des corps. En nous basant sur la figure 1.14 et en réutilisant les équations relatives à la cinématique du corps i, nous pouvons écrire les expressions suivantes relatives aux position, vitesses et accélérations de son centre de masse G^i. Celui-ci est localisé dans le repère $\{O'^i, \{\hat{\mathbf{X}}^\mathbf{i}\}\}$ solidaire du corps i par le vecteur $\mathbf{d}^{ii} = \overrightarrow{O'^i G^i}$, dont les composantes sont constantes dans $\{\hat{\mathbf{X}}^i\}$.

$$\mathbf{x}^i = \mathbf{p}^i + \mathbf{d}^{ii}, \tag{1.24}$$

$$\dot{\mathbf{x}}^i = \dot{\mathbf{p}}^i + \widetilde{\boldsymbol{\omega}}^i \,.\, \mathbf{d}^{ii} \tag{1.25}$$

$$\ddot{\mathbf{x}}^i = \ddot{\mathbf{p}}^i + (\dot{\widetilde{\boldsymbol{\omega}}}^i + \widetilde{\boldsymbol{\omega}}^i \, \widetilde{\boldsymbol{\omega}}^i) \, \mathbf{d}^{ii} \tag{1.26}$$

Écrivons les équations du mouvement d'un corps i sous forme vectorielle en nous basant sur la figure 1.15.

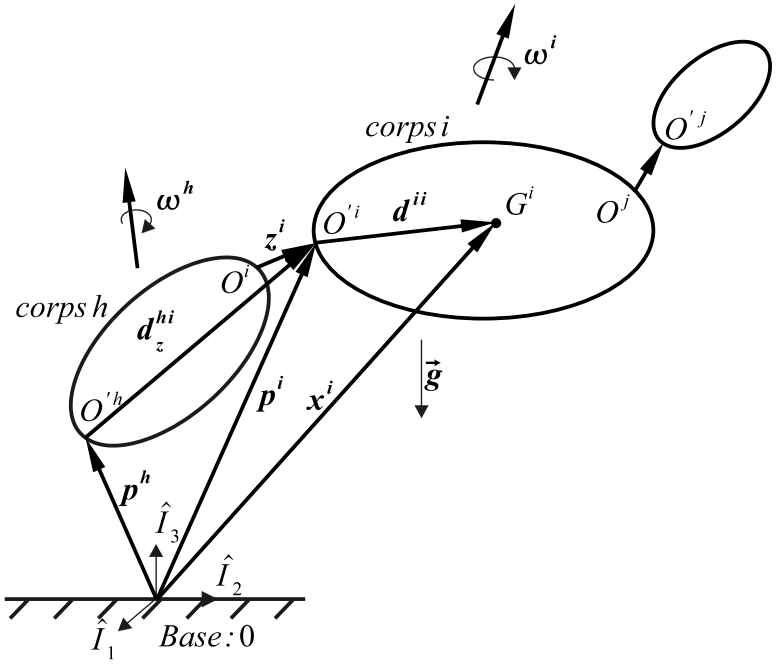


FIG. 1.14 – Corps i — cinématique

Équations de translation du corps i, sous forme vectorielle :

$$\mathbf{F}^i - \sum_{j \in \bar{\imath}} \mathbf{F}^j + \mathbf{F}^i_{ext} + m^i \mathbf{g} = m^i \ddot{\mathbf{x}}^i \tag{1.27}$$

où $\mathbf{F}^i$, defini dans (1.1.4), représente la force (résultante) appliquée par l'articulation i, sur le corps i au point d'ancrage O'^i de celui-ci sur son parent[6], comme l'indique la figure 1.15, $\mathbf{F}^i_{ext}$ est la résultante des forces extérieures (hormis la gravité $\mathbf{g}$) appliquée au centre de masse G^i du corps i, et $\sum_{j \in \bar{\imath}}$ représente une somme sur l'ensemble des corps enfants attachés au corps i (j et k sur la figure 1.15).

Afin d'écrire l'équation (1.27) de façon plus compacte, il est intéressant de définir les grandeurs suivantes :

[6]On notera que ces conventions adoptées pour l'implémentation dans ROBOTRAN sont différentes de celles qui sont utilisées dans [13]

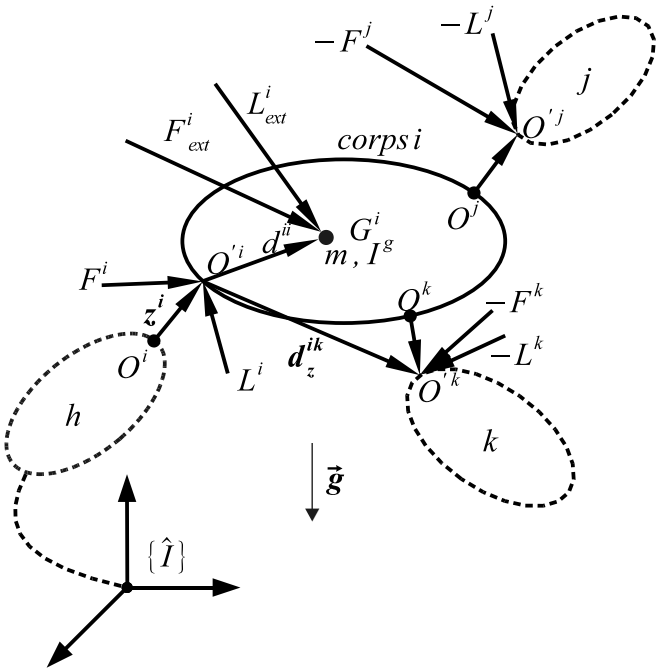


FIG. 1.15 – Corps i — dynamique

$$\boldsymbol{\beta}^i \triangleq \widetilde{\dot{\boldsymbol{\omega}}}^i + \widetilde{\boldsymbol{\omega}}^i \, \widetilde{\boldsymbol{\omega}}^i \tag{1.28}$$

$$\boldsymbol{\alpha}^i \triangleq \ddot{\mathbf{p}}^i - \mathbf{g} \tag{1.29}$$

avec $\mathbf{g}$, le vecteur gravité.

En utilisant (1.12), nous écrivons la relation (1.29) de façon récursive :

$$\boldsymbol{\alpha}^i = \boldsymbol{\alpha}^h + \boldsymbol{\beta}^h \, \mathbf{d}^{hi}_z + 2\,\widetilde{\boldsymbol{\omega}}^i \, \boldsymbol{\psi}^i \dot{q}^i + \boldsymbol{\psi}^i \ddot{q}^i \tag{1.30}$$

Ces définitions nous permettent d'écrire (1.26) sous la forme :

$$\ddot{\mathbf{x}}^i - \mathbf{g} = \boldsymbol{\alpha}^i + \boldsymbol{\beta}^i \, \mathbf{d}^{ii} \tag{1.31}$$

En utilisant l'équation (1.31), on peut réécrire (1.27) comme ceci :

$$\mathbf{F}^i = \sum_{j \in \bar{\imath}} \mathbf{F}^j + m^i \, (\boldsymbol{\alpha}^i + \boldsymbol{\beta}^i \, \mathbf{d}^{ii}) - \mathbf{F}^i_{ext}$$

ou encore

$$\begin{aligned} \mathbf{F}^i &= \sum_{j \in \overline{\imath}} \mathbf{F}^j + \mathbf{W}^i \\ \text{avec} \quad \mathbf{W}^i &\triangleq m^i \left(\boldsymbol{\alpha}^i + \boldsymbol{\beta}^i \, \mathbf{d}^{ii}\right) - \mathbf{F}^i_{ext} \end{aligned} \tag{1.32}$$

De même, les équations de rotation du corps i par rapport à son centre de masse G^i (équation d'Euler (10)) s'écrit :

$$\mathbf{L}^i - \sum_{j \in \overline{\imath}} \mathbf{L}^j + \mathbf{L}^i_{ext} - \widetilde{d}^{ii} \, \mathbf{F}^i - \sum_{j \in \overline{\imath}} (\widetilde{\mathbf{d}}^{ij}_z - \widetilde{\mathbf{d}}^{ii}) \, \mathbf{F}^j = \mathbf{I}^i \, \dot{\boldsymbol{\omega}}^i + \widetilde{\boldsymbol{\omega}}^i \, \mathbf{I}^i \, \boldsymbol{\omega}^i \tag{1.33}$$

La définition de $\mathbf{W}^i$ nous permet d'écrire de façon plus concise l'équation de rotation du corps i par rapport à son centre de masse comme ceci :

$$\mathbf{L}^i = \sum_{j \in \overline{\imath}} \{\mathbf{L}^j + \widetilde{\mathbf{d}}^{ij}_z \; \mathbf{F}^j\} + \widetilde{\mathbf{d}}^{ii} \; \mathbf{W}^i - \mathbf{L}^i_{ext} + \mathbf{I}^i \, \dot{\boldsymbol{\omega}}^i + \widetilde{\boldsymbol{\omega}}^i \, \mathbf{I}^i \, \boldsymbol{\omega}^i \tag{1.34}$$

Les forces généralisées articulaires Q^i sont obtenues par projection sur les axes des articulations :

$$Q^i = \mathbf{F}^i \; \boldsymbol{\psi}^i + \mathbf{L}^i \; \boldsymbol{\varphi}^i \tag{1.35}$$

Les équations (1.32) et (1.34) pourront être générées de façon récursive, partant des corps terminaux et en remonant vers la racine du système. Cela correspond au parcours récursif inverse du formalisme de Newton-Euler, qui produit la forme implicite (1.36) du modèle *dynamique inverse* (après projection des grandeurs sur les axes des articulations).

1.3.2 Formulation implicite

$$Q = \Phi(q, \dot{q}, \ddot{q}, ...) \tag{1.36}$$

Cinématique

Le schéma de calcul sous forme vectorielle peut-être consulté dans [13]. La programmation de ce schéma de calcul requiert que les expressions soient exprimées en terme de composantes dans les repères des corps i à chaque étape. On utilisera alors la forme matricielle suivante :

Initialisation :

$$\alpha^0 = -g \ , \ \omega^0 = \dot{\omega}^0 = 0$$

Récursion :
For $i = 1 : N^{body}$

$$\begin{aligned}
h &= \text{inbody}(i) \\
\omega^i &= R^{i,h}\,\omega^h + \varphi^i\,\dot{q}^i \\
\dot{\omega}^i &= R^{i,h}\left(\dot{\omega}^h + \tilde{\omega}^i\,\varphi^i\,\dot{q}^i\right) + \varphi^i\,\ddot{q}^i \\
\beta^i &= \tilde{\dot{\omega}}^i + \tilde{\omega}^i\tilde{\omega}^i \\
\alpha^i &= R^{i,h}(\alpha^h + \beta^h\,d_z^{hi}) + 2\,\tilde{\omega}^i\,\psi^i\,\dot{q}^i + \psi^i\,\ddot{q}^i
\end{aligned}$$

end.

Dynamique

La formulation récursive sous forme vectorielle découle directement des équations (1.32) et (1.34) et est également reprise dans [13].

La programmation de ce schéma de calcul requiert que les expressions soient exprimées en terme de composantes dans les repères des corps i à chaque étape. On utilisera alors la forme matricielle suivante :

For $i = N^{body} : 1$

$$\begin{aligned}
W^i &= m^i(\alpha^i + \beta^i\,d^{ii} - F^i_{ext} \\
F^i &= \sum_{j\,\in\,\bar{\imath}} R^{i,j}\,F^j + W^i \\
L^i &= \sum_{j\,\in\,\bar{\imath}} \left(R^{i,j}\,L^j + \widetilde{d}_z^{ij}\,R^{i,j}\,F^j\right) \\
&\quad + \widetilde{d}^{ii}\,W^i - L^i_{ext} + I^i\dot{\omega}^i + \tilde{\omega}^i I^i\omega^i
\end{aligned}$$

end.

Les équations articulaires sont alors obtenues par projection sur les axes des articulations :

$$Q^i = \left(\psi^i\right)^T F^i + \left(\varphi^i\right)^T L^i$$

La projection consiste en pratique à choisir parmi les six composantes de force $\{F^i, L^i\}$ celle qui correspond à la direction et au type de l'articulation. Les cinq autres composantes appelées *composantes de réaction* sont reprises par la structure du système.

Nous illustrerons les conséquences de ces projections en terme de nombre d'opérations dans la section *Applications* de ce chapitre.

1.3.3 Formulation semi-explicite

En modifiant le schéma de calcul de Newton-Euler afin d'isoler les accélérations articulaires [17], il est possible d'obtenir la forme semi-explicite suivante des équations du mouvement :

$$M(q)\,\ddot{q} + c(q,\,\dot{q}, F_{ext}, L_{ext}, g) = Q \tag{1.37}$$

Il s'agit d'un système linéaire d'équations où apparaissent la matrice de masse généralisée $M(q)$, les accélérations généralisées articulaires $\ddot{q}$, le vecteur c qui contient les termes de Coriolis, gyroscopiques et de gravité, ainsi que les forces et couples externes et, les forces généralisées articulaires Q. Pour obtenir explicitement les accélérations, il suffit de résoudre ce système linéaire d'équations.

Cinématique

En isolant les accélérations dans les équations (1.11), (1.28) et 1.30, nous obtenons les nouvelles expressions de $\dot{\boldsymbol{\omega}}^i, \boldsymbol{\beta}^i$ et $\boldsymbol{\alpha}^i$:

$$\dot{\boldsymbol{\omega}}^i = \sum_{k:\,k\leq i} \mathbf{O}_M^{ik}\,\ddot{q}^k + \dot{\boldsymbol{\omega}}_c^i \tag{1.38}$$

$$\boldsymbol{\beta}^i = \sum_{k:\,k\leq i} \mathbf{B}_M^{ik}\,\ddot{q}^k + \boldsymbol{\beta}_c^i \tag{1.39}$$

$$\boldsymbol{\alpha}^i = \sum_{k:\,k\leq i} \mathbf{A}_M^{ik}\,\ddot{q}^k + \boldsymbol{\alpha}_c^i \tag{1.40}$$

La programmation de l'algorithme de génération requiert que les grandeurs vectorielles soient projetées dans les repères des corps auxquels elles se rapportent[7]. Ce qui conduit au schéma de calcul suivant [17] :

Initialisation :

$$\alpha_c^0 = -g\ ;\ \omega^0 = \dot{\omega}_c^0 = 0$$
$$O_M^{ik} = A_M^{ik} = 0 \quad (\forall\, i = 0 : N^{body}, \forall k = 0 : i)$$

Récursion :

[7] Par exemple : $\boldsymbol{\psi}^i = [\hat{\mathbf{X}}^i]^T \psi^i$, $\mathbf{d}_z^{hi} = [\hat{\mathbf{X}}^h]^T d_z^{hi}, \ldots$

For $i = 1 : N^{body}$

$$
\begin{aligned}
h &= \text{inbody}(i) \\
\omega^i &= R^{i,h}\,\omega^h + \varphi^i\,\dot{q}^i \\
\dot{\omega}_c^i &= R^{i,h}\,\dot{\omega}_c^h + \tilde{\omega}^i\varphi^i\,\dot{q}^i \\
\beta_c^i &= \tilde{\dot{\omega}}_c^i + \tilde{\omega}^i\tilde{\omega}^i \\
\alpha_c^i &= R^{i,h}\left(\alpha_c^h + \beta_c^h\,d_z^{hi}\right) + 2\,\tilde{\omega}^i\,\psi^i\,\dot{q}^i
\end{aligned}
$$

For $k = 1 : i$

$$
\begin{aligned}
O_M^{ik} &= R^{i,h}\,O_M^{hk} + \delta^{ki}\,\varphi^i \\
A_M^{ik} &= R^{i,h}\left(A_M^{hk} + \widetilde{O}_M^{hk}\,d_z^{hi}\right) + \delta^{ki}\,\psi^i
\end{aligned}
$$

où $\delta^{ki} = 1$ si $k = i$, 0 sinon

end

end.

Dynamique

Afin d'obtenir la matrice de masse M et le vecteur c (équation (1.37)), les expressions de $\mathbf{F}^i$ et $\mathbf{L}^i$ sont également modifiées pour y mettre en évidence les accélérations généralisées $\ddot{q}^k$:

$$\mathbf{F}^i = \sum_k \mathbf{F}_M^{ik}\,\ddot{q}^k + \mathbf{F}_c^i \tag{1.41}$$

$$\mathbf{W}^i = \sum_k \mathbf{W}_M^{ik}\,\ddot{q}^k + \mathbf{W}_c^i \tag{1.42}$$

$$\mathbf{L}^i = \sum_k \mathbf{L}_M^{ik}\,\ddot{q}^k + \mathbf{L}_c^i \tag{1.43}$$

Les termes portant les indices $_M$ et $_c$ sont les constituants, respectivement, de la matrice de masse M et du vecteur c. Ils peuvent être calculés de façon récursive en introduisant les relations (1.38), (1.39) et (1.40) dans les expressions de $\mathbf{F}^i$ (1.32) et de $\mathbf{L}^i$ (1.34).

Tout comme pour la partie cinématique, la programmation de l'algorithme de génération requiert que les différentes grandeurs soient projetées dans les repères des corps auxquels elles se rapportent. Nous considérons que F_{ext}^i et L_{ext}^i sont les composantes des forces et couples externes appliqués au corps i dans le repère $\{\hat{\mathbf{X}}^i\}$ de celui-ci.

Nous obtenons alors le schéma suivant :

For $i = N^{body} : 1$

$$W_c^i = m^i(\alpha_c^i + \beta_c^i(z^i + d^{ii}) - F_{ext}^i$$
$$F_c^i = \sum_{j \in \bar{\imath}} R^{i,j}\, F_c^j + W_c^i$$
$$L_c^i = \sum_{j \in \bar{\imath}} \left(R^{i,j}\, L_c^j + \tilde{d}_z^{ij}\, R^{i,j}\, F_c^j \right) + \tilde{d}_z^{ii}\, W_c^i - L_{ext}^i + I^i \dot{\omega}_c^i + \tilde{\omega}^i I^i \omega^i$$

For $k = 1 : i$[8]

$$W_M^{ik} = m^i \left(A_M^{ik} + O_M^{ik}\, d^{ii} \right)$$
$$F_M^{ik} = \sum_{j \in \bar{\imath}} \left(R^{i,j}\, F_M^{jk} + W_M^{ik} \right)$$
$$L_M^{ik} = \sum_{j \in \bar{\imath}} \left(R^{i,j}\, L_M^{jk} + \tilde{d}_z^{ij}\, R^{i,j}\, F_M^{jk} \right) + \tilde{d}_z^{ii}\, W_M^{ik} + I^i O_M^{ik}$$

end
end.

Tout comme pour la formulation implicite, nous obtenons finalement les équations du mouvement articulaires en projetant les équations vectorielles (1.41) et (1.43) sur les axes des articulations. Dans ce cas ci, nous obtenons les termes de l'équation i du système semi-explicite (1.37) en projetant les grandeurs F_c, L_c , F_M et L_M sur l'axe de l'articulation i :

$$c_{(i)} = \left(\psi^i\right)^T F_c^i + \left(\varphi^i\right)^T L_c^i \quad \forall i, \tag{1.44}$$
$$M_{(i,\,j)} = \left(\psi^i\right)^T F_M^{ij} + \left(\varphi^i\right)^T L_M^{ij} \quad \forall i \text{ et } \forall j : j \leq i. \tag{1.45}$$

Nous pouvons donc écrire l'équation i du mouvement du système (1.37) sous la forme suivante :

$$\sum_j M_{(i,\,j)}\, \ddot{q}^j + c_{(i)} = \mathcal{Q}_{(i)} \tag{1.46}$$

La matrice de masse généralisée M ainsi obtenue est symétrique et définie positive. Démontrer cette affirmation est assez difficile à cause de la nature récursive de la méthode utilisée [13]. Cette propriété de la matrice de masse va être mise à profit dans le choix de la méthode de résolution du système d'équations (1.37).

[8] par symétrie de la matrice de masse

1.3.4 Modèles dynamiques directs explicites

La génération explicites des expressions des accélérations[9] nécessite la résolution du système d'équations du mouvement (1.37) obtenu par l'utilisation de la formulation semi-explicite. Toutefois, lorsque la cinématique de certaines articulations est imposée, il n'est pas nécessaire de recalculer les valeurs des accélérations relatives à ces articulations, étant imposées a priori. Une procédure de réduction peut alors être appliquée avant résolution. En répartissant les coordonnées généralisées en deux ensembles q_l et q_c selon qu'elles sont *libres* ou *commandées*, on peut récrire le système d'équations (1.37) comme ceci :

$$\begin{pmatrix} M_{ll} & M_{lc} \\ M_{cl} & M_{cc} \end{pmatrix} \begin{pmatrix} \ddot{q}_l \\ \ddot{q}_c \end{pmatrix} + \begin{pmatrix} c_l \\ c_c \end{pmatrix} - \begin{pmatrix} Q_l \\ Q_c \end{pmatrix} = 0 \tag{1.47}$$

Les valeurs des accélérations commandées étant connues, on peut alors redéfinir une matrice de masse réduite M_r et un terme indépendant réduit F_r comme ceci :

$$M_r = M_{ll} \qquad \text{et} \qquad F_r = (c_l - Q_l) - M_{lc}\,\ddot{q}_c \tag{1.48}$$

Le système linéaire d'équations utilisé est alors le suivant :

$$M_r\,\ddot{q}_l + F_r = 0 \tag{1.49}$$

Résolution du système d'équations

Afin d'obtenir explicitement les accélérations généralisées libres $\ddot{q}_l$, nous utilisons une méthode directe de résolution de système linéaire d'équations. Nous procédons en deux étapes pour la résolution [18].

D'abord nous factorisons la matrice de masse en utilisant une méthode qui tire profit de sa symétrie. Deux méthodes classiques[18], largement utilisées peuvent être utilisées pour cela : la méthode de Cholesky ou la méthode LDL^T. Ces deux méthodes ont une complexité $O(n^3/6)$.

Ensuite nous résolvons le système d'équations par éliminations successives, en profitant de la forme triangulaire des matrices obtenues après factorisation.

Factorisation de Cholesky GG^T La décomposition de Cholesky peut être appliquée à toute matrice A symétrique et définie positive. Elle permet de calculer la matrice triangulaire inférieure G telle que $A = GG^T$. Dans l'algorithme que voici, le terme a_{ij} est remplacé par g_{ij} pour $i \geq j$.

For $k = 1 : n$

$$a_{kk} = (a_{kk} - \sum_{p=1}^{k-1} a_{kp}^2)^{1/2}$$

[9] lesquelles sont fournies directement lorsqu'on utilise des formalismes $O(N)$.

For $i = k+1 : n$

$$a_{ik} = (a_{ik} - \sum_{p=1}^{k-1} a_{ip}\, a_{kp})/a_{kk}$$

end

end

Factorisation LDL^T Cette méthode ne nécessite pas d'extraction de racine carrée. Elle permet de calculer les matrices L et $D = diag(d_1, \ldots, d_n)$ telles que $A = LDL^T$. Dans l'algorithme que voici, le terme a_{ij} est remplacé par l_{ij} pour $i \geq j$ et par d_i si $i = j$.

For $k = 1 : n$

For $p = 1 : k-1$

$$r_p = d_p\, a_{kp}$$

end

$$d_k = a_{kk} - \sum_{p=1}^{k-1} a_{kp}\, r_p$$

For $i = k+1 : n$

$$a_{ik} = (a_{ik} - \sum_{p=1}^{k-1} a_{ip}\, r_p)/d_k$$

end

end

Dans l'implémentation de ce schéma, des variables auxiliaires sont créées pour contenir les inverses des termes diagonaux d_k et les n divisions par d_k sont remplacées par des multiplications par l'inverse de d_k. Ceci ne réduit pas le nombre d'opérations mais seulement le nombre de divisions, lesquelles sont des opérations plus coûteuses que toutes les autres opérations arithmétiques.

Résolution La résolution d'un système d'équations $Ly = b$ dont la matrice L a une structure triangulaire inférieure et une taille $n \times n$ peut se faire selon la méthode de *substitution directe* que voici [18] :

For $i = 1 : n$
  $y_i = b_i$
  For $j = 1 : i - 1$
    $y_i = y_i - l_{ij}\, y_j$
  end
  $y_i = y_i / l_{ii}$
end

En pratique, les termes l_{ii} provenant d'une factorisation LDL^T ou LU sont unitaires, la division par l_{ii} est donc inutile.

La résolution d'un système d'équations $Ux = y$ dont la matrice U a une structure triangulaire supérieure et une taille $n \times n$ peut se faire selon la méthode de *substitution inverse* que voici [18] :

For $i = n : 1$
  $x_i = y_i$
  For $j = i + 1 : n$
    $x_i = x_i - u_{ij}\, x_j$
  end
  $x_i = x_i / u_{ii}$
end

Lorsque la factorisation LDL^T est utilisée, les termes u_{ii} de la matrice U sont en réalité les termes d_k de la matrice diagonale D. Dès lors la division par u_{ii} est remplacée par une multiplication par l'inverse de d_k qui a été stocké dans une variable auxiliaire lors de l'étape de factorisation.

Ces deux éliminations ont chacune une complexité de l'ordre de $O(n^2/2)$ opérations, soit $O(n^2)$ pour les deux.

n	LDL^T	Cholesky
5	113	110
10	628	595
20	3858	3690
50	49148	47975
100	363298	358450
200	2786645	2766900

TAB. 1.3 – Comparaison des méthodes en terme de nombre d'opérations

Discussion Afin de comparer les deux méthodes de factorisation, nous avons généré avec ROBOTRAN la solution symbolique d'un système d'équations linéaires de type $Ax = b$. Les chiffres repris dans la table 1.3 correspondent aux nombres d'opérations à effectuer pour factoriser la matrice A et pour résoudre le système $LDL^T x = b$ ou $GG^T x = b$ respectivement. On remarque que le nombre total d'opérations de la méthode de Cholesky n'est inférieur que de quelques pour cent pour des matrices de taille $N \leq 200$, cependant elle nécessite N divisions et N extractions de racine carrée de plus que la méthode LDL^T. Cela constitue un léger désavantage pour des petits systèmes. En effet les divisions et les extractions de racines carrées sont des opérations relativement plus lourdes que les multiplications ou les additions.Hormis ces petites différences caractéristiques, il n'y a pas d'arguments décisifs, en termes de comportement numérique par exemple, permettant de distinguer ces méthodes, leur complexité théorique $O(n^3/6)$ étant identique.

Ces deux méthodes sont disponibles dans ROBOTRAN mais par défaut, nous utilisons la méthode LDL^T pour factoriser la matrice de masse M, résoudre le système d'équations du mouvement et, obtenir explicitement les accélérations généralisées $\ddot{q}$.

1.4 Forces articulaires

Dans l'écriture (1.37) du système d'équations du mouvement que nous utilisons, le terme Q qui se trouve dans le membre de droite correspond au vecteur des forces généralisées articulaires. Ces forces articulaires peuvent provenir d'éléments actifs ou passifs localisés dans les articulations.

Qu'il s'agisse des couples développés par les actionneurs d'un robot, ou des forces élastiques ou de frottement dans des articulations non idéales, le calcul de ces efforts nécessite la plupart du temps l'évaluation d'un modèle spécifique à fournir par l'utilisateur du modèle généré par ROBOTRAN.

Nous proposons donc d'insérer parmi les équations du modèle, une équation $Q = fctext(q, \dot{q}, t, \ldots)$ qui fait appel à une routine externe. Celle-ci sert alors d'interface entre les équations dynamiques du système mécanique et les équations nécessaires au calcul des efforts articulaires. Cette routine reçoit entre autres arguments les valeurs des coordonnées et des vitesses articulaires, ainsi que la valeur du temps de simulation. Elle renvoie un tableau Q qui contient toutes les valeurs des forces articulaires du système.

Cette technique est également utilisée pour l'introduction aisée des forces extérieures appliquées sur des corps du système.

1.5 Forces extérieures

Dans cette section nous allons nous intéresser aux forces (et couples) extérieures qui peuvent être appliquées sur chaque corps, et dont la résultante est notée $\mathbf{F}_{ext}$ ($\mathbf{L}_{ext}$) dans les équations de la dynamique 1.37. Ces forces peuvent provenir d'une interaction entre le système et son environnement ou entre différents corps du système.

Les cas de contacts avec l'environnement sont innombrables, mais citons tout de même les véhicules qui roulent sur le sol, les robots manipulateurs ou marcheurs qui touchent leur espace de travail avec leur préhenseur ou leurs pattes. Dans certains cas l'interaction peut avoir lieu sans contact, comme pour les ailes d'un avion ou les pales d'un hélicoptère.

Un cas typique d'interaction entre différents corps d'un système est celui des suspensions de véhicule. En effet, il arrive souvent que les éléments de suspensions, le ressort et l'amortisseur, ne soient pas modélisés comme des corps du système, mais uniquement comme des éléments qui transmettent des forces sur le châssis et sur un bras de suspension du véhicule.

Quelle que soit la nature de l'interaction, la force appliquée sur le corps dépend généralement de l'évaluation d'un modèle constitutif dont l'évaluation nécessite la connaissance des conditions d'interaction.

Nous proposons ici de calculer ces conditions d'interaction, sur base de grandeurs cinématiques relatives aux corps impliqués. Nous proposons trois interfaces permettant de traiter la plupart des types d'interactions rencontrés en pratique dans l'étude des systèmes articulés.

1. Nous considérons tout d'abord le cas général où un point d'application peut être identifié (ou choisi), nous proposons de fournir toutes les informations cinématiques relatives à ce point d'application.
2. Nous considérons ensuite le cas des interactions entre deux corps dans laquelle une ligne d'application de la force est définie par deux points d'applications appartenant à chacun des corps impliqués.
3. Finalement, nous considérons le cas particulier des systèmes ayant une ou plusieurs roues en contact avec le sol. La cinématique de la roue est alors un élément indispensable à l'évaluation du modèle constitutif du contact roue/sol.

Dans la plupart des cas les composantes des forces et couples extérieurs fournies par l'évaluation du modèle constitutif sont exprimées dans un repère différent de celui du corps. La connaissance de la cinématique du corps impliqué et du repère de référence dans lequel sont exprimées les forces et couples extérieurs, nous permet de calculer les composantes dans le repère du corps et éventuellement les couples de transport vers le centre de gravité, pour satisfaire aux conventions utilisées.

1.5.1 Forces de contact

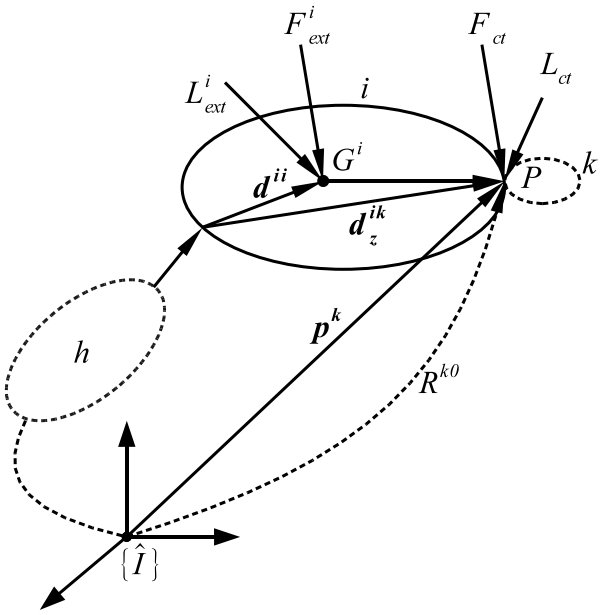


FIG. 1.16 – Force appliquée sur un corps en un point

Nous traitons ici du cas où un corps i est soumis à une force extérieure de contact F_{ct} (et éventuellement un couple L_{ct}) appliquée en un point d'application P identifiable dont les coordonnées sont connues ou peuvent être calculées dans le repère du corps i. La situation correspondante est illustrée sur la figure 1.16. Nous considérons dans ce cas que les composantes des forces et couples éventuels appliqués au corps i sont fournies par l'exécution d'une fonction externe qui implémente un modèle constitutif de l'interaction entre le corps et le monde extérieur. On considère que ces composantes seront exprimées dans le repère de base. Dans ce cas nous proposons de :

1. calculer toutes les grandeurs cinématiques absolues $\mathbf{p}^k$, $R^{k,0}$, $\dot{\mathbf{p}}^k$, $\boldsymbol{\omega}^k$, etc. relatives au point d'application *pa* identifié, qui sont nécessaires à l'évaluation du modèle constitutif de l'interaction. La méthode progressive présentée dans la section 1.2.1 sera utilisée. Notons qu'un corps fictif k sera défini à la position $\mathbf{d}^{ik}_z$ correspondant au point d'application selon la convention définie pour le calcul des grandeurs cinématiques. La matrice de rotation R^k est égale à la matrice R^i de rotation du corps i.

2. évaluer la fonction externe qui calcule les composantes de forces et couples appliquées sur le corps. En regroupant les composantes de forces et de couples dans une même entité Wr on peut écrire une équation correspondant à l'évaluation du modèle de contact implémenté dans une fonction externe :

$$Wr = fctext(p^k, R^{k,0}, \dot{p}^k, \omega^k, \ldots, t) \tag{1.50}$$

avec $Wr = \{F_1, F_2, F_3, L_1, L_2, L_3\}$ où F_1 est la composante de F_{ct} selon $\hat{\mathbf{I}}_1$, etc.

3. finalement, calculer les composantes F^i_{ext} et L^i_{ext} des forces et couples extérieurs équivalents appliquées sur le corps i en son centre de gravité, en tenant compte du transport de la force du point d'application P au centre de gravité du corps i :

$$F^i_{ext} = R^{k,0}\ F_{ct} \tag{1.51}$$

$$L^i_{ext} = R^{k,0}\ L_{ct} + (d^{ik}_z - d^{ii}) \times F^i_{ct} \tag{1.52}$$

1.5.2 Forces de liaison point à point

Nous considérons ici le cas de deux corps reliés par un élément actif ou passif. Celui-ci applique une force sur chacun des corps i et j aux points d'ancrage respectivement A et B. Cette force est d'ailleurs appliquée selon la direction de la droite joignant les points d'ancrage.

La configuration générale du système est décrite sur la figure 1.17

Les vecteurs positions des points A et B peuvent être calculés en utilisant les schémas récursifs présentés dans la section 1.2.1.

Le vecteur $\mathbf{z}$ et sa dérivée temporelle $\dot{\mathbf{z}}$ peuvent être calculés facilement :

$$\mathbf{z} = \overrightarrow{BA} = \mathbf{p}^a - \mathbf{p}^b = \mathbf{p}^{ca} - \mathbf{p}^{cb}$$

$$\dot{\mathbf{z}} = \dot{\overrightarrow{BA}} = \dot{\mathbf{p}}^a - \dot{\mathbf{p}}^b = \dot{\mathbf{p}}^{ca} - \dot{\mathbf{p}}^{cb}$$

On peut calculer le vecteur $\mathbf{z}$ soit en utilisant les vecteurs positions absolues $\mathbf{p}^a$ et $\mathbf{p}^b$ des points A et B, soit en utilisant leurs positions relatives par rapport au point de référence O'^c d'un corps c défini comme le dernier ancêtre commun des corps i et j. Il peut être différent du corps de base si les chaînes cinématiques ont une branche commune, comme c'est le cas sur la figure 1.17. La décision concernant le choix le plus approprié du corps de référence sera prise en fonction de l'ensemble des caractéristiques du système étudié.

1. Nous calculons la distance Z entre les points d'ancrage A et B ainsi que la variation temporelle $\dot{\mathsf{Z}} = \dot{\mathbf{z}} \cdot \mathbf{z}/\mathsf{Z}$ de cette distance.

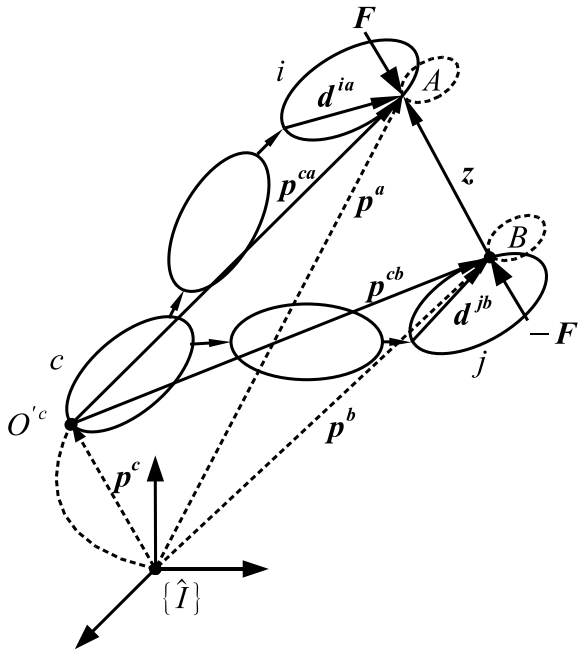


FIG. 1.17 – Forces de liaison entre deux corps

2. Pour calculer l'intensité scalaire de la force F, on fait généralement appel à une routine extérieure. Nous désignons ici $fctext$ cette routine qui évalue les équations constitutives correspondantes. Outre la longueur Z et sa dérivée $\dot{\mathsf{Z}}$, le modèle constitutif de l'interaction peut nécessiter d'autres paramètres comme la valeur du temps t ou la valeur de variables d'états x_z relatives à la dynamique interne de l'élément de liaison, ou encore des valeurs de commande u_z dans le cas d'un élément actif.

$$\mathsf{F} = fctext(\mathsf{Z}, \dot{\mathsf{Z}}, t, x_z, u_z) \tag{1.53}$$

3. Il reste finalement à calculer les forces et couples de transport à appliquer au centre de gravité des corps i et j pour prendre en compte leur interaction.

$$\mathbf{F} = \mathsf{F}\frac{\mathbf{z}}{\mathsf{Z}} \tag{1.54}$$

$$\mathbf{F}^i_{ext} = \mathbf{F} \qquad \mathbf{L}^i_{ext} = (\mathbf{d}^{ia}_z - \mathbf{d}^{ii}) \times \mathbf{F}^i_{ext} \tag{1.55}$$

$$\mathbf{F}^j_{ext} = -\mathbf{F} \qquad \mathbf{L}^j_{ext} = (\mathbf{d}^{jb}_z - \mathbf{d}^{jj}) \times -\mathbf{F}^j_{ext} \tag{1.56}$$

Dans certains cas, les équations constitutives sont relativement simples et pourraient être introduites directement parmi les autres équations du modèle plutôt que de faire appel à une routine extérieure. Toutefois, cela nécessiterait de proposer une librairie de modèles constitutifs, ou de proposer un module symbolique multidisciplinaire permettant de générer des modèles d'éléments de liaisons. Nous ne présentons pas les différentes possibilités dans le cadre de ce travail.

1.5.3 Forces de contact pneu/sol

Dans le domaine des systèmes articulés, on est très souvent amené à modéliser des véhicules à roues équipées de pneumatiques. L'étude des interactions entre les roues du véhicules et le sol et, leurs conséquences sur le comportement du véhicule, est très souvent une des motivations principale pour la modélisation du système. Il s'avère donc utile de pouvoir générer symboliquement les grandeurs cinématiques nécessaires à l'évaluation d'un modèle de contact pneu/sol, afin de pouvoir produire des modèles symboliques complets pour la simulation de véhicules [96].

Nous présentons dans ce qui suit une méthode permettant de déterminer la cinématique du contact pneu/sol sous certaines hypothèses que voici :

- la roue est un corps relié à son parent par une articulation rotoïde dont l'axe est aligné sur le deuxième axe $\hat{\mathbf{Y}}_2$ du repère,
- la roue est considérée du point de vue géométrique, comme un disque sans épaisseur, dont le rayon r_w varie en fonction de l'écrasement de la roue réelle,
- le point de référence du corps est le centre de la roue,
- le contact entre la roue 'disque' et la sol est considéré comme ponctuel et ce point de contact Q est unique,
- du point de vue dynamique, les efforts normaux et de friction distribués dans la galette de contact entre la roue et le sol sont remplacées par une force et un couple résultant appliqués au point de contact Q,
- le sol est considéré comme infiniment rigide, nous négligeons donc sa déformation,
- le profil du sol est donné par la fonction[10] $z_{sol} = \mu(x, y)$ continue et dérivable, où x et y sont des coordonnées selon les deux premiers axes du repère de base, z_{sol} étant la hauteur du sol correspondante.

[10] de classe $\mathcal{C}^1$

Conventions et Définitions

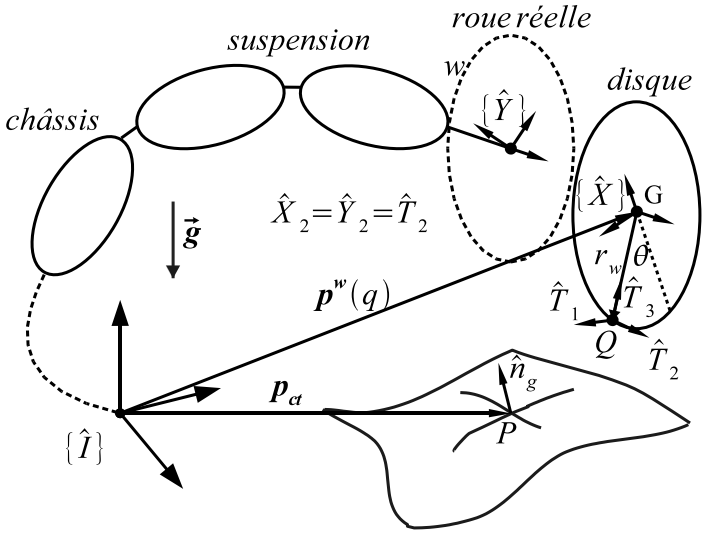


FIG. 1.18 – Cinématique de la roue

Afin de déterminer la position absolue du point de contact Q, nous allons nous aider de la figure 1.18. Sur cette figure, nous illustrons un ensemble d'éléments utiles relatifs à la roue réelle en rotation, et au disque géométrique.

- Le repère associé au corps réel est noté $\{\hat{\mathbf{Y}}\}$ et le repère $\{\hat{\mathbf{X}}\}$ est associé au disque et est obtenu en fixant arbitrairement à zéro l'angle de rotation de la roue par rapport à son corps parent. Les deux premières hypothèses impliquent que le repère $\{\hat{\mathbf{X}}\}$ a exactement la même orientation que celui du corps parent de la roue. Ces deux repères n'ont pas nécessairement la même origine car le vecteur position relative de l'articulation de la roue $\mathbf{d}_z^{hi}$, où i est l'indice de la roue, n'est pas forcément nul.
- L'angle θ et le rayon variable du disque r_w sont introduits pour localiser le point de contact Q dans le plan de la roue $\{\hat{\mathbf{X}}_1, \hat{\mathbf{X}}_3\}$.
- Le repère $\{\hat{\mathbf{T}}\}$, appelé 'repère tangent', est localisé au point de contact Q. Son vecteur $\hat{T}^1$ est tangent au disque, $\hat{T}_2$ est parallèle à l'axe de la roue $\hat{Y}_2$ et $\hat{T}_3$ est aligné sur le vecteur $\overrightarrow{QG}$. On peut donc écrire les relations

suivantes :

$$\mathbf{r_w} = \overrightarrow{GQ} = -r_w\ \hat{\mathbf{T}}_3 \quad , r_w > 0 \tag{1.57}$$

$$[\hat{\mathbf{T}}] = R_2(\vartheta)[\hat{\mathbf{X}}] \tag{1.58}$$

$$\text{avec } R_2(\vartheta) \triangleq \begin{pmatrix} c\vartheta & 0 & -s\vartheta \\ 0 & 1 & 0 \\ s\vartheta & 0 & c\vartheta \end{pmatrix} \text{ où } \begin{array}{l} c\vartheta \triangleq \cos\vartheta \\ s\vartheta \triangleq \sin\vartheta \end{array} \tag{1.59}$$

- pour plus de concisions on note r_{ct} les composantes du vecteur $\mathbf{r_w}$ dans le repère $\{\hat{\mathbf{T}}\}$: $r_{ct} = \begin{pmatrix} 0 \\ 0 \\ -r_w \end{pmatrix}$

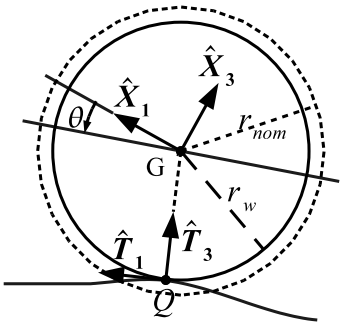


FIG. 1.19 – Variables r_w et θ

- la position absolue du point Q est donnée par le vecteur $\mathbf{p_{ct}} = \overrightarrow{OQ}$ avec :

$$\mathbf{p_{ct}}(q, r_w, \theta) = \mathbf{p}^w(q) + \mathbf{r}_w(q) = [\hat{\mathbf{I}}]^T \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \tag{1.60}$$

- la position du point de contact P sur le sol est donnée par le vecteur $\overrightarrow{OP}$:

$$\overrightarrow{OP} = [\hat{\mathbf{I}}]^T \begin{pmatrix} x \\ y \\ \mu(x,y) \end{pmatrix} \tag{1.61}$$

- Nous utilisons également un repère $\{\hat{\mathbf{S}}\}$, appelé *repère au sol*, localisé au point de contact P. Son axe $\hat{\mathbf{S}}_3 = \hat{n}_g$ est normal au sol, son axe $\hat{\mathbf{S}}_1$ est égal à l'axe $\hat{\mathbf{T}}_1$, et son axe $\hat{\mathbf{S}}_2$ est simplement tangent au sol et vérifie la relation $\hat{\mathbf{S}}_2 = \hat{\mathbf{S}}_3 \times \hat{\mathbf{S}}_1$.

- le vecteur $\hat{\mathbf{n}}_\mathbf{g}$ peut être calculé par l'expression suivante :

$$\hat{\mathbf{n}}_g = \frac{1}{\sqrt{1+\mu_x'^2(x,y)+\mu_y'^2(x,y)}}[\hat{\mathbf{I}}]^T \begin{pmatrix} -\mu_x'(x,y) \\ -\mu_y'(x,y) \\ 1 \end{pmatrix} \tag{1.62}$$

$$\text{avec } \mu_x' \triangleq \frac{\partial \mu}{\partial x} \text{ et } \mu_y' \triangleq \frac{\partial \mu}{\partial y} \tag{1.63}$$

Le problème consiste alors à rechercher les valeurs des quatre variables x, y, θ et r_w, introduites ci-dessus, telles que les conditions suivantes soient satisfaites :

- les points Q et P sont confondus :

$$g_1(x,y,\theta,r_w) \equiv \overrightarrow{OQ} = \overrightarrow{OP} \tag{1.64}$$

- la tangente à la roue au point de contact est perpendiculaire à la normale au sol :

$$g_2(x,y,\theta,r_w) \equiv \hat{\mathbf{T}}_1 \cdot \hat{\mathbf{S}}_3 = 0 \tag{1.65}$$

On trouvera un développement plus complet des équations (1.64) et (1.65) dans [13]. Celles-ci sont généralement non-linéaires et peuvent être résolues itérativement pas une méthode numérique de type Newton-Raphson.

Point de contact sur sol plat

Lorsque le sol est plan, nous proposons d'introduire la solution analytique dans les modèles de contacts générés. Les développements géométriques suivant sont proposés pour être utilisés dans ce contexte uniquement. Toutefois, cela couvre la plupart des applications relatives aux véhicules routiers.

Dans une première étape nous recherchons la valeur de l'angle θ.

Pour cela nous allons utiliser la matrice de rotation $R^{w,0}$ qui vérifie la relation suivante $[\hat{X}] = R^{w,0}[\hat{I}]$. Cette matrice donne l'orientation du repère fixé au disque géométrique par rapport au repère de référence.

Cette matrice peut également être obtenue par composition successive de trois matrices de rotation élémentaire.

1. La première rotation est effectuée autour de l'axe vertical $\hat{I}_3$, selon un angle ζ appelé angle de lacet de la roue. Cette matrice de rotation correspond à la matrice de passage du repère de référence au repère au sol : $[\hat{\mathbf{S}}] = R_3(\zeta)[\hat{\mathbf{I}}]$.

$$R_3(\zeta) = \begin{pmatrix} \cos(\zeta) & \sin(\zeta) & 0 \\ -\sin(\zeta) & \cos(\zeta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.66}$$

2. La seconde rotation est effectuée autour de l'axe tangent au sol $\hat{\mathbf{S}}_1$, selon un angle φ appelé angle de cambrure ou inclinaison de la roue. Cette matrice correspond à la matrice de passage du repère au sol au repère tangent :$[\hat{\mathbf{T}}] = R^1(\varphi)[\hat{\mathbf{S}}]$.

$$R_1(\varphi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & \sin(\varphi) \\ 0 & -\sin(\varphi) & \cos(\varphi) \end{pmatrix} \tag{1.67}$$

3. La dernière rotation est effectuée autour de l'axe de la roue disque $\hat{\mathbf{Y}}_2$, selon un angle $-\vartheta$ appelé angle d'enroulement de la roue. Cette matrice correspond à la matrice de passage du repère tangent au repère du disque :$[\hat{\mathbf{X}}] = R^2(-\vartheta)[\hat{\mathbf{T}}]$. Remarquons que cette matrice est exactement la transposée de la matrice $R_2(\vartheta)$ présentée plus haut. Nous continuerons donc à utiliser l'angle ϑ définit plus haut.

$$R_2(-\vartheta) = \begin{pmatrix} \cos(\vartheta) & 0 & \sin(\vartheta) \\ 0 & 1 & 0 \\ -\sin(\vartheta) & 0 & \cos(\vartheta) \end{pmatrix} \tag{1.68}$$

Le produit de ces trois matrices de rotations successives donne la relation suivante :

$$R^{w,0}(q) = R_2^T(\vartheta).R_1(\varphi).R_3(\zeta)$$

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} c\vartheta c\zeta + s\varphi s\vartheta s\zeta & c\vartheta s\zeta - s\varphi s\vartheta c\zeta & c\varphi s\vartheta \\ -c\varphi s\zeta & c\varphi c\zeta & s\varphi \\ -s\vartheta c\zeta + s\varphi c\vartheta s\zeta & -s\vartheta s\zeta - s\varphi c\vartheta c\zeta & c\varphi c\vartheta \end{pmatrix}$$

L'utilisation de quelques relations trigonométriques inverses nous permet d'extraire les trois angles de l'expression de la matrice $R^{w,0}(q)$:

$$\begin{aligned} \varphi &= \arcsin(r_{23}) = atan2(r_{23}, \sqrt{r_{13}^2 + r_{33}^2}) \\ \cos(\varphi) &= \sqrt{r_{13}^2 + r_{33}^2} = \sqrt{1 - \sin(\varphi)^2} \\ \sin(\varphi) &= r_{23} \\ \vartheta &= -atan2(-r_{13}, r_{33}) \\ \tan(\vartheta) &= r_{13}/r_{33} \\ \cos(\vartheta) &= 1/\sqrt{(1 + \tan(\vartheta)^2)} \\ \sin(\vartheta) &= \tan(\vartheta)\cos(\vartheta) \\ \zeta &= atan2(-r_{21}, r_{22}) \\ \cos(\zeta) &= r_{22}/\sqrt{r_{21}^2 + r_{22}^2} \\ \sin(\zeta) &= -r_{21}/\sqrt{r_{21}^2 + r_{22}^2} \end{aligned}$$

La seconde étape consiste alors à calculer la valeur du rayon r_w nécessaire pour calculer la position du point Q.

Si le sol est plan, la hauteur $z_{sol}(x, y)$ est constante, et le rayon de contact est situé dans le plan $\{\hat{\mathbf{S}}_2, \hat{\mathbf{S}}_3\}$ ce qui rend le problème relativement trivial :

$$r_w = (\mathbf{p}^w \cdot \hat{\mathbf{S}}_3 - z_{sol}(x, y))/\cos(\varphi) \tag{1.69}$$

On notera que :

- une valeur du rayon r_w supérieure à la valeur nominale r_{nom} du rayon de la roue signifie qu'il n'y a pas de contact avec le sol.
- il est possible de travailler avec un sol de hauteur variable en prenant les deux premières coordonnées du centre de la roue pour évaluer la hauteur du sol. Ce calcul n'est plus strictement correct mais peut constituer une approximation satisfaisante dans une multitude de cas si l'angle d'inclinaison de la roue est faible et si la surface du sol est quasiment plane. Un exemple d'application est la simulation de la réponse d'un véhicule placé sur un dispositif d'excitation vertical. Ce type d'analyse est effectuée, par exemple, pour étudier le confort du véhicule et permet de développer des contrôleurs efficaces pour la gestion de suspensions actives.

Les coordonnées absolues p_{ct} des points Q et P sont alors calculées par les relations suivantes :

$$p_{ct} = p^w + R^{0,w} \begin{pmatrix} 0 \\ 0 \\ -r_w \end{pmatrix} \tag{1.70}$$

Cinématique du point de contact

Le vecteur position du point de contact géométrique étant connu, il est maintenant possible de calculer la vitesse du point de contact. Cependant nous allons calculer deux vitesses différentes pour ce point de contact :

1. la vitesse $\mathbf{v}_{ct}$ du point de contact matériel de la roue réelle en rotation. Dans ce cas on tient compte de la rotation de la roue par rapport à son corps parent $\mathbf{\Omega}^w = \dot{q}^w \, \hat{\mathbf{X}}_2$,
2. la vitesse $\mathbf{v}_{ct}^{geo}$ du point de contact géométrique appartenant à la roue disque dont la vitesse de rotation relative est considérée nulle.

$$\mathbf{v}_{ct}^{geo} = \mathbf{v}_{ct} - \dot{q}^w \, r_w \, \hat{\mathbf{T}}_1 \tag{1.71}$$

On notera que la variation temporelle $\dot{r}_w$ de la valeur du rayon a été négligée ici.

A l'aide de ces deux vitesses, nous allons définir quelques indicateurs des conditions de contacts utilisés par les modèles de contacts roue/sol.

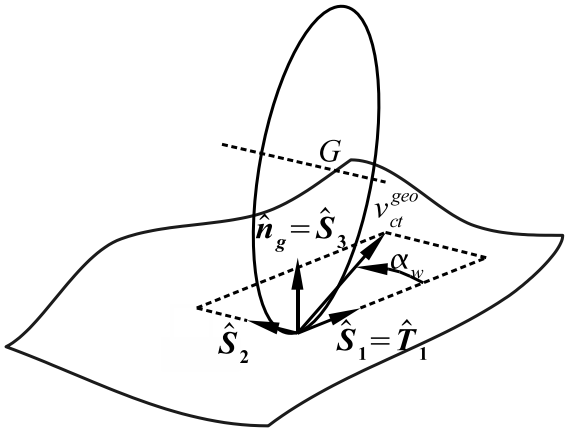


FIG. 1.20 – Angle de glissement latéral

- *l'angle de glissement latéral* α_w, est l'angle mesuré au sol dans le plan $\{\hat{\mathbf{S}}_1, \hat{\mathbf{S}}_2\}$ entre l'axe $\hat{\mathbf{S}}_1$ et la projection du vecteur $\mathbf{v}_{ct}^{geo}$ dans ce plan, comme illustré sur la figure 1.20. Un angle positif indique que la roue glisse vers sa gauche en avançant sur le sol. (C'est généralement le cas d'une roue qui tourne vers la droite.)

$$\alpha_w = atan2(\mathbf{v}_{ct}^{geo} \cdot \hat{\mathbf{S}}_2, \mathbf{v}_{ct}^{geo} \cdot \hat{\mathbf{S}}_1) \tag{1.72}$$

- la vitesse de glissement longitudinale κ_w est la projection selon l'axe $\hat{\mathbf{S}}_1$, de la vitesse $\mathbf{v}_{ct}$ du point matériel de la roue réelle en contact avec le sol. Ce point coïncide donc avec Q.

$$\kappa_w = \mathbf{v}_{ct} \cdot \hat{\mathbf{S}}_1 \tag{1.73}$$

- *le glissement longitudinal* γ_w est définit par le rapport de la vitesse de glissement à la vitesse longitudinale du centre de la roue.

$$\gamma_w = \kappa_w / (\dot{\mathbf{p}}^w \cdot \hat{\mathbf{S}}_1) \tag{1.74}$$

- *la pénétration* de la roue dans le sol qui vaut $p_s = min\{0, \cos(\varphi_w)(r_w - r_{nom})\}$. Cette valeur est mesurée dans la direction de la normale au sol et est négative lorsque la roue est en contact.
- *la vitesse de pénétration* de la roue dans le sol est la variation de la pénétration. Elle est approximée par la projection de la vitesse du point matériel de contact sur la normale au sol $\dot{\mathbf{p}}_s = \mathbf{v}_{ct} \cdot \hat{\mathbf{S}}_3$ lorsque la roue est en contact. Elle est nulle sinon.

Forces et couples de contact

Les modèles de forces de contact sont variés : ils diffèrent par leur niveau de raffinement et le nombre de composantes de forces et couples calculés. Il n'est naturellement pas toujours nécessaire d'utiliser le modèle le plus complexe pour une analyse numérique particulière. Par exemple, un modèle qui ne calcule que la composante de force normale au sol sur base de la pénétration de la roue, peut suffire à une recherche d'équilibre statique, alors que pour une analyse numérique du comportement en courbe, on utilisera un modèle plus complet dont les équations constitutives prennent en compte plusieurs conditions de contact pour le calcul des efforts appliqués à la roue.

Les composantes des forces et des couples sont exprimées dans le repère au sol $\{\hat{\mathbf{S}}\}$, pour la plupart des modèles et on considère que la résultante de force est appliquée au point Q. Les trois composantes F_{long}, F_{lat} et F_{vert} de la force de contact $\mathbf{F}_{ct}$, correspondent respectivement aux efforts longitudinal, latéral et vertical.

Les trois composantes M_{roll}, M_{pitch} et M_{yaw} du couple de contact $\mathbf{M}_{ct}$, correspondent respectivement aux moments de basculement, de résistance au roulement et d'alignement.

Notre but n'est pas de développer les expressions de ces composantes de forces en fonction des conditions de contact. Le lecteur intéressé par cet aspect trouvera plus d'informations dans les ouvrages suivant plus particulièrement dédicacés à ce sujet : [97, 98, 99].

En pratique, la composante verticale F_{vert} est souvent calculée par une approximation assez simple et peut en fait être considérée comme une condition de contact plutôt que comme un résultat, et dans ce sens servir d'indicateur des conditions de contact.

$$F_{vert} = K_{rad}\, p_s + D_{rad}\, \dot{p}_s \tag{1.75}$$

où K_{rad} et D_{rad} représentent respectivement les coefficients de raideur et d'amortissement radial équivalent du pneu.

On notera que cette approximation n'est valable que pour un pneu parfaitement circulaire roulant sur un sol plat. Elle ne permet pas d'étudier des phénomènes dynamiques plus complexes pouvant survenir lorsque le pneu présente une légère excentricité ou lorsque l'on roule sur un sol ondulé à une vitesse particulière qui pourrait conduire à l'excitation des modes propres de la carcasse du pneu [99]. Mais elle reste une très bonne approximation pour la plupart des simulations de véhicules routiers.

Deux modèles couramment utilisés sont présentés dans [13]. Il s'agit des modèles *Calspan* et *Bakker/Pacejka*. Ils sont également repris en annexe de ce document.

En regroupant les composantes de forces et de couples dans une même entité on peut écrire une équation correspondant à l'évaluation du modèle de contact

implémenté dans une fonction externe :

$$W_r = fctext(p_s, \alpha_w, \phi_w, \gamma_w, \dot{p}_s, t) \tag{1.76}$$

$$avec\ W_r = \{F_{long}, F_{lat}, F_{vert}, M_{roll}, M_{pitch}, M_{yaw}\} \tag{1.77}$$

Projection dans le repère de la roue réelle

La projection des efforts de contact sur le corps correspondant à la roue nécessite de calculer :

1. les composantes des forces et couples extérieurs dans le repère $\{\hat{\mathbf{Y}}\}$ du corps réel en rotation avec la roue,
2. le couple de transport nécessaire pour appliquer ces composantes au centre de gravité du corps w selon la convention établie par ROBOTRAN.

Pour passer du repère au sol $\{\hat{\mathbf{S}}\}$, dans lequel sont exprimées les composantes calculées par le modèle de contact, au repère en rotation $\{\hat{\mathbf{Y}}\}$, il faut tenir compte de la rotation de ce dernier. Cette rotation s'effectue autour de l'axe $\hat{Y}_2$ et on peut donc définir une matrice de rotation $R^2(q_w)$:

$$R^2(q_w) = \begin{pmatrix} \cos(q_w) & 0 & -\sin(q_w) \\ 0 & 1 & 0 \\ \sin(q_w) & 0 & \cos(q_w) \end{pmatrix} \text{ telle que } [\hat{\mathbf{Y}}] = R^2(q_w)[\hat{\mathbf{X}}] \tag{1.78}$$

On peut alors écrire : $[\hat{\mathbf{Y}}] = R^2(q_w)R^2(-\vartheta)R^1(\varphi)[\hat{\mathbf{S}}] \triangleq R^{YS}[\hat{\mathbf{S}}]$.

Le couple de transport des forces du point de contact Q au centre de la roue P est donné par la relation suivante :

$$F^w_{ext} = R^2(q_w)R^2(-\vartheta)R^1(\varphi)F_{ct} \tag{1.79}$$

$$L^w_{ext} = R^2(q_w)R^2(-\vartheta)(R^1(\varphi)M_{ct} + (\tilde{r}_{ct}R^1(\varphi)F_{ct})) \tag{1.80}$$

1.6 Génération symbolique et implémentation

Nous présentons dans cette section diverses extensions et améliorations apportées au logiciel ROBOTRAN, afin de permettre la gestion des concepts présentés dans ce chapitre, et d'augmenter les capacités de traitement symbolique de ROBOTRAN mais également la performance des modèles générés.

Les **extensions** concernent essentiellement l'ajout de nouveaux types d'expressions symboliques pour le traitement des interactions avec l'environnement, mais également l'ajout de certains attributs à la définition d'une expression. Elles renforcent encore le logiciel ROBOTRAN dans sa capacité à traiter la génération symbolique des équations du mouvement de systèmes mécaniques articulés, plus efficacement que les logiciels de manipulation symbolique à usage général.

Les **améliorations** consistent en l'introduction de **nouveaux traitements symboliques** visant à optimiser les modèles, d'une part en réduisant le nombre total d'opérations arithmétiques et d'autre part en réduisant le nombre d'équations récursives. Sur ce plan, l'objectif est de doter ROBOTRAN de fonctionnalités équivalentes ou supérieures à celles qui sont disponibles de façon standard dans les logiciels de manipulation symbolique actuels, essentiellement en ce qui concerne leur aptitude à générer du code optimisé lors de l'exportation des équations symboliques dans divers langages de programmation.

1.6.1 Extensions

Afin de pouvoir générer symboliquement les équations présentées dans la section 1.5 nous avons ajouté deux nouvelles *natures* d'expressions dans ROBOTRAN. La première est une expression qui peut servir à la fois à l'appel de fonction externe et à la manipulation de grandeur vectorielle. La seconde correspond à des fonctions à un seul argument.

De nouveaux attributs ont également été ajoutés aux définitions des structures des expressions et des équations.

Nouveaux attributs des expressions

La génération symbolique d'équations nécessite la manipulation d'expressions symboliques. Ces expressions sont représentées en mémoire par des arbres, des listes, ou des structures de données. Une expressions peut être un noeud ou un arbre entier selon sa nature et sa complexité. Les structures qui définissent les arbres et les noeuds doivent donc posséder tous les attributs nécessaires pour permettre des manipulations faciles et efficaces. Des attributs utiles ou indispensables pour certains traitements n'étaient pas présents dans la version initiale [12] de ROBOTRAN. Certains nouveaux attributs sont requis par les extensions qui sont présentées dans les paragraphes suivants.

Référence à l'équation qui définit une variable intermédiaire. Le membre de gauche d'une équation de ce type est un symbole, le nom d'une variable, qui est définie par le membre de droite de l'équation. Ce membre de droite est généralement une expression plus complexe, qui peut être représentée par un arbre dont les noeuds sont des opérateurs ou des fonctions et dont les feuilles sont des expressions simples, des symboles correspondant à des données ou à des variables définies par d'autres équations.

Il va sans dire que dans ROBOTRAN, la structure d'une équation contient des pointeurs vers les expressions de son membre de gauche et de droite. Il était logique et très utile d'ajouter à la structure des expressions, un champ qui peut contenir une référence à l'équation qui la définit, comme illustré sur la figure 1.21. Naturellement toutes les expressions ne sont pas définies par une

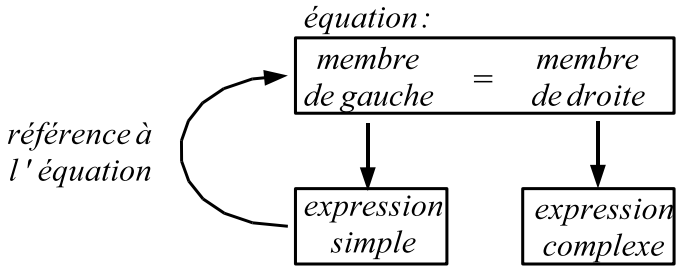


FIG. 1.21 – Référence à l'équation qui définit une variable

équation et cet attribut est vide pour la plupart des expressions, mais lorsque l'expression correspond au membre de gauche d'une équation, la disponibilité de cette référence facilite les manipulations.

La nature secondaire d'une expression Cet attribut secondaire permet de distinguer, au sein d'une même famille, des expressions qui ont toutes la même nature principale. Il s'avère également utile pour étendre le champ d'application de certaines expressions existantes.

Par exemple, les fonctions `SQRT` et `ATAN`, présentées plus loin, sont toutes deux des fonctions unaires, ce qui est leur nature principale. Elles se distinguent par leur nature secondaire qui indique de quelle opération il s'agit, *racine carrée* ou *arc tangente*.

La liste des arguments d'une fonction Les expressions disponibles initialement dans ROBOTRAN étaient uniquement des symboles, des opérations arithmétiques binaires ou les sinus ou cosinus des variables articulaires rotoïdes que l'on peut considérer comme des fonctions unaires. Néanmoins, l'introduction des fonctions multi-variables, nécessite de disposer d'une liste des arguments de cette fonction.

Nous avons donc introduit en parallèle à l'attribut *argument*, déjà présent et utilisé par les sinus et cosinus articulaires, un attribut *liste d'arguments* qui permet de stocker l'ensemble des arguments d'une fonction. Cette liste est notée `rhs.args` sur la figure 1.22 qui illustre la représentation d'une expression correspondant à un appel à une fonction externe.

On notera que le premier élément de la liste est une expression simple correspondant au nom de la fonction.

L'indice de l'élément d'un vecteur Il s'est avéré indispensable d'être capable de manipuler non seulement des variables scalaires, mais également des

variables vectorielles, correspondant à des n^{uple} d'éléments scalaires qui sont représentés par des tableaux dans la plupart des langages de programmation. En effet certaines fonctions externes appelées peuvent renvoyer plusieurs valeurs, dans un tableau. Il est utile que l'expression de chaque élément de ce tableau contienne l'indice de l'élément.

L'attribut *indice* est déclaré pour cet usage. Sur la figure 1.23 qui illustre la structure d'un vecteur et de ses éléments, cet attribut des éléments est noté `rhs.idx[1]` .

Fonctions multi-variables

L'interaction du système avec son environnement nécessite la plupart du temps de faire appel à des routines externes qui implémentent les modèles constitutifs de ces interactions.

La présence dans le système d'éléments actifs ou passifs complexes ayant une dynamique propre, tels que des moteurs, des amortisseurs, des vérins, etc. requiert également l'évaluation de fonctions externes qui implémentent les modèles dynamiques de ces éléments.

Il est donc utile et bien souvent nécessaire de faire appel à des fonctions externes, dans un modèle symbolique de système multicorps. Ces appels à des fonctions externes servent donc d'*interface* entre le modèle mécanique et les modèles des autres éléments du système, ce qui offre une grande facilité d'utilisation des modèles générés par ROBOTRAN.

Pour en être capable *sans casser la structure récursive des équations*, il faut créer une expression symbolique qui est fonction de plusieurs autres expressions.

Il s'agit d'un nouveau type d'expression dont la nature principale est définie par la valeur `XFCT`. Il est décliné en deux variantes distinctes par leur nature secondaire, et dont l'usage est intimement lié.

Une fonction externe à plusieurs arguments La première déclinaison est utilisée pour définir une fonction à plusieurs arguments et ainsi pouvoir insérer un appel de routine externe dans une équations symbolique. L'attribut de nature secondaire prend alors la valeur `CALL`. Comme illustré sur la figure 1.22, le premier argument de la liste `rhs.args` pointe vers une expression simple de type `CST` qui contient le nom de la fonction. Les arguments suivants sont les expressions à passer en paramètres à la fonction.

On remarquera que

- l'expression '*fctext*' pointée par `rhs.args[0]`, le premier argument de la liste, n'est utilisée que comme un conteneur du nom de la fonction mais n'apparaît pas directement dans l'arborescence des expressions. C'est une expression '*flottante*'.
- cette expression '*fctext*' possède une référence à l'expression qui définit la fonction multi-variables. Cette référence est stockée dans le champ

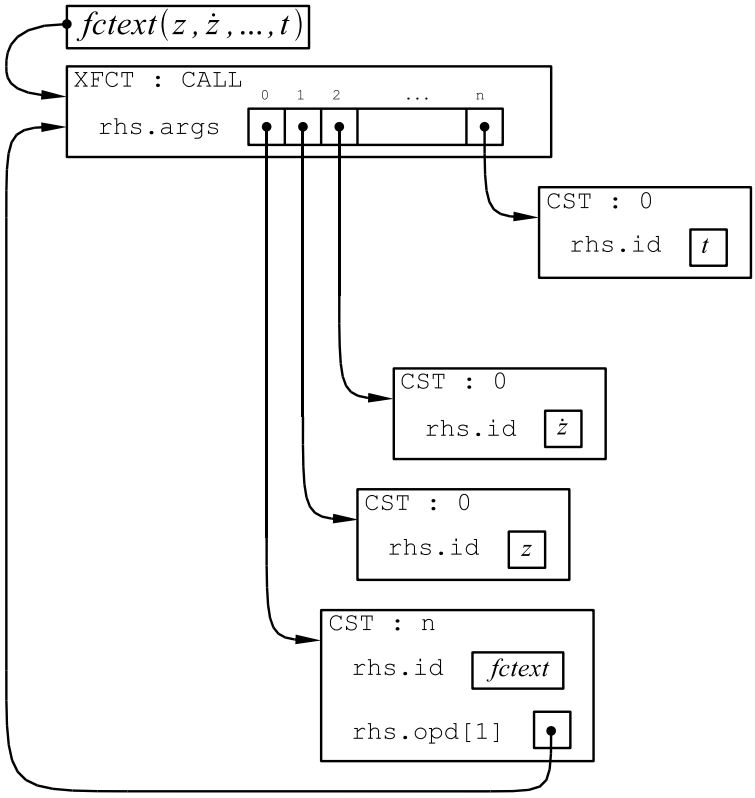


FIG. 1.22 – Représentation d'une fonction à n arguments

`rhs.opd[1]`.

- le nombre `n` de paramètres dont dépend la fonction est stocké dans l'attribut nature secondaire de l'expression simple '*fctext*'. Cette valeur non nulle indique par ailleurs qu'il ne s'agit pas d'une expression classique, mais du nom d'une fonction multi-variables.

Une expression qui correspond à une fonction multi-variables ne fera jamais l'objet de simplification symboliques en raison de sa nature a priori quelconque.

Un symbole correspondant à un vecteur La seconde déclinaison est utilisée pour manipuler des symboles de vecteurs. L'attribut de nature secondaire prend alors la valeur `VEC`.

A l'origine, tous les symboles présent dans les équations générées par ROBOTRAN étaient considérés comme des scalaires. Néanmoins l'interaction avec

des fonctions externes requiert de pouvoir utiliser des vecteurs. En effet, une fonction peut prendre un vecteur comme argument ou encore renvoyer plusieurs valeurs dans un vecteur. Tous les langages de programmation visés pour l'impression des modèles générés supportent les variables de type 'vecteur', sous forme de tableau en FORTAN, MATLAB, C(++), JAVA, etc. ou sous forme de pointeurs en C ou encore de référence en JAVA ou C++.

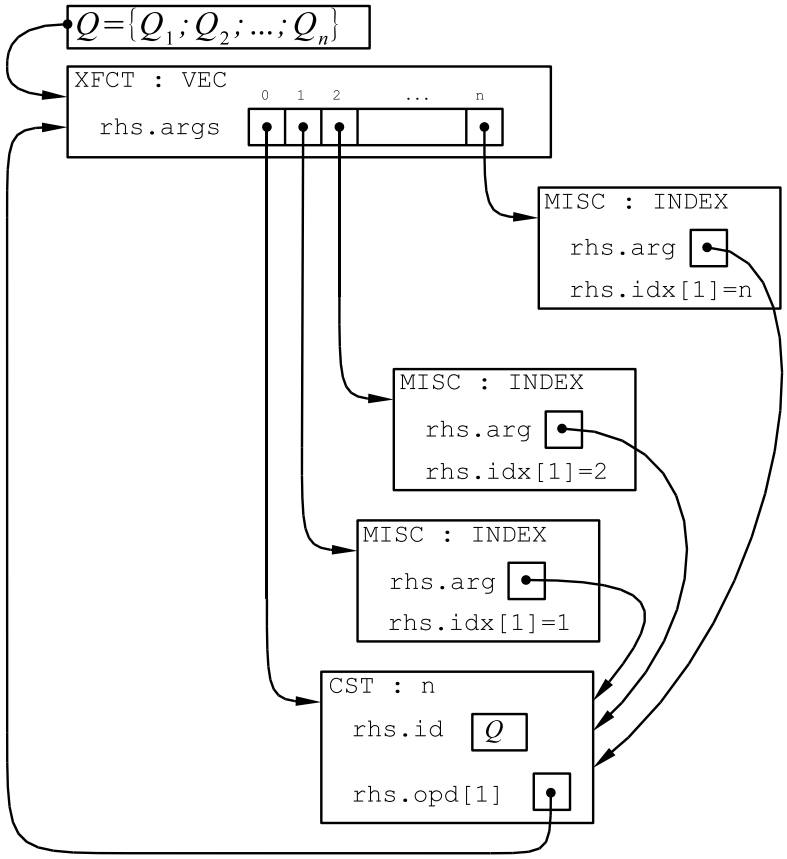


FIG. 1.23 – Représentation d'un vecteur de n éléments

La figure 1.23 illustre une combinaison d'expressions pour la construction d'un vecteur. Il s'agit de la configuration utilisée pour récupérer des valeurs multiples renvoyées par une fonction. Avec cette configuration, on peut générer l'équation $Q = fctext(Q, q, \dot{q}, \ldots, t)$; dans laquelle la fonction $fctext$ renvoie n résultats dans le vecteur Q. On remarquera que

- comme c'était déjà le cas pour les fonctions multi-variables, le premier argument de la liste `rhs.args` pointe également vers une expression simple de type `CST`, qui contient cette fois le nom 'Q'. Mais cette fois, c'est cette expression 'Q' qui est utilisé dans le membre de gauche des équations et c'est l'expression qui définit le vecteur qui est '*flottante*'.
- cette expression 'Q' contient à nouveau une référence à l'expression qui définit le vecteur. Cette référence est notée `rhs.opd[1]` sur la figure 1.23.
- les éléments du vecteur sont ici des expressions d'un nouveau type `MISC` qui est présenté ci-après. La valeur `INDEX` de leur attribut nature secondaire indique qu'il s'agit d'éléments d'un vecteur.

Fonctions unaires

Pour le cas de cette extension ci, la motivation provient à nouveau de la volonté de générer symboliquement les équations permettant de calculer les composantes des forces extérieures agissant sur les corps. Dans les sections 1.5.2 et 1.5.3, on peut constater que le calcul de la cinématique des points d'applications nécessite l'utilisation de nombreuses fonctions trigonométriques directes et/ou inverses, ainsi que de racines carrées. Toutes ces fonctions, hormis *atan*2, sont des fonctions à un seul argument.

Les expressions de types `sin` et `cos` qui existaient dans ROBOTRAN, n'étaient en réalité pas des fonctions trigonométriques, mais des variables auxiliaires particulières correspondant aux *sinus* et *cosinus* des coordonnées articulaires associées aux articulations rotoïdes.

Nous avons donc introduit une collection d'expressions dont la caractéristique commune est de dépendre d'une autre expression. Ce nouveau type appelé `MISC`[11] est également décliné en plusieurs variantes.

Fonctions à un argument Une série de ces variantes correspondent à des fonctions à un seul argument, telles que les *sinus*, *cosinus*, *tangente*, *racine carrée*, etc.

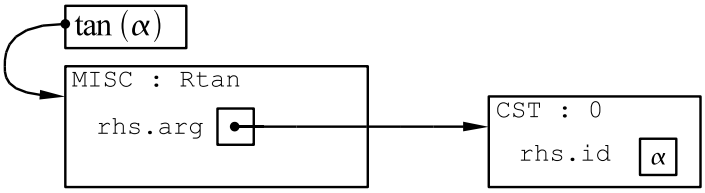


FIG. 1.24 – Représentation d'une fonction à un argument

[11] pour Miscellaneous

La figure 1.24 illustre une expression de type `MISC` dont la nature secondaire `Rtan` précise le type de fonction dont il s'agit. La fonction trigonométrique *tangente* est prise comme exemple ici. Le champ `rhs.arg` pointe vers l'expression qui est utilisée comme argument de la fonction. Cela peut être une variable simple ou une expression plus complexe.

Les fonctions ajoutées sont les suivantes :

1. `SQRT` : *racine carrée*,
2. `ACOS` : *arc cosinus*,
3. `ASIN` : *arc sinus*,
4. `ATAN` : *arc tangente*,
5. `RSIN` : *sinus*,
6. `RCOS` : *cosinus*,
7. `RTAN` : *tangente*,

On notera que les types des fonctions *sinus* et *cosinus* ont un nom différent des types `sin` et `cos` existants, car ces dernières sont dédicacées aux coordonnées articulaires et font l'objet de traitements symboliques particuliers.

Des règles de simplification symbolique ont été ajoutées et sont appliquées lorsque certaines combinaisons de ces nouvelles fonctions sont rencontrées dans les expressions, comme $1/\sqrt{1+\tan^2\alpha} \mapsto cos\alpha$ par exemple.

Elément d'un vecteur Des expressions de type `MISC` peuvent également être utilisées pour accéder à un élément d'un vecteur. Leur nature secondaire prend alors la valeur `INDEX`.

Cette variante a déjà été illustrée sur la figure 1.23 ci-dessus. L'attribut *argument* `rhs.arg` de l'élément fait alors référence à l'expression qui défini le nom du vecteur, laquelle contient elle même une référence à l'équation dans laquelle le vecteur a été initialisé. Ces expressions sont particulièrement utiles pour récupérer les composantes de forces et couples calculées par un modèle constitutif externe, comme on l'a vu dans la section 1.5.3, dont l'équation (1.76) est un exemple.

1.6.2 Nouveaux traitements symboliques

Nous présentons ici quelques nouvelles manipulations appliquées aux expressions symboliques et à la liste des équations après leur génération.

Organisation du traitement symbolique

On peut distinguer plusieurs types de manipulations ou de traitements symboliques selon l'objet du traitement et la phase du processus où ils sont effectués.

Ainsi on considère différemment les traitements relatifs à la topologie du modèle, aux expressions symboliques ou encore aux équations générées. Certains traitements visent à réduire le nombre d'opérations arithmétiques alors que d'autres visent à réduire le nombre de variables présentes dans le modèle. Selon leur nature et leur objectif, ces traitements pourront être appliqués **avant**, **pendant** ou **après** la génération des équations du système.

Outre les traitements effectués automatiquement par ROBOTRAN, on pourrait aussi prendre en considération les manipulations effectuées par l'ingénieur au moment de la description du système en termes de corps, d'articulations et de composants externes ou encore lors du développement et de l'écriture des formalismes récursifs utilisés.

Ainsi l'utilisation des formalismes récursifs pour la génération des équations cinématiques et dynamiques, tels que nous le proposons, permet d'avoir un certain contrôle a priori sur le nombre d'équations et d'opérations qui seront générées pour un modèle et un système donné. Ce n'est pourtant pas une caractéristique commune à tous les logiciels de génération de modèle de systèmes articulés.

Les conventions de modélisation des systèmes ont également une importance relativement importante quant aux possibilités de réduction du nombre d'équations qui seront présentes dans le modèle. Un exemple classique est celui du choix du type de coordonnées utilisées : on connaît bien la différence entre un choix de coordonnées absolues ou relatives au niveau du formalisme.

Pour des systèmes complexes, les choix effectués au niveau de la description topologique du système comme par exemple le choix des coupures ou l'ajout de chaînes cinématiques fictives, peuvent avoir des conséquences importantes sur la compacité et l'efficacité du modèle généré. Ces considérations seront développées dans le chapitre relatif aux systèmes à topologie complexe.

Traitements effectués avant la génération

Dans la section 1.5.2, nous avons mentionné le cas où on pouvait faire une économie d'équations en calculant les positions et vitesses des points d'application des forces de liaison ainsi que les matrices de rotation des repères des corps reliés, par rapport au repère du parent commun le plus proche, plutôt que par rapport au repère inertiel.

Cette procédure doit évidemment être appliquée avant la génération des équations cinématiques relatives à ces éléments.

La première étape consiste à marquer les corps pour lesquels on va devoir calculer des grandeurs cinématiques absolues, tels que la position, la vitesse, la matrice de rotation, etc. C'est le cas pour tous les corps en interaction avec l'environnement ainsi que leurs ancêtres. Le marquage se fait en remontant l'arborescence depuis les corps en question et en marquant chaque parent jusqu'à ce qu'on rencontre un parent déjà marqué.

Ensuite, pour les corps soumis à des forces de liaison point à point, on recherche l'ancêtre commun le plus proche. Pour cela, on peut construire, la liste des corps de la chaîne cinématique partant de la base et allant vers chacun des corps. On parcourt alors la liste depuis la base en comparant les corps de chaque liste jusqu'au dernier ancêtre commun.

Si cet ancêtre commun est marqué, on utilisera les grandeurs cinématiques absolues pour le traitement des forces de liaison. Sinon, on initiera les parcours récursifs progressifs en partant de cet ancêtre commun, en le considérant comme corps de base pour la génération de la cinématique relative des points d'applications des deux corps.

Traitements effectués pendant la génération

ROBOTRAN est un logiciel initialement développé pour la génération d'expressions symboliques non-récursives. Dans ce contexte, un soin tout particulier a été apporté aux simplifications trigonométriques. De puissantes routines de détection et de simplification des combinaisons trigonométriques classiques ont été développées dans [12]. Néanmoins ces routines présentent deux limitations importantes dans le contexte de ce travail :

1. elles se focalisent sur les expressions trigonométriques et ne simplifient pas certaines combinaisons arithmétiques élémentaires,
2. elles s'avèrent relativement inefficaces voire inutiles lorsqu'elles sont appliquées à des équations générées par des formalismes récursifs.

Les deux traitements décrits ci-dessous, permettent de compenser ces limitations et de réaliser encore plus de simplifications.

Mise en évidence Cette simplification élémentaire permet de regrouper des éléments communs à plusieurs termes d'une addition ou d'une soustraction. La procédure consiste à identifier des éléments communs entre les termes, à les extraire des termes et à les placer en évidence. L'expression suivante illustre le principe de base.

$$a \cdot p \cdot x + b \cdot p \cdot z \mapsto p \cdot (a \cdot x + b \cdot z) \tag{1.81}$$

La situation n'est pas toujours aussi simple que dans l'exemple ci-dessus. En effet, pour des expression plus complexe comme celle-ci $a*b*c+a*d*f+b*c*e$, la simplification optimale nécessite de traiter les trois termes simultanément. Cela nécessite de développer les expressions en redistribuant tous les termes éventuellement déjà mis en évidence. Nous ne faisons pas cela. Nous travaillons seulement au niveau des opérations d'addition et de soustraction, qui dans ROBOTRAN prennent seulement deux arguments. En traitant uniquement deux termes à la fois, nous obtenons une méthode simple et rapide, qui offre de bons résultats pour un usage intensif en cours de génération. Naturellement ce choix comporte des limitations. L'expression ci dessus, peut se présenter

de trois façons différentes selon le groupement des termes et conduire à des optimisations différentes :

$$\begin{aligned}
1: &\ (a\cdot b\cdot c + b\cdot c\cdot e) + a\cdot d\cdot f \mapsto b\cdot c\cdot (a+e) + a\cdot d\cdot f \\
2: &\ (a\cdot b\cdot c + a\cdot d\cdot f) + b\cdot c\cdot e \mapsto a\cdot (b\cdot c + d\cdot f) + b\cdot c\cdot e \\
3: &\ a\cdot b\cdot c + (a\cdot d\cdot f + b\cdot c\cdot e) \mapsto a\cdot b\cdot c + (a\cdot d\cdot f + b\cdot c\cdot e)
\end{aligned}$$

On remarquera que :

1. dans le premier cas on obtient une simplification maximale de deux multiplications,
2. dans le second cas, seulement un terme est mis en évidence et une seule multiplication est économisée,
3. dans la troisième configuration, aucune mise en évidence ne peut-être effectuée.

Lorsqu'une expression contient beaucoup de termes et que les quatre opérations arithmétiques de base sont présentes, le problème de la mise en évidence optimale est assez lourd et complexe. En développant complètement les expressions en somme de produit et en les re-factorisant de façon optimale, on pourrait probablement effectuer quelques simplifications supplémentaires. Ce processus devrait être appliqué au moment de la création d'une équation sur base de l'expression plutôt que dans les routines d'addition et de soustraction symbolique, pour maintenir une performance optimale lors de la phase de génération des équations.

La mise en évidence optimale n'a toutefois pas été implémentée. En effet, nous avons constaté en faisant quelques tests préliminaires que le gain obtenu sur les équations restait vraiment très faible. On constate même l'apparition de nouvelles variables trigonométriques auxiliaires dans le cas d'articulations rotoïdes successives à axes parallèles, ce qui a pour effet paradoxal d'augmenter le nombre d'opérations au lieu de le réduire.

Nous avons donc préféré ne pas alourdir le processus de génération en y incluant un processus de factorisation optimale dont le rendement pourrait s'avérer très faible dans le contexte de la génération symbolique basée sur des formalismes récursifs.

Nous n'utilisons donc qu'une version simple et rapide de la méthode de mise en évidence dans ROBOTRAN.

Dérécursification partielle La génération récursive des équations est une méthode très efficace pour limiter le nombre d'opérations en réutilisant des variables intermédiaires que l'on substitue à des expressions plus complexes et ainsi réduire la taille des expressions à manipuler. Cette technique de génération récursive constitue d'ailleurs un avantage essentiel du logiciel ROBOTRAN par rapport à DYNAFLEX [6] ou SYMOFROS [74, 75] qui sont d'autres outils

de génération d'équations basés sur le logiciel symbolique commercial MAPLE. Néanmoins, cette substitution des variables intermédiaires a pour effet de limiter le champ d'action des routines de simplification symbolique. En effet, les variables intermédiaires sont alors considérées comme des symboles élémentaires et *masquent* ainsi les expressions plus complexes qui les définissent.

Ce problème peut être aisément contourné en substituant aux variables intermédiaires, le membre de droite des équations qui les définissent. Ce processus baptisé "dé-récursification" peut être appliqué de façon récursive jusqu'à ce qu'il n'y ait plus de variables intermédiaires dans l'expression. Cela se produit automatiquement dans quasiment tous les logiciels symboliques génériques et conduit à des performances médiocres en ce qui concerne la capacité à traiter des grands systèmes. Il est donc indispensable de contrôler ce processus. Pour cela, il suffit de fixer le niveau de '*profondeur*' de l'expansion.

On dira qu'on effectue une expansion de niveau 1 si on substitue toutes les variables intermédiaires initiales d'une expression par leur définition.

$$\begin{array}{ll} x = a + b & x = a + b \\ y = c + x \quad \longmapsto & y = c + a + b \\ z = y \cdot d & z = (c + x) \cdot d \end{array}$$

On effectuera une expansion de niveau 2 si on remplace à nouveau les variables intermédiaires de l'expression obtenue par une expansion de niveau 1, etc.

Étant donné son principe, on pourrait s'attendre à ce que ce processus d'expansion n'ait comme conséquence que d'augmenter la taille et la complexité, c'est-à-dire le nombre d'opérations, d'une expression. Ce n'est pas toujours le cas ! Il arrive régulièrement qu'une expression obtenue après une expansion de niveau 1 ou 2 seulement, puissent être réduite en une expression plus simple que l'expression initiale ! Dans ce cas, le gain est net et sans appel : l'expression initiale est alors remplacée par la nouvelle.

$$\begin{array}{ll} x = \cos a \cdot u & x = \cos a \cdot u \\ y = \sin a \cdot u \quad \longmapsto & y = \sin a \cdot u \\ z = \cos a \cdot x + \sin a \cdot y & z = \cos a \cdot \cos a \cdot u + \sin a \cdot \sin a \cdot u \\ & \quad = u \end{array}$$

Parfois, l'expression obtenue après expansion contient un nombre d'opérations inférieur à la somme des opérations de l'expression initiale et des expressions qui définissent les variables remplacées. Dans ce cas, le remplacement serait intéressant seulement si les variables remplacées ne sont pas utilisées par ailleurs,ce qui est difficile à établir en cours de génération. Ce cas de figure ne peut donc pas être exploité avant la fin de la phase de génération des équations.

Souvent, cette expansion n'apporte aucune simplification intéressante, surtout si on choisit un niveau d'expansion trop élevé, comme on peut s'y attendre. L'expression initiale est alors conservée.

Traitements effectués après la génération

Lorsque toutes les équations ont été générées, on peut alors déterminer une série d'informations utiles relatives à celles-ci, et procéder à une phase d'optimisation globale de la liste. Certains traitements présentés ici ont été inspirés ou requis par l'étude de la vectorisation [100] ou de la parallélisation mais se sont révélés très intéressants également hors de ce contexte. Toute réduction du nombre d'opérations est bénéfique dans le cas de l'exécution séquentielle.

L'ensemble de ces traitements permet de reconstruire un schéma d'équations récursives à partir d'un ensemble d'équations éventuellement non récursives. Ce type de traitement est également disponible dans certains logiciels symboliques commerciaux, comme MAPLE, sous la forme de fonctions d'optimisation du code pour l'exportation des équations dans une routine en utilisant la syntaxe standard comme celle du C, du FORTRAN ou encore de MATLAB. Ceci explique que même en utilisant des formalismes non récursifs, les programmes de génération d'équations de systèmes articulés basés sur ces logiciels commerciaux sont parfois capables de produire des modèles aussi compacts que ceux produits par ROBOTRAN. Toutefois,par expérience, le temps nécessaire à l'optimisation du code est un problème majeur dans le cas de la modélisation de systèmes contenant plus d'une dizaine de corps avec ces logiciels.

Recherche des parents et enfants des équations Lors de la phase de sélection des équations utiles, la liste des équations est parcourue en partant des équations finales marquées comme définissant les résultats du modèle et en remontant de façon récursive à travers les expressions des membres de droite, jusqu'à ce que qu'on ne trouve plus de variables intermédiaires non marquées.

On appelle *équation parent* d'une équation Eq, toutes les équations qui définissent les variables intermédiaires présentes dans le membre de droite de l'équation Eq. Réciproquement, l'équation Eq est une *équation enfant* de toutes ses équations parents. Une équation enfant de l'équation Eq est une équation dont le membre de droite contient la variable intermédiaire qui correspond au membre de gauche de Eq. La table 1.4 illustre ces relations pour une petite série d'équations récursives.

Chaque équation contient donc une liste de pointeurs vers ses parents et ses enfants ainsi que leur nombre respectif. Ces listes sont initialisées lors de la sélection des équations. Elles constituent l'information de base nécessaire à la plupart des traitements suivants.

Equations		Enfants	Parents
ω_{22}	$= \dot{q}_1 \sin q_2$	β_{S22}, β_{62}	-
ω_{32}	$= \dot{q}_1 \cos q_2$	β_{32}, β_{62}	-
$\dot{\omega}_{22}$	$= \dot{q}_1 \dot{q}_2 \cos q_2 + \ddot{q}_1 \sin q_2$	$\beta_{32}, \ldots$	-
$\dot{\omega}_{32}$	$= \ddot{q}_1 \cos q_2 - \dot{q}_1 \dot{q}_2 \sin q_2$	$\ldots$	-
β_{S22}	$= \omega_{22}\omega_{22} - \dot{q}_2 \dot{q}_2$	$\ldots$	ω_{22}
β_{32}	$= \dot{\omega}_{22} + \dot{q}_2 \omega_{32}$	$\ldots$	$\omega_{32}, \dot{\omega}_{22}$
β_{62}	$= \omega_{22}\omega_{32} - \ddot{q}_2$	$\ldots$	ω_{22}, ω_{32}

TAB. 1.4 – Équations, enfants et parents

Recherche des sous-expressions communes Les équations générées sont généralement constituées sur base d'une expression plus ou moins complexe et définissent des variables intermédiaires. La création des équations et donc des variables intermédiaires dépend du formalisme utilisé. Il se peut toutefois que le formalisme ou son implémentation n'aient pas été étudiés avec soin ou bien que certaines variables intermédiaires ne puissent pas être définies de façon générique au niveau du formalisme, mais que des expressions communes puissent être mises en évidence localement. Cela nécessite une inspection globale assez lourde et complexe mais qui peut apporter une économie intéressante et impossible à obtenir au niveau de l'écriture du formalisme. La méthode proposée se déroule en deux étapes, une *décomposition des expressions* en expressions binaires suivie d'une *recherche des expressions binaires identiques*.

Décomposition des équations Les expressions complexes qui constituent les membres de droite des équations sont constituées essentiellement d'opérations arithmétiques de base et de variables intermédiaires. Il y a évidemment aussi quelques données et autres opérations trigonométriques ou autres appels à fonctions externes. Les opérations arithmétiques disponibles dans ROBOTRAN sont toutes binaires [12].

Notre objectif est de décomposer ces expressions complexes en expressions binaires élémentaires. Ainsi les sous-expressions communes pourront ensuite être aisément identifiées par simple inspection de la liste des équations. Pour cela nous allons créer de nouvelles équations et de nouvelles variables intermédiaires pour chaque terme des opérations de base rencontrées, si ces termes sont eux mêmes des opérations de base.

Ainsi l'équation $F = m \cdot (\alpha - \dot{\omega} \cdot L)$ sera décomposée et produira les équations

intermédiaires suivantes :

$$\begin{aligned} F_{dd} &= \dot{\omega} \cdot L \\ F_d &= \alpha_{22} - F_{dd} \\ F &= m \cdot F_d; \end{aligned}$$

On remarquera que :

- les équations intermédiaires reflètent l'organisation des termes dans l'expression de départ,
- les noms des variables intermédiaires sont créés en apposant au nom de la variable définie par l'expression originale, un suffixe correspondant au terme (de gauche ou de droite) que cette nouvelle variable remplace,
- une nouvelle équation est insérée dans la liste des équations avant l'équation dont elle est issue de manière à maintenir un ordre logique.

On appellera *équations binaires* les équations formées par ce processus, en référence à leur membre de droite généralement constitué d'une seule opération arithmétique et de deux termes.

Les listes des équations enfants et parents des équations décomposées sont modifiées pour rester cohérentes avec la nouvelle liste d'équations. Chaque nouvelle équation intermédiaire créée par ce processus compte normalement à ce stade un seul enfant. Toutes les équations ne possèdent plus que deux parents au plus, sauf dans le cas d'un membre de droite comportant un appel à routine externe.

Élimination des doublons Lorsque les équations ont été décomposées en équations binaires, on procède à la comparaison des membres de droite de toutes les équations pour vérifier si deux d'entre-elles sont identiques. Ce traitement est relativement coûteux car sa complexité est de l'ordre de $O(n^2/2)$, si n est le nombre d'opérations. Ainsi, le temps de traitement d'un modèle contenant quelques dizaines de milliers d'opérations peut prendre quelques minutes au lieu de quelques secondes.

Cette méthode peut également être appliquée sur des expressions complexes, mais la probabilité de trouver deux équations identiques est nettement plus faible que dans le cas de la comparaison d'équations binaires.

Chaque équation Eq est comparée à toutes les équations suivantes dans la liste. Lorsque une équation $Eq\prime$ identique à Eq est découverte, elle est supprimée de la liste après que les opérations suivantes aient été effectuées :

1. la variable intermédiaire définie par l'équation $Eq\prime$ est remplacée dans toutes les équations enfants de $Eq\prime$ par la variable définie par l'équation Eq,
2. la liste des parents de chaque équation enfant de $Eq\prime$ est mise à jour,
3. ainsi que la liste des enfants de l'équations originale Eq.

Par exemple, la série d'équations

```
OM13 = qd(2)+qd(3);
...
OpF23 = qd(1)*C2p3*(qd(2)+qd(3));
OpF33 = -qd(1)*S2p3*(qd(2)+qd(3));
```

décomposée en équations binaires

```
OM13 = qd(2)+qd(3);
...
OpF23dd = qd(2)+qd(3);
OpF23d = C2p3*OpF23dd;
OpF23 = qd(1)*OpF23d;
OpF33ddd = qd(2)+qd(3);
OpF33dd = S2p3*OpF33ddd;
OpF33d = qd(1)*OpF33dd;
OpF33 = OpF33d;
```

contient plusieurs fois la sous-expression `qd(2)+qd(3)` qui sera remplacée par `OM13` pour donner finalement

```
OM13 = qd(2)+qd(3);
...
OpF23 = qd(1)*OM13*C2p3;
OpF33 = qd(1)*OM13*S2p3;
```

Ainsi, c'est toujours la première variable intermédiaire de la liste des équations décomposées qui est utilisée pour remplacer ses jumelles dans les expressions où elles apparaissaient.

Elimination des changements de signes Ce traitement un peu particulier est la finalisation de certaines modifications apportées à la routine de soustraction symbolique.

Par défaut les termes des expressions symboliques sont classés selon un ordre basé sur des priorités définies en fonctions de la nature des termes et de leur ordre lexicographique [12], dans le cas des variables. Ainsi, l'opération de soustraction transformait automatiquement l'expression $b-a$ en $-a+b$, ce qui a pour conséquence d'introduire un opérateur de soustraction qui ne correspond pas à une vraie soustraction mais à un changement de signe.

De plus, l'expression $-a+b$ était stockée comme $(0-a)+b$ en utilisant un `ZERO` dans l'expression. Nous avons modifié cette règle de telle sorte que le 0 se retrouve au début de l'expression comme ceci : $-(a-b)$. De cette manière, on provoque une migration systématique des 0 vers le premier terme de toutes les expressions symboliques.

L'objet de l'élimination des changements de signe, consiste alors à détecter les variables auxiliaires définies par des expressions dont le premier terme est un 0 comme celle-ci : $x = -(a - b)$. Ces variables x sont alors redéfinies en éliminant le changement de signe, $x^* = a - b$ et toutes leurs occurrences dans les autres équations sont remplacées pour tenir compte du changement de signe. L'équation $y = d - x + z$ devient alors $y = d + x^* + z$.

Ce traitement réduit le nombre d'opérateurs de soustraction présents dans les expressions symboliques et rend le compte du nombre d'opérateurs de soustraction plus cohérent avec le nombre réel d'opérations de soustraction à effectuer.

Élimination des variables intermédiaires inutiles Certaines variables intermédiaires ne sont utilisées qu'une seule fois. C'est le cas de toutes les variables intermédiaires créées par la procédure de décomposition des équations, mais c'est également le cas pour une partie de celles qui proviennent directement de l'utilisation d'un formalisme récursif.

Une variable intermédiaire inutile peut être détectée par inspection de la liste des ses équations enfants. Elle ne doit avoir qu'une seule équation enfant et n'apparaître qu'une seule fois dans le membre de droite de cette équation enfant.

Dans ce cas, nous remplaçons l'occurrence unique de la variable par sa définition, ce qui peut avoir comme effet secondaire de gagner quelques opérations arithmétiques.

L'élimination d'une variable intermédiaire inutile réduit le nombre d'affectation et donc le nombre de variables locales de la fonction qui implémente le modèle généré. Cela a généralement un effet bénéfique sur la vitesse de compilation du modèle et sur la vitesse d'exécution de celui-ci, ce qui est la motivation principale de ce traitement.

On notera que l'élimination d'une équation intermédiaire implique la mise à jour de la liste des enfants des parents de cette équation et de la liste des parents de l'équation enfant.

Réduction du nombre de variables intermédiaires Il est possible de réduire encore plus le nombre de variables intermédiaires, qu'en éliminant celles qui sont inutiles. L'idée clé est de renommer les variables intermédiaires de façon générique et de réutiliser le nom d'une variable à partir de la dernière équation enfant dans laquelle elle est utilisée.

Les équations sont numérotées dans l'ordre et les variables intermédiaires qu'elles définissent sont renommées en commençant par la première équation. Les noms utilisés sont numérotés et insérés dans une liste avec le numéro de la dernière équation dans laquelle ils apparaissent. Un compteur mémorise le nombre de noms utilisés. A chaque étape i, on vérifie dans la liste des noms si un

nom déjà utilisé est disponible. Un nom est disponible à l'étape i si le numéro de la dernière équation dans laquelle il est utilisé est inférieur à i. Si un nom est disponible, il est réutilisé pour renommer la variable intermédiaire définie par l'équation i. Sinon, un nom supplémentaire est créé, affecté à l'équation i, ajouté à la liste des noms utilisés, et le compteur de noms est incrémenté.

La série de huit équations que voici

```
BS96 = -(OM16*OM16+OM26*OM26);
BeF36 = OpF26+OM16*OM36;
BeF66 = OM26*OM36-OpF16;
AlF16 = AlF14*C6+AlF25*S6;
AlF26 = -(AlF14*S6-AlF25*C6);
OpM16_1 = OpM14_1*C6+OpM25_1*S6;
OpM26_1 = -(OpM14_1*S6-OpM25_1*C6);
AlM16_1 = AlM14_1*C6+AlM25_1*S6;
```

sera récrite en réutilisant des noms de variables `VX_43,VX_42,VX_46,VX_48`.

```
VX_43 = -(VX_60*VX_60+VX_61*VX_61);
VX_42 = VX_44+VX_60*VX_62;
VX_64 = VX_61*VX_62-VX_63;
VX_65 = VX_31*C6+VX_46*S6;
VX_46 = -(VX_31*S6-VX_46*C6);
VX_66 = VX_34*C6+VX_48*S6;
VX_48 = -(VX_34*S6-VX_48*C6);
VX_67 = VX_36*C6+VX_50*S6;
```

On constate que seulement quatre nouvelles variables auxiliaires ont été définies `VX_64,VX_65,VX_66,VX_67` pour la même série d'équations au lieu de huit, initialement.

Appliquée à l'ensemble des équations d'un modèle, ce traitement permet de réduire fortement – 3 fois et plus – le nombre total de variables locales présentes dans la fonction qui implémente le modèle. Le but visé ici est de réduire la taille mémoire temporaire nécessaire pour l'évaluation des équations.

Regroupement des paramètres Une autre manipulation intéressante consiste à renommer les paramètres géométriques et dynamiques du système de manière à les regrouper en un seul vecteur de paramètres. Cela réduit le nombre d'opérations arithmétiques d'adressage indirect lors de l'évaluation du modèle. Lorsque tous les paramètres sont regroupés en un seul tableau à une dimension, leur valeur est accessible par un adressage indirect simple, alors que deux adressages indirects sont nécessaire pour accéder aux éléments d'un tableau à deux dimensions.

Observons la différence entre une série d'équations originales

```
FB16_1 = m[6]*(AlM16_1+OpM26_1*l[3][6]);
FB26_1 = m[6]*(AlM26_1-OpM16_1*l[3][6]);
FB36_1 = m[6]*AlM35_1;
CM16_1 = In[1][6]*OpM16_1-FB26_1*l[3][6];
CM26_1 = In[5][6]*OpM26_1+FB16_1*l[3][6];
CM36_1 = In[9][6]*OpM35_1;
```

et la même série d'équations où les données ont été regroupées dans un tableau `Dat` :

```
FB16_1 = Dat[9]*(AlM16_1+OpM26_1*Dat[5]);
FB26_1 = Dat[9]*(AlM26_1-OpM16_1*Dat[5]);
FB36_1 = Dat[9]*AlM35_1;
CM16_1 = Dat[20]*OpM16_1-FB26_1*Dat[5];
CM26_1 = Dat[21]*OpM26_1+FB16_1*Dat[5];
CM36_1 = Dat[22]*OpM35_1;
```

Ce traitement offre également une plus grande simplicité d'utilisation des fonctions générées dans des environnements de simulation qui n'offrent pas toujours des interfaces de programmation très flexibles en ce qui concerne la gestion des tableaux à plusieurs dimensions. C'est précisément le cas de l'interface MEX de la version actuelle de MATLAB[12].

1.6.3 Dérivation symbolique

Dans de nombreuses d'applications, il est utile voire indispensable de disposer des dérivées partielles de certaines grandeurs en fonctions de certains paramètres.

Parmi ces applications, nous pouvons citer

- *l'intégration numérique* de systèmes dit "raides" nécessite de disposer des matrices tangentes $M \triangleq \frac{\partial \Phi}{\partial \ddot{q}^T}$, $G \triangleq \frac{\partial \Phi}{\partial \dot{q}^T}$ et $K \triangleq \frac{\partial \Phi}{\partial q^T}$ obtenues par dérivation partielle de la forme implicite (1.36) des équations du mouvement en fonction des coordonnées, vitesses et accélérations généralisées.
- *le contrôle de système* pour lequel il est souvent intéressant de disposer du modèle d'état linéarisé que l'on peut exprimer à l'aide des mêmes matrices M, G et K. La matrice Jacobienne J du repère de l'outil est également très utilisée pour le contrôle des robots.
- *l'optimisation paramétrique* pour laquelle la connaissance de la sensibilité $\frac{\partial \Phi}{\partial p}$ d'une fonction objectif Φ par rapport à un paramètre p est nécessaire à l'utilisation de certaines méthodes déterministes.
- *l'identification dynamique* où la détermination du jeu minimal $\{p\}$[13] de paramètres dynamiques d'un système arborescent [91, 101], repose sur

[12] MATLAB release 12

[13] $\{p\} \subseteq \{\delta\}$ où $\{\delta\}$ est l'ensemble des paramètre barycentriques

la reformulation $Q{=}\Psi(q,\dot{q},\ddot{q},...)\,\delta$ du modèle dynamique inverse (1.36). Cette formulation est linéaire en fonction des paramètres barycentriques δ [19, 20]. Il est nécessaire de disposer de la matrice $\Psi = \frac{\partial \Phi}{\partial \delta^T}$ sous une forme explicite afin de pouvoir déterminer les valeurs des paramètres par des techniques de régression linéaire.

Pour toutes ces applications, la possibilité de disposer des matrices de dérivées partielles sous forme symbolique comporte les avantages suivant :

- Le *temps calcul* : l'évaluation de la formulation symbolique est plus rapide que l'évaluation numérique par des méthodes de différences finies ou de filtrage numérique.
- La *précision numérique* : la forme symbolique est similaire à une dérivée analytique alors qu'un processus de dérivation numérique implique une erreur d'approximation.
- La *portabilité* : la génération automatique des dérivées partielles permet d'obtenir leur définition sous forme de routine indépendante facile à implanter dans n'importe quel environnement de calcul numérique.

Hormis les procédures numériques, différentes méthodes permettent d'obtenir les dérivées partielles sous forme symbolique :

- la *génération symbolique directe*. Cette méthode requiert qu'une formulation explicite des dérivées partielles soit disponible et implémentée dans un environnement de calcul symbolique. Elle est donc limitée à quelque cas particuliers de dérivées partielles. C'est le cas pour la génération récursive des matrices Jacobienne J et de masse M dans ROBOTRAN.
- la *dérivation symbolique récursive*. Cette procédure qui se base sur un parcours récursif d'une liste d'équations permet d'en générer les dérivées partielles équation par équation. Elle était déjà utilisée dans la version originale de ROBOTRAN pour la génération de la matrice Ψ en identification et pour le calcul des matrices tangentes dans le cas de la simulation de poutres flexibles [102]. Nous avons généralisé cette procédure afin de pouvoir générer les dérivées partielles de n'importe quelle expression produite par ROBOTRAN en fonction de n'importe quel paramètre.
- la *dérivation automatique*. Cette méthode tout à fait similaire à la dérivation symbolique récursive, ne s'applique pas à la liste des équations au niveau du générateur symbolique, mais à la routine générée par celui-ci par le biais de l'utilisation d'outils génériques de dérivation automatique tels que ADIFOR, ADIC ou ADOL-C [103]. Ces outils permettent de produire, a partir d'un fichier contenant une routine qui implémente une fonction quelconque du type $y = F(x)$ avec $F : R^n \mapsto R^m$, un autre fichier contenant une routine qui implémente la fonction F ainsi que ses dérivées partielles $\frac{\partial F}{\partial x}$.

Bien qu'elle soit plus flexible et d'usage plus général, la technique de dérivation automatique produit néanmoins du code plus long et plus lourd en terme d'évaluation, que la méthode de dérivation symbolique récursive appliquée au

sein d'un logiciel de génération symbolique d'équations [103].

Ces considérations justifient donc l'utilité d'une procédure générique permettant de dériver les équations récursives générées par ROBOTRAN en fonction de n'importe quel jeu de variables ou de paramètres existant dans les équations.

Dérivation symbolique récursive

Le principe de la dérivation récursive est assez simple. Il s'agit de parcourir la liste des équations en commençant par le début. Pour chaque équation, on applique les règles de dérivation au membre de droite, et on crée une nouvelle équation qui définit la dérivée partielle. Certains termes du membre de droite correspondent à des variables auxiliaires définies par des équations précédentes, dont les dérivées partielles ont déjà été calculées précédemment et peuvent donc être réutilisée.

Le processus est illustré sur la figure 1.25 pour deux équations qui définissent des variables auxiliaires x et y. Leurs dérivées partielles relatives aux paramètres $p = \{a, c, q\}$ sont calculées.

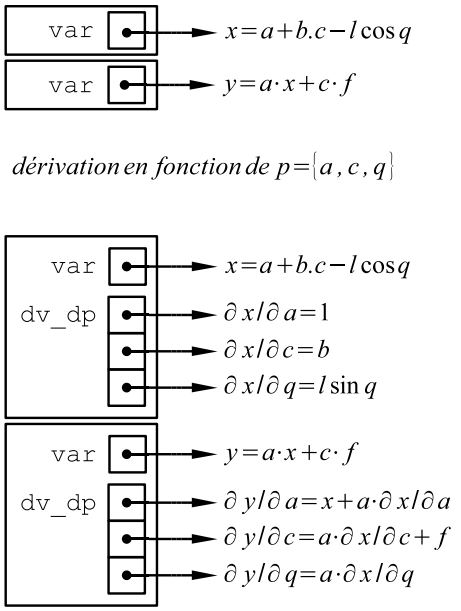


FIG. 1.25 – Dérivation symbolique récursive

Pour chaque équation, on crée une liste de références vers les équations qui correspondent à ses dérivées partielles. Ainsi, l'équation qui définit x possède

naturellement une référence, appelée `var` sur la figure 1.25, à x. Lors du calcul des dérivées partielles de x par rapport à chacun des paramètres p, des équations sont créées qui définissent $\frac{\partial x}{\partial a}$, $\frac{\partial x}{\partial c}$, etc. Les références vers ces nouvelles équations sont stockées dans la liste `dv_dp` de l'équation x afin qu'elles puissent être utilisée ultérieurement, lors du calcul des dérivées partielles de l'équations qui définit y, dans cet exemple.

Deux cas particuliers méritent notre attention. Il s'agit d'une part des équations qui contiennent un appel à une fonction externe, et d'autre part du cas de la dérivation par rapport à une coordonnée articulaire.

Appel à fonction externe Lorsque la procédure de dérivation rencontre une expression qui correspond à un appel à une fonction externe *Fct*, cette expression est transformée en un appel à une nouvelle fonction externe Fct_dFdp équivalente à celle que l'on peut obtenir en dérivant la fonction externe originale *Fct* à l'aide d'un outil de dérivation automatique. Cette nouvelle fonction devra renvoyer la valeur de f ainsi que ses dérivées partielles.

En effet, la cohérence et l'exactitude du schéma d'équations récursives ne peut être maintenu que si l'utilisateur peut fournir les valeurs des dérivées partielles de la fonction par rapport aux paramètres. Il pourra recourir à l'utilisation d'outil de dérivation automatique si le calcul analytique et l'implémentation des expressions des dérivées partielles est difficile ou impossible à la main.

En pratique une équation contenant un appel à une fonction

$$f = \mathsf{Fct}(x, y)$$

qui renvoie un scalaire f sera remplacée par les équations suivantes

$$\begin{aligned} X &= \{x, \partial x/\partial p_1, \partial x/\partial p_2, \ldots\} \\ Y &= \{y, \partial y/\partial p_1, \partial y/\partial p_2, \ldots\} \\ F &= \mathsf{Fct\_dFdp}(X, Y) \end{aligned}$$

avec

$$F = \{f, \partial f/\partial p_1, \partial f/\partial p_2, \ldots\}$$

où F est cette fois un vecteur, contenant la valeur de f ainsi que ses dérivées partielles.

Si la fonction originale renvoie un vecteur, la nouvelle fonction renverra une matrice dont la première colonne est le vecteur original et les autres colonnes contiennent les dérivées partielles. Cette matrice sera toutefois stockée ligne par ligne dans un vecteur car, à ce jour, ROBOTRAN ne supporte pas encore les symboles correspondant à des éléments de type matriciel.

Dérivée par rapport à une coordonnée articulaire Lorsque le système est soumis à des contraintes algébriques, provenant de boucles cinématiques[14] par exemple, et que parmi les paramètres se trouve une coordonnée articulaire, il y a lieu de tenir compte du couplage entre les coordonnées indépendantes q_u et les coordonnées dépendantes q_v. La matrice de couplage[15] B_{vu} est utilisée à cet effet.

On écrira alors la Jacobienne d'une fonction par rapport aux coordonnées articulaires indépendantes q_u de la façon suivante :

$$\frac{dF}{dq_u} = \frac{\partial F}{\partial q_u} + \frac{\partial F}{\partial q_v} \cdot \frac{\partial q_v}{\partial q_u} = \frac{\partial F}{\partial q_u} + \frac{\partial F}{\partial q_v} \cdot B_{vu}$$

Ainsi la dérivée partielle $\frac{\partial q_v^i}{\partial q_u^j}$ d'une coordonnée articulaire dépendante q_v^i par rapport à une coordonnée articulaire indépendante q_u^j est donnée par l'élément B_{uv}^{ij} de la matrice de couplage des vitesses. Cette matrice est définie par la relation $\dot{q}_v = B_{vu} \cdot \dot{q}_u$.

1.7 Applications

Dans cette section, nous illustrons plus en détails certains concepts présentés dans ce chapitre, à l'aide de deux systèmes.

Le premier est un bras robotique à structure série. Nous utilisons ce système pour observer la complexité des modèles obtenus par les formalismes récursifs Newton Euler et $O(N)$, ainsi que pour observer les gains obtenus par l'application des nouveaux traitements symboliques implantés dans ROBOTRAN.

Le second système est une moto. Outre les résultats de formalismes et les simplifications symboliques, nous montrons les équations qui sont générées par ROBOTRAN pour le calcul des forces de liaison point à point et des forces de contact roue/sol, conjointement aux équations dynamiques, en vue d'obtenir le jeu complet des équations du mouvement du système, sous forme symbolique.

1.7.1 Le bras robot manipulateur PUMA

Le robot PUMA est un robot manipulateur série à six degrés de liberté construit par Unimation. Il est représenté sur la figure 1.26.

Il est modélisé dans ROBOTRAN par une chaîne de six corps reliés par des articulations rotoïdes `R`. La séquence d'articulations et de corps est reprise dans la table 1.5.

[14] Ces notions sont développées dans le chapitre suivant
[15] voir section 2.4.2

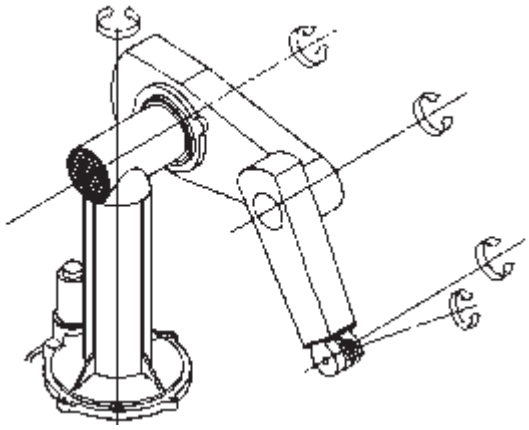


FIG. 1.26 – Le robot PUMA

Sol		
R3	q1	Tronc
R1	q2	Épaule
R1	q3	Avant-bras
R3	q4	Poignet
R1	q5	fictif
R3	q6	Effecteur

TAB. 1.5 – Structure du robot PUMA

Modèles dynamiques

Nous avons généré plusieurs modèles dynamiques en utilisant les formalismes implicites et semi-explicites présentés, ainsi qu'une variante du formalisme récursif Newton-Euler qui utilise un jeu minimal de paramètres barycentriques et un formalisme de type $O(N)$, dont la complexité évolue linéairement avec le nombre N d'articulations. Les résultats obtenus correspondent naturellement avec ce qui avait déjà été annoncé dans [12] en ce qui concerne l'avantage des formalismes récursifs pour les systèmes de taille réduite.

La nouveauté est ici l'introduction de la résolution algébrique symbolique qui permet d'obtenir *explicitement* les accélérations pour les modèles dynamiques directs générés à partir de formalisme récursif semi-explicites.

Deux niveaux d'optimisation symbolique sont utilisés. Une optimisation standard qui implique un test de simplification à la création des chaque équa-

tion par une expansion de niveau deux et une élimination des variables intermédiaires "inutiles". L'optimisation avancée implique en plus la recherche et l'élimination des sous expressions communes par décomposition complète et régénération du schéma récursif d'équations.

Q	standard	optimisé
barycentrique	359	325
classique	390	361

TAB. 1.6 – Complexité des modèles dynamiques inverses du robot PUMA

Dynamique inverse La table 1.6 reprend les nombres d'opérations obtenus en utilisant la formulation implicite pour générer le modèle dynamique inverse du robot.

Dynamique directe La table 1.7 reprend les nombres d'opérations obtenus en utilisant la formulation semi-explicite pour générer des modèles dynamiques semi direct du robot. Ces modèles permettent de calculer la matrice de masse M et le vecteur c des termes non linéaires de l'équation (1.37).

M et c	standard	optimisé
barycentrique	601	527
classique	747	689

TAB. 1.7 – Complexité des modèles dynamiques directs semi-explicites du robot PUMA

La table 1.8 reprend les nombres d'opérations obtenus en utilisant la formulation semi-explicite suivie d'une résolution du système $M\ddot{q} + c = Q$ pour générer des modèles dynamiques direct du robot avec les formalismes Newton Euler récursif et obtenir les accélérations articulaires $\ddot{q}$. Celle-ci sont directement produites lorsqu'on utilise le formalisme $O(n)$.

$\ddot{q}$	standard	optimisé
barycentrique	785	711
classique	931	873
Ordre(n)	1381	1220

TAB. 1.8 – Complexité des modèles dynamiques directs explicites du robot PUMA

On constate que pour des petits systèmes[16], les formalismes récursifs ont l'avantage sur les formalisme $O(n)$. L'utilisation des paramètres barycentriques est également un moyen efficace pour réduire encore le nombre d'opérations des modèles.

Simplifications symboliques

Nous prenons ici comme exemple la génération du modèle dynamique direct explicite du robot PUMA, pour illustrer quelques-unes des simplifications symboliques effectuées sur les équations. Un modèle équivalent généré avec la version originale de ROBOTRAN contiendrait un peu plus de 970 opérations, pour le calcul explicite des accélérations généralisées.

On effectue une première génération du modèle sans activer les simplifications symboliques avancées.

On obtient un premier modèle qui comporte 958 opérations arithmétiques et trigonométriques. Dans ce modèle, les variables intermédiaires relatives aux sinus et cosinus des coordonnées indirectes provenant des articulations rotoïdes consécutives à axes parallèles ont toutefois déjà été créées par les routines trigonométriques. c'est le cas ici des articulations de l'épaule q_2 et du coude q_3 du robot.

```
% Trigonometric Variables

C2p3 = C2*C3-S2*S3;
S2p3 = C2*S3+S2*C3;
```

En activant l'expansion partielle, au niveau 1 seulement, on obtient un modèle qui compte 946 opérations.

La séquence initiale d'équations que voici

```
OM22 = qd(1)*S2;
OM32 = qd(1)*C2;
OpF22 = qd(1)*qd(2)*C2;
OpF32 = -qd(1)*qd(2)*S2;
OM23 = OM22*C3+OM32*S3;
OM33 = -(OM22*S3-OM32*C3);
OpF23 = C3*(OpF22+qd(3)*OM32)+S3*(OpF32-qd(3)*OM22);
OpF33 = C3*(OpF32-qd(3)*OM22)-S3*(OpF22+qd(3)*OM32);
```

est transformée en la suivante, où apparaissent les sinus et cosinus des coordonnées indirectes $q_2 + q_3$ provenant des articulations rotoïdes 2 et 3 qui sont consécutives à axes parallèles :

[16] Nombre de corps en série inférieur à 10 [12]

```
OM22 = qd(1)*S2;
OM32 = qd(1)*C2;
OpF22 = qd(1)*qd(2)*C2;
OpF32 = -qd(1)*qd(2)*S2;
OM23 = qd(1)*S2p3;
OM33 = qd(1)*C2p3;
OpF23 = qd(1)*C2p3*(qd(2)+qd(3));
OpF33 = -qd(1)*S2p3*(qd(2)+qd(3));
```

Douze opérations ont été éliminées dans cette séquence d'équations. Dans le cas du robot PUMA, une expansion plus poussée, au niveau 2 ou 3 ne produit pas plus de simplifications.

La simplification suivante consiste à activer l'élimination des variables intermédiaires qui ne sont utilisées qu'une seule fois. La taille du modèle se réduit à 931 opérations. Voici quelques exemples de substitution et d'éliminations effectuées :

```
FB26_4 = -m(6)*OpM16_4*l(3,6);
CM16_4 = In(1,6)*OpM16_4-FB26_4*l(3,6);
```

devient

```
CM16_4 = OpM16_4*(In(1,6)+m(6)*l(3,6)*l(3,6));
```

Une multiplication et une soustraction ont disparu.

De même, les équations

```
FB16_5 = -m(6)*l(3,6)*S6;
FB26_5 = -m(6)*l(3,6)*C6;
CM16_5 = In(1,6)*C6-FB26_5*l(3,6);
CM26_5 = -(In(5,6)*S6-FB16_5*l(3,6));
CM15_5 = CM16_5*C6-CM26_5*S6;
```

deviennent

```
CM15_5 = C6*C6*(In(1,6)+m(6)*l(3,6)*l(3,6)) +
         S6*S6*(In(5,6)+m(6)*l(3,6)*l(3,6));
```

On remarque dans ce second cas que si les termes d'inertie `In(1,6)` et `In(5,6)` étaient identiques, on obtiendrait des simplifications encore plus importantes.

Remarque Ces deux exemples font également apparaître des combinaisons de paramètres dynamiques constants. Ces combinaisons pourraient également être détectées et remplacées par un seul paramètre pré calculé. Bien

que cette recherche pourrait être effectuée au niveau des opérations arithmétiques de base, elle fait l'objet d'une procédure de *regroupement systématique* lors de l'utilisation du formalisme à paramètres barycentriques et n'a donc pas été implémentée pour cette raison. Les simplifications obtenues sont néanmoins généralement très importantes dans le cas des modèles de robots séries, comme on a pu le constater en comparant les nombres d'opérations des modèles dans les tables 1.6, 1.7 et 1.8.

Le dernier stade de simplification consiste en la recherche et l'élimination des sous expressions identiques par décomposition totale et reconstruction du schéma récursif d'équations. Cette opération est relativement lourde pour les modèles de gros systèmes comptant plusieurs dizaines de milliers d'opérations, comme des véhicules automobiles ou des trains, mais peut être effectuées sans problème sur de petits mécanismes ou des robots à structure série ou parallèle composés de dix à vingt corps et comptant quelques milliers d'opérations, sans augmentation sensible du temps de génération du modèle.

Le modèle obtenu avec cette dernière procédure de simplification contient 873 opérations.

On constate que les nouvelles procédures de simplification symbolique apportent un gain de 10 à 15% sur le nombre d'opérations selon la topologie et la taille du système.

1.7.2 Un modèle simple de moto

Le modèle de moto présenté ici est inspiré de [104].

Nous utilisons ce système pour illustrer la génération symbolique d'équations utiles pour le calcul de forces point à point agissant entre deux corps ainsi que pour le calcul de forces de contact, entre les roues et le sol.

La figure 1.27 représente le schéma d'une moto. Ce véhicule simple possède une structure arborescente. La séquence des articulations et des corps qui le composent est reprise dans la table 1.9.

Le châssis (corps 6) qui possède six degrés de liberté est relié au sol par une chaîne cinématique fictive constituée de six articulations fictives élémentaires. Le bras oscillant (corps 7) et le châssis sont soumis aux efforts qui leur sont appliqués par un ensemble ressort amortisseur présent dans la suspension de la moto. Cet élément n'est pas modélisé comme un corps à part entière du système, mais son action sur les corps 6 et 7 est modélisée par le calcul d'une force F_{lnk} de liaison point à point. L'articulation 10 est introduite pour obtenir un repère qui possède l'inclinaison correcte par rapport au châssis et dont un axe est aligné avec l'axe de rotation du guidon. Cette articulation est fictive et bloquée. L'articulation 11 est introduite pour prendre en compte l'effet de torsion longitudinal du châssis. Les corps 8 et 14 sont les deux roues de la moto

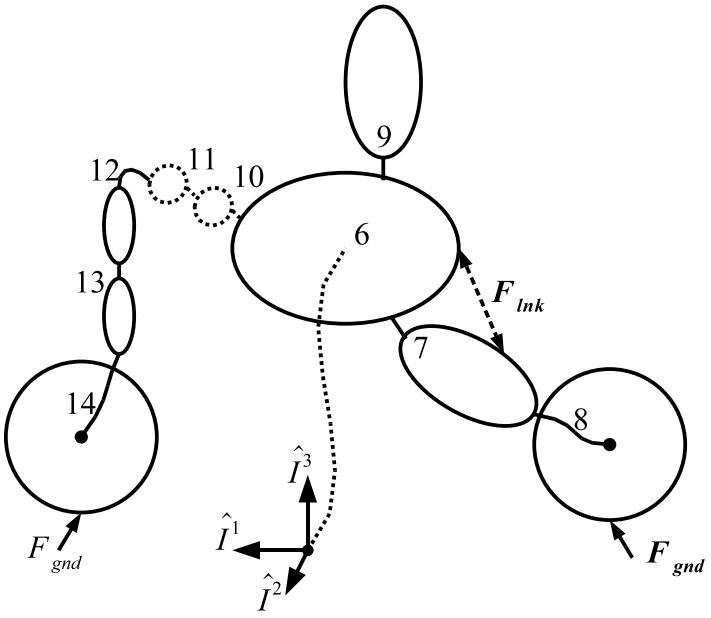


FIG. 1.27 – Représentation schématique d'une moto

qui sont en contact avec le sol, en situation normale. Le système possède donc 13 degrés de liberté, vu que l'articulation 10 est bloquée.

Dynamique directe La table 1.10 reprend les nombres d'opérations obtenus en utilisant la formulation semi-explicite suivie d'une résolution symbolique du système $M_r\ddot{q}_f+F_r = 0$ pour générer des modèles dynamiques directs de la moto, mais sans tenir compte des équations nécessaires pour les calculs cinématiques et dynamiques relatifs aux forces de liaison point à point ou de contact des roues avec le sol. On a généré également un modèle dynamique direct en utilisant le formalisme $O(N)$ afin de comparer les deux approches.

On constate que pour ce système, le formalisme Newton-Euler récursif classique perd l'avantage par rapport au formalisme $O(N)$, car le nombre de degrés de libertés et donc la taille du système d'équations à résoudre pour obtenir les expressions explicites des accélérations entraîne un coût calcul plus important en raison de la complexité $O(N^3)$ de la résolution. L'utilisation des paramètres barycentriques reste toutefois le meilleur choix pour obtenir le modèle le plus compact.

Le modèle dynamique direct explicite dans lequel sont incluses les équations de la cinématique des points d'applications des forces de liaison et de contact,

Sol		
T1	q1	fictif
T2	q2	fictif
T3	q3	fictif
R3	q4	fictif
R1	q5	fictif
R2	q6	Châssis
R2	q7	Bras oscillant
R2	q8	Roue arrière
R1	q9	Pilote
R2	q10	fictif
R1	q11	fictif
R3	q12	Tête de fourche
T3	q13	Fourreaux
R2	q14	Roue avant

TAB. 1.9 – Articulations et corps d'une moto

	standard	optimisé
barycentrique	2896	2576
classique	3371	2950
Ordre(n)	3145	2821

TAB. 1.10 – Complexité des modèles dynamiques direct explicites de la moto.

ainsi que la projection de ces forces sur les corps, contient $\boxed{3915}$ opérations, avec les options de génération standard, et $\boxed{3382}$ avec toutes les options de simplification, ce qui correspond à une réduction de 13%. Dans ce modèle complet, les forces de liaisons et de contact des roues avec le sol sont toutefois obtenues en appelant des routines externes qui implémentent le modèle constitutif du combiné ressort amortisseur pour la suspension et les équations constitutives du contact des roues avec le sol. Les nombres d'opérations nécessaires pour l'évaluation de ces modèles n'interviennent donc pas dans le décompte des opérations du modèle généré par ROBOTRAN.

Calcul des forces de suspension Afin d'illustrer la génération symbolique des équations relatives au calcul des forces de liaison point-à-point et leur application sur les corps reliés, nous insérons ci-après une partie du code source du modèle généré par ROBOTRAN. Le code est structuré de façon intelligible et n'est pas trop long pour le système choisi. Les équations générées sont reparties en trois groupes :

1. Tout d'abord, on trouve les équations cinématiques qui permettent de calculer la distance `Z` entre les points d'ancrage de l'élément qui relie les corps, le combiné ressort-amortisseur dans ce cas, ainsi que sa dérivée temporelle `Zd`.
2. La force de liaison `Flink` qui s'applique sur chacun des corps est obtenue en appelant une routine extérieure `LinkForce(...)` qui reçoit tous les arguments nécessaires pour évaluer le modèle constitutif de l'élément de liaison.
3. Ensuite viennent les équations de projection de la force de liaison sur chacun des corps reliés. On y calcule les composantes de la force de liaison dans les repères locaux des corps ainsi que le couple de transport de la force du point d'application aux points de référence des corps.

```
% Link Kinematics: Distance Z , Relative Velocity Zd

Plnk11 = RLlnk6_116+d(1,7)-dbp(1,1);
Plnk31 = RLlnk6_316+d(3,7)-dbp(3,1);
Z1 = sqrt(Plnk11*Plnk11+Plnk31*Plnk31);
e11 = Plnk11/Z1;
e31 = Plnk31/Z1;
Zd1 = -qd(7)*(RLlnk6_116*e31-RLlnk6_316*e11);

% Link Force Computation

Flink1 = LinkForce(Z1,Zd1,tsim,pdbl,pint,1);

% Link Dynamics : Force projection on body-fixed frames

fPlnk11 = Flink1*e11;
fPlnk31 = Flink1*e31;
frc(1,6) = frc(1,6)+fPlnk11;
frc(3,6) = frc(3,6)+fPlnk31;
trq(2,6) = trq(2,6)+fPlnk11*dbp(3,1)-fPlnk31*dbp(1,1);

fSlnk11 = Flink1*(e11*C7-e31*S7);
fSlnk31 = Flink1*(e11*S7+e31*C7);
frc(1,7) = frc(1,7)-fSlnk11;
frc(3,7) = frc(3,7)-fSlnk31;
trq(2,7) = trq(2,7)-fSlnk11*dbs(3,1)+fSlnk31*dbs(1,1);
```

Calcul des forces de contact avec le sol Nous illustrons également la génération symbolique des équations permettant de calculer les forces de contact en insérant une partie seulement, du code source du modèle généré par ROBOTRAN.

Les équations générées sont également reparties en trois groupes :

1. Tout d'abord, on trouve les équations cinématiques qui permettent de calculer les variables indiquant la configuration de la roue par rapport au sol ainsi que la cinématique du point de contact. On notera l'insertion

d'un appel à une fonction externe qui fournit la valeur de la hauteur du sol.

2. Les composantes de force et de couple de contact exprimées dans le repère au sol sont obtenues en appelant la routine `WheelForcesT(...)`, à laquelle on fournit tous les arguments nécessaires pour évaluer le modèle constitutif du contact pneu/sol.
3. Ensuite viennent les équations de projection des résultantes de force et de couple de contact sur la roue. On y calcule les composantes dans le repère de la roue en rotation ainsi que le couple de transport de la force du point de contact au centre de la roue qui est le point de référence du corps.

On notera également, que la fonction `WheelForcesT(...)` renvoie les six composantes de force et de couple de contact dans une variable unique `whlWr1` qui n'est donc pas une variable scalaire, mais un vecteur de six éléments. Les composantes sont alors utilisées dans les équations en utilisant les variables indicées `whlWr1(1)`, `whlWr1(2)`,etc.

```
% Wheel/Ground Contact Kinematics

...
%
tgta1 = RO_0_37/RO_0_97;
cosa1 = 1/sqrt(1+tgta1*tgta1);
sina1 = cosa1*tgta1;

Zgnd1 = GroundLevel(PO_0_18,PO_0_28,tsim,pdbl,pint,1);

rwhl1 = (PO_0_38-Zgnd1)/C5;
rzI11 = -rwhl1*(RO_0_17*sina1+RO_0_77*cosa1);
rzI21 = -rwhl1*(RO_0_27*sina1+RO_0_87*cosa1);
rzI31 = -rwhl1*(RO_0_37*sina1+RO_0_97*cosa1);
Vctw11 = VI_0_18+OM_0_28*rzI31-OM_0_38*rzI21;
Vctw21 = VI_0_28-OM_0_18*rzI31+OM_0_38*rzI11;
Vctw31 = VI_0_38+OM_0_18*rzI21-OM_0_28*rzI11;
Vcts11 = C4*Vctw11+S4*Vctw21;
Vcts21 = -S4*Vctw11+C4*Vctw21;
Vws11 = C4*VI_0_18+S4*VI_0_28;
slipang1 = atan(Vcts21/Vws11);
gliss1 = -Vcts11/(cos(slipang1)*sqrt(VI_0_18*VI_0_18+VI_0_28*VI_0_28));

% Wheel/Ground Contact Force Computation

whlWr1 = WheelForcesT(rwhl1,slipang1,q(5),gliss1,Vctw31,tsim,pdbl,pint,1);

% Wheel/Ground Contact Force projection on body-fixed frames

fl11 = whlWr1(1);
fl21 = whlWr1(2)*C5+whlWr1(3)*S5;
fl31 = -(whlWr1(2)*S5-whlWr1(3)*C5);
```

```
fw11 = cosa1*fl11+fl31*sina1;
fw31 = cosa1*fl31-fl11*sina1;
frc(1,8) = frc(1,8)+fw11*C8-fw31*S8;
frc(2,8) = frc(2,8)+fl21;
frc(3,8) = frc(3,8)+fw11*S8+fw31*C8;
cw11 = cosa1*(whlWr1(4)+fl21*rwhl1)-sina1*(whlWr1(5)*S5-whlWr1(6)*C5);
cw31 = -(cosa1*(whlWr1(5)*S5-whlWr1(6)*C5)+fl21*rwhl1*sina1+sina1*whlWr1(4));
trq(1,8) = trq(1,8)+cw11*C8-cw31*S8;
trq(2,8) = trq(2,8)-rwhl1*(cosa1*fw11-fw31*sina1)+whlWr1(5)*C5+whlWr1(6)*S5;
trq(3,8) = trq(3,8)+cw11*S8+cw31*C8;
...
```

1.8 Conclusions

Dans ce premier chapitre, nous avons présenté les conventions de modélisation ainsi que les formalismes récursifs utilisés pour la génération symbolique de modèles cinématiques et dynamiques. Grâce à quelques extensions des librairies symboliques du logiciel ROBOTRAN, nous avons pu générer des *modèles symboliques complets* pour l'étude de systèmes multicorps à topologie arborescente.

L'insertion d'appels à des routines externes, au sein des équations récursives, permet de connecter facilement le modèle mécanique du système multicorps avec des modèles externes, pour le calcul de forces d'interaction avec l'environnement, par exemple. L'ajout de nouvelles *natures* d'expressions symboliques dans ROBOTRAN, telles que les fonctions à plusieurs arguments, permet d'utiliser ces appels de routine sans rompre le chaînage récursif des équations, ce qui offre l'avantage de maintenir la possibilité d'effectuer des manipulations symboliques des modèles, telles que la dérivation symbolique récursive, par exemple.

L'ajout de méthodes de résolution symbolique de systèmes d'équations linéaires a également rendu possible la génération de *modèles dynamiques directs sous forme explicite* en partant du formalisme Newton-Euler semi-explicite.

Finalement, nous avons également implémenté quelques nouvelles fonctionnalités relatives à la simplification symbolique des équations récursives. Nous avons obtenu une réduction du nombre d'opérations de l'ordre de 10 à 15% par rapport aux modèles équivalents générés par la version originale du logiciel [12].

Chapitre 2

Modélisation des systèmes à topologie bouclée

La plupart des systèmes articulés réels comportent des boucles cinématiques. Leur structure n'est alors pas strictement arborescente, mais comporte des chaînes cinématiques qui se referment sur elles-mêmes. Ces systèmes ont toujours un nombre de degrés de liberté inférieur au nombre d'articulations élémentaires présentes dans leur structure. C'est le cas, par exemple, de la plupart des mécanismes de suspension des véhicules automobiles ainsi que des mécanismes de commandes des gouvernes des avions ou des pales des hélicoptères. Les robots parallèles en sont également un exemple classique bien connu.

Dans ce chapitre, nous allons montrer comment on peut exploiter avantageusement l'approche symbolique pour traiter ces systèmes et en générer les équations du mouvement.

Nous exposons une méthode, basée sur le calcul symbolique, qui permet de générer et de résoudre les équations de contraintes et de générer[1] un jeu réduit d'équations purement différentielles pour l'étude du mouvement des systèmes à topologie bouclée.

Cette méthode n'est pas limitée au formalisme récursif présenté. D'autres formalismes ou d'autres choix de coordonnées tels que ceux mentionnés dans le chapitre d'introduction, pourraient être utilisés également.

Toutefois, afin de réutiliser les formalismes récursifs présentés dans le chapitre précédent et implantés dans le logiciel ROBOTRAN, nous transformons les structures bouclées en structures arborescentes par une technique de coupure et génération d'équations de contraintes associées à ces coupures.

On trouve dans la littérature d'autres techniques de traitement des boucles cinématiques dont celle proposée par l'équipe des Professeurs Hiller et Kecske-

[1] sous forme symbolique

méthy [45, 46, 47]. Leur technique permet de conserver toutes les articulations des boucles et d'obtenir, dans la plupart des cas, une solution analytique des équations de contraintes assurant la fermeture de ces boucles. Bien que celle-ci semble avantageuse pour beaucoup des systèmes, elle n'a pas été implémentée dans ROBOTRAN à ce jour[2] car cela nécessiterait de modifier les conventions de description des systèmes utilisées par ce logiciel, ce qui sort du cadre de ce travail.

2.1 Introduction

Comme nous l'avons montré dans le chapitre traitant de la modélisation des systèmes simples, l'utilisation de formulations récursives pour la génération des équations cinématiques et dynamiques repose sur la représentation arborescente de la topologie du système. Dans le cas des systèmes à topologie fermée, cette représentation n'est pas unique. On peut en effet construire plusieurs arbres différents couvrant les corps et articulations du système. Le choix d'un de ces arbres possibles, correspond en quelque sorte à l'ouverture d'une boucle cinématique.

Différentes façon d'ouvrir une structure fermée sont illustrées sur la figure 2.1.

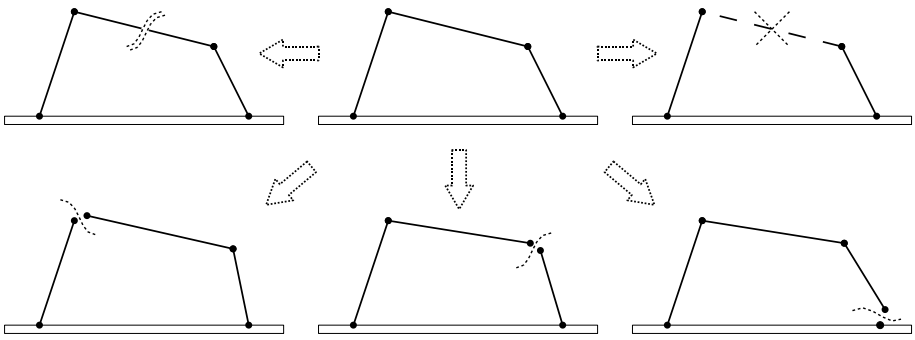


FIG. 2.1 – Exemples d'ouverture d'une boucle cinématique

L'ouverture d'une chaîne fermée de corps et d'articulations peut être faite en *coupant* soit un corps, soit une articulation.

Dans les deux cas, le système arborescent obtenu par *coupure* n'est plus équivalent au système original. En effet la coupure d'un corps génère un système avec un corps de plus et la coupure d'une articulation correspond en quelque sorte à la supprimer. D'autre part, le corps ou l'articulation victime

[2] Il s'agit toutefois d'une des perspectives envisagées

de la coupure est le siège de forces et couples de liaisons internes qui ont une influence sur le comportement du système.

Deux conditions impératives doivent être respectées, si on désire reproduire exactement le comportement du système original :

1. les contraintes géométriques imposées par le corps ou l'articulation coupée doivent être vérifiées à tout instant,
2. les forces (et couples) internes originales doivent être prises en compte.

Contraintes géométriques Les contraintes géométriques à respecter prendront la forme d'équations algébriques généralement non-linéaires et dans lesquelles le temps n'intervient pas explicitement. Elles peuvent être générées symboliquement pour différents types de coupure [12, 13], ainsi que leurs dérivées premières et secondes. Ces équations sont notées :

$$h(q) = 0 \tag{2.1}$$

$$\dot{h}\left(q, \dot{q}\right) = J(q)\dot{q} = 0 \tag{2.2}$$

$$\ddot{h}\left(q, \dot{q}, \ddot{q}\right) = J(q)\ddot{q} + \dot{J}(q, \dot{q})\dot{q} = 0 \tag{2.3}$$

Forces internes Les forces internes de coupure présentes dans le système original deviennent des forces externes pour le système ouvert. Ces forces d'interaction dépendent des contraintes et de la dynamique du système. L'utilisation du *Principe des Puissances Potentielles* nous permet de calculer ces interactions et de les introduire dans les équations du mouvement obtenues par un formalisme récursif appliqué au système arborescent [13].

En appelant Q' les forces généralisées d'interaction dans les coupures, on peut récrire les équations du mouvement (1.37) du système arborescent sous leur forme semi-explicite comme ceci :

$$M(q)\ddot{q} + c(\dot{q}, q, t, F_{ext}, L_{ext}, g) = Q + Q'$$

On peut montrer aisément[3], que la puissance potentielle des forces d'interaction est nulle

$$Q'^T \cdot \Delta\dot{q} = 0$$

pour un champ de vitesse potentielle $\Delta\dot{q}$ compatible avec les contraintes géométriques :

$$J(q) \cdot \Delta\dot{q} = 0$$

Ces deux équations permettent d'écrire l'expression des forces généralisées comme une combinaison linéaire des lignes de la matrice Jacobienne des contraintes :

$$Q' = J^T(q) \cdot \lambda$$

[3] En utilisant par exemple la troisième loi de Newton : le principe d'action-réaction

où λ représente le vecteur des coefficients de la combinaison linéaire, communément appelés *multiplicateurs de Lagrange*.

On peut écrire le système d'équations du mouvement d'un système multicorps soumis à des contraintes sous la forme suivante :

$$\begin{aligned} M(q)\ddot{q} + c(q, \dot{q}, F_{ext}, L_{ext}, g, t) &= Q + J^T(q)\,\lambda \\ h(q) &= 0 \end{aligned} \tag{2.4}$$

Il n'est pas nécessaire de recourir à un principe variationnel pour obtenir les termes de ces équations du mouvement. Ils peuvent être obtenu par l'utilisation de formalismes récursifs [13] tels que ceux présentés dans le chapitre précédent.

Autres contraintes Les équations de contraintes présentes dans un modèle ne proviennent pas nécessairement toujours de la coupure d'une boucle cinématique. Elles peuvent servir à préciser le comportement d'un composant ou d'une partie du système. On pensera par exemple aux mouvements de translation et de rotation contraints d'une roue qui roule sans glisser sur le sol ou encore aux rotations des différentes parties d'un élément de transmission tel qu'une boite de vitesse ou un différentiel.

Des contraintes simples telles que des combinaisons linéaires de coordonnées généralisées peuvent également être utilisées pour l'assemblage d'un système modélisé à l'aide de sous-sytèmes.

La possibilité d'imposer des contraintes qui dépendent explicitement du temps est offerte, dans ROBOTRAN, par le biais de la commande cinématique des articulations uniquement, ce qui est discuté à la section 2.2.4.

Les autres types de contraintes telles que des conditions de roulement sans glissement par exemple, peuvent être prisent en compte, à condition qu'elles ne dépendent pas explicitement du temps. Elles peuvent toutefois faire intervenir des coordonnées généralisées dont la valeur est imposée en fonction du temps. Les équations de ces contraintes doivent être fournies par l'intermédiaire d'une fonction externe, comme expliqué à la section 2.2.5. En effet, ROBOTRAN ne peut pas générer systématiquement les équations symboliques pour toutes les possibilités de contraintes.

2.2 Génération des équations de contraintes

Nous montrons dans cette section comment obtenir les équations de contraintes relatives à la coupure d'un corps ou d'une articulation. Il y a de nombreuses manières d'ouvrir une chaîne cinématique fermée [12, 45, 46, 47]. Nous ne présentons ici qu'un jeu de coupures simples proposées par [12], qui permettent de modéliser la majorité des systèmes rencontrés.

Toutes les équations de contraintes exprimant des conditions de fermeture de boucles sont générées symboliquement en utilisant les méthodes récursives

présentées dans la section relative à la génération des grandeurs cinématiques du chapitre précédent.

D'autres contraintes prescrites par l'utilisateur et implémentées dans des fonctions externes peuvent également être gérées symboliquement sous la condition de fournir une liste des coordonnées articulaires qui y sont impliquées, ainsi que leur Jacobienne.

2.2.1 Coupure d'un corps

Cette coupure est la plus générale et peut être appliquée en toutes circonstances pour ouvrir n'importe quelle chaîne cinématique fermée. Un corps du système réel est découpé en deux partie et modélisé par deux corps dans la représentation arborescente.

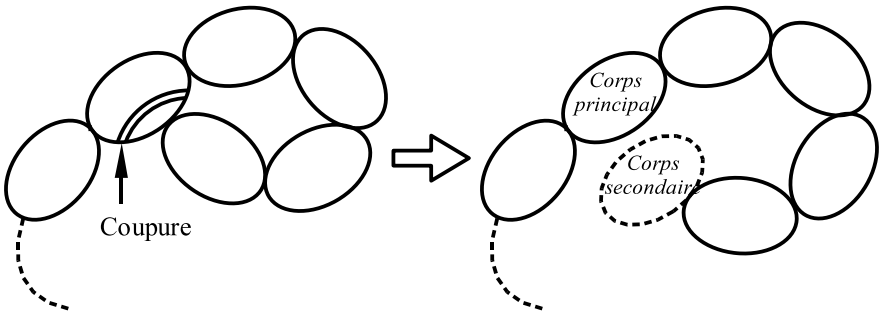


FIG. 2.2 – Coupure d'un corps

Les deux parties sont appélées *corps principal* et *corps secondaire*, comme illustré sur la figure 2.2. Le corps secondaire est généralement un corps fictif ajouté dont la masse et l'inertie sont nulles, mais il est également possible de répartir la masse et l'inertie du corps original entre les deux parties.

Les contraintes géométriques résultant de la coupure d'un corps en deux parties vont exprimer que ces deux parties sont fixées rigidement l'une à l'autre. Autrement dit, on va exprimer que les repères $\{\hat{\mathbf{X}}^p\}$ et $\{\hat{\mathbf{X}}^s\}$ respectivement solidaires des corps principal et secondaire illustrés sur la figure 2.3 sont identiques, c'est-à-dire qu'ils ont la même origine et la même orientation. Ceci résultera en l'écriture de six contraintes algébriques scalaires :

- Trois contraintes de translation imposent que les positions des points P^p et P^s respectivement situés sur le corps principal et secondaire soient identiques.
  En notant x^p et x^s les composantes respectives, dans le repère commun

choisi[4], des vecteurs positions des points P^p et P^s, on peut écrire les contraintes de translations comme ceci :

$$h_t(q) \triangleq x^s(q) - x^p(q) = 0 \tag{2.5}$$

- Trois contraintes de rotation imposent que les repères $\{\hat{\mathbf{X}}^p\}$ et $\{\hat{\mathbf{X}}^s\}$ ont la même orientation. En définissant la matrice de rotation $R(q)$ par $[\hat{\mathbf{X}}^s] \triangleq R(q)[\hat{\mathbf{X}}^p]$, on peut écrire la contrainte comme ceci :

$$R(q) = R^{sT}(q)\, R^p(q) = E \tag{2.6}$$

où E est la matrice identité. Autrement dit, on impose que les matrices de rotation $R^p(q)$ et $R^s(q)$ associées à ces deux repères soient identiques. Cette égalité matricielle est constituée de neufs contraintes scalaires. Cependant, comme toute matrice de rotation, la matrice othogonale $R(q)$, peut-être obtenue par composition de trois matrices de rotation élémentaire. En fait seulement trois contraintes parmi les neuf sont indépendantes. On montre dans [105, 13] quels éléments non diagonaux de la matrice de rotation peuvent être utilisés comme équations de contraintes de rotation.

$$h_r(q) \triangleq \begin{pmatrix} -R_{(3,2)}(q) \\ R_{(3,1)}(q) \\ -R_{(2,1)}(q) \end{pmatrix} = 0 \tag{2.7}$$

Les dérivées première et seconde des contraintes de translation sont obtenues simplement en se basant sur les vitesses et accélérations des deux points P^p et P^s :

$$\begin{aligned} \dot{h}_t(\dot{q}, q) \triangleq \dot{x}^s(\dot{q}, q) - \dot{x}^p(\dot{q}, q) &= 0 \\ \ddot{h}_t(\ddot{q}, \dot{q}, q) \triangleq \ddot{x}^s(\ddot{q}, \dot{q}, q) - \ddot{x}^p(\ddot{q}, \dot{q}, q) &= 0 \end{aligned} \tag{2.8}$$

Les expressions des vitesses cartésiennes $\dot{x}^s$ et $\dot{x}^p$ sont linéaires en fonctions des vitesses généralisées et la Jacobienne des contraintes $J_t(q)$ est obtenue simplement par différence des matrices Jacobiennes de translation associées aux vitesses $\dot{\mathbf{x}}^p$ et $\dot{\mathbf{x}}^s$.

$$J_t(q) = J_t^s(q) - J_t^p(q) \tag{2.9}$$

Cette Jacobienne $J_t(q)$ sera associée aux multiplicateurs de Lagrange λ_t pour les contraintes (2.5).

On montre dans [13], à l'aide du *Principe des Puissances Potentielles*, que les multiplicateurs de Lagrange associés aux contraintes de translation correspondent aux composantes de la résultante des forces internes de liaison, exprimées dans le repère $[\hat{\mathbf{Y}}]$ choisi pour le calcul des grandeurs cinématiques relatives aux points P^p et P^s.

[4] le repère commun peut être différent de celui du corps de base

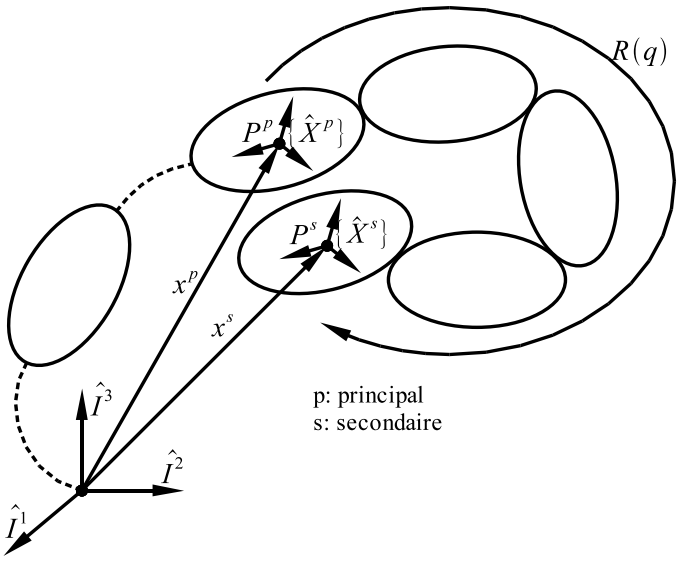


FIG. 2.3 – Condition de fermeture

$$\lambda_t = F$$

En ce qui concerne les équations de contraintes en rotation, nous utilisons la relation suivante :

$$\dot{h}_r(\dot{q}, q) \triangleq \omega^s - \omega^p = J_r(q)\dot{q} = 0$$

dans laquelle la vitesse angulaire relative du repère $\{\hat{\mathbf{X}}^s\}$ par rapport à $\{\hat{\mathbf{X}}^p\}$ est utilisée, sans toutefois être la dérivée des pseudo-contraintes d'orientation (2.7). Cette technique est justifiée dans [105, 13] et assure la convergence du processus de résolution.

Comme les vitesses angulaires $\omega^s = J_r^s(q)\dot{q}$ et $\omega^p = J_r^p(q)\dot{q}$ sont linéaires par rapport aux vitesses généralisées, on peut à nouveau obtenir la Jacobienne des contraintes $J_r(q)$ par différence des matrices Jacobiennes de rotation associées aux repères $\{\hat{\mathbf{X}}^p\}$ et $\{\hat{\mathbf{X}}^s\}$.

$$J_r(q) = J_r^s(q) - J_r^p(q) \tag{2.10}$$

Nous utilisons alors naturellement la différence des accélérations angulaires en guise de dérivée seconde des contraintes d'orientation :

$$\ddot{h}_r(\dot{q}, q) \triangleq \dot{\omega}^s - \dot{\omega}^p = 0$$

On peut également montrer que les multiplicateurs de Lagrange λ_r associés à la Jacobienne des pseudo-contraintes d'orientation sont les composantes, dans le repère choisi pour exprimer les composantes des vitesses angulaires, du couple pur de liaison qui maintient l'orientation relative constante des deux parties du corps original.

Remarque Sur base des six contraintes engendrées par la coupure d'un corps, on peut créer toute une série d'autres situations, correspondant à des coupures d'articulation, en choisissant le repère dans lequel on exprime les grandeurs cinématiques utilisées pour l'écriture des contraintes et en éliminant une ou plusieurs des équations de contraintes.

Nous ne détaillerons pas toutes les combinaisons possibles mais les deux exemples suivants illustrent ce principe.

En éliminant une des trois contraintes de translation, on donne un degré de liberté relatif au deux corps principal et secondaire dans la direction d'un axe du repère choisi pour exprimer les contraintes. La situation correspond alors au cas de la coupure d'une articulation prismatique reliant les deux corps si on choisit le repère d'un des deux corps pour exprimer les contraintes.

En éliminant les trois contraintes de rotation, on obtient une situation équivalente à la coupure d'une articulation sphérique, une rotule, reliant deux corps. Ce cas est proposé ci-après.

2.2.2 Coupure d'une articulation sphérique

Cette seconde coupure est utilisable dans le cas où deux corps de la chaîne cinématique fermée sont reliés par une rotule idéale[5]. Cette situation relativement fréquente est illustrée sur la figure 2.4.

Il est alors plus intéressant d'ouvrir la chaîne au niveau de la rotule que de couper un corps, étant donné le nombre réduit d'équations de contraintes engendrées. Celles-ci sont uniquement des contraintes de translation qui imposent une position identique aux deux points P et Q, qui correspondent au centre de la rotule et appartiennent respectivement aux deux corps i et j reliés par celle-ci.

Ainsi, en notant $x^P(q)$ et $x^Q(q)$ les composantes exprimées dans un repère commun, des vecteurs position des deux points qui coïncident avec le centre de la rotule, on peut écrire les contraintes géométriques comme ceci :

$$h(q) \triangleq x^Q(q) - x^P(q) = 0 \tag{2.11}$$

Les dérivées première et seconde sont obtenues simplement par égalité des composantes de vitesse et d'accélération des points P et Q.

On retiendra que la coupure d'une articulation sphérique

[5] sans jeu, ni couple transmis entre les deux corps

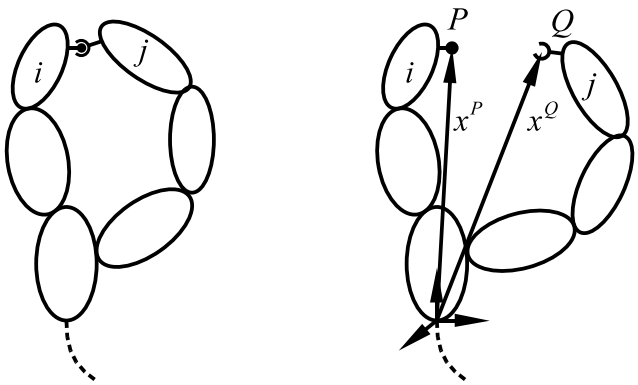


FIG. 2.4 – Coupure d'une articulation sphérique

- élimine du modèle les trois variables de rotation de l'articulation, ce qui réduit le nombre de coordonnées généralisées du modèle,
- n'engendre que trois équations de contraintes géométriques, ce qui réduit le nombre d'équations à résoudre par rapport au cas de la coupure d'un corps. De plus la convergence numérique du processus de résolution des contraintes de translation est généralement meilleure que celle des équations de rotation, basée sur une Jacobienne issue de pseudo-contraintes.

2.2.3 Coupure d'une bielle

Cette procédure de coupure est utilisée lorsque deux corps sont reliés par une bielle de connection dont la masse et l'inertie sont négligeables. Les extrémités de la bielle doivent correspondre à des articulations sphériques idéales (ou éventuellement une articulation sphérique et un joint universel).

Dans ce cas, la bielle a simplement pour effet de maintenir une distance l constante entre deux points P et Q de la structure, comme illustré sur la figure 2.5).

La bielle est le siège de contraintes internes de traction ou de compression dont la résultante est une force de liaison qui est appliquée aux corps i et j, dans la direction de la bielle. Les effets d'inertie de la bielle ou de frottement dans les articulations sphériques sont négligés.

L'équation de contrainte est unique et provient de la condition géométrique imposée par la bielle. La distance séparant les points P et Q est calculée sur base des positions relatives des deux points. Le vecteur $\overrightarrow{PQ}$ est définit par la relation $\mathbf{x}(q) \triangleq \mathbf{x}^Q(q) - \mathbf{x}^P(q)$. La condition géométrique fixe cette distance à

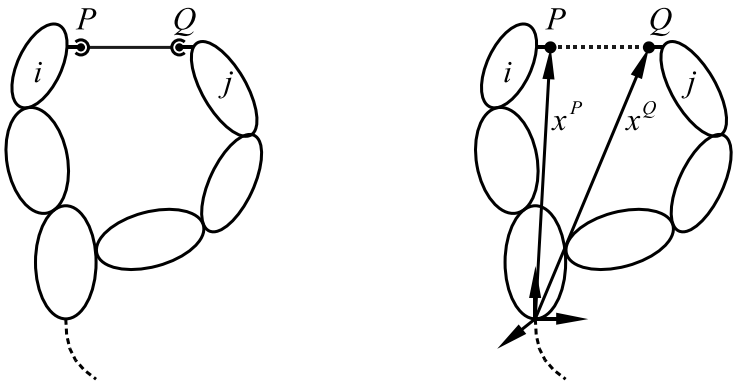


FIG. 2.5 – Coupure d'une bielle

la valeur l :

$$\| \mathbf{x}(q) \| = l$$

En pratique, pour obtenir une expression plus compacte de sa dérivée, l'équation de la contrainte est exprimée comme ceci :

$$h(q) \triangleq \frac{1}{2} \| \mathbf{x}(q) \|^2 - \frac{l^2}{2} = \frac{1}{2} \mathbf{x}(q) \cdot \mathbf{x}(q) - \frac{l^2}{2} = 0 \tag{2.12}$$

Cela évite l'extraction de racine carrée et l'expression de la dérivée première est simplement :

$$\dot{h}(\dot{q}, q) = \mathbf{x}(q) \cdot \dot{\mathbf{x}}(\dot{q}, q) = 0 \tag{2.13}$$

Cette expression peut-être récrite en tenant compte de la dépendance linéaire des expressions des vitesses des points P et Q en fonctions des vitesses généralisées :

$$\dot{h}(\dot{q}, q) = \mathbf{x}(q) \cdot \frac{\partial \dot{\mathbf{x}}(q)}{\partial \dot{q}^T} \dot{q} \tag{2.14}$$

D'où on peut tirer facilement l'expression de la Jacobienne de la contrainte :

$$J(q) = \mathbf{x}(q) . \frac{\partial \dot{\mathbf{x}}(q)}{\partial \dot{q}^T} \tag{2.15}$$

L'expression de la dérivée seconde de la contrainte, dans laquelle apparaissent les accélérations des points P et Q, est obtenue de façon semblable :

$$\ddot{h}(\ddot{q}, \dot{q}, q) = \dot{\mathbf{x}}(\dot{q}, q) \cdot \dot{\mathbf{x}}(\dot{q}, q) + \mathbf{x}(q) \cdot \ddot{\mathbf{x}}(\ddot{q}, \dot{q}, q) \tag{2.16}$$

On montre dans [13] comment obtenir la relation entre le multiplicateur de Lagrange λ de la contrainte et la force de liaison F :

$$\lambda = \frac{F}{\| \mathbf{x} \|} = \frac{1}{l} F \tag{2.17}$$

On constate donc que λ représente une force de liaison par unité de longueur $[Newton/m]$

On retiendra que la coupure d'une bielle

- élimine les variables de rotations des articulations situées aux extrémités de la bielle, ce qui réduit le nombre de coordonnées généralisées dans le modèle
- n'engendre qu'une seule équation de contrainte géométrique,
- doit être utilisée avec précaution, dans la mesure où elle consiste à supprimer un corps et deux articulations du système réel.

2.2.4 Articulations commandées

Le cas des articulations dont la cinématique est imposée a priori constitue un cas particulier de contrainte. L'évolution temporelle de la coordonnée généralisée est dictée par une fonction $f(t)$ décrivant le mouvement imposé. Dans le cas où cette fonction ne dépend pas des autres coordonnées généralisées, l'équation de contrainte associée à l'articulation j dépend cette fois explicitement du temps et s'écrit simplement :

$$h(q,t) \triangleq q^j - f(t) = 0 \tag{2.18}$$

Les expressions des dérivées d'une telle contrainte peuvent être déduites aisément. La ligne de la Jacobienne associée ne contiendra qu'un seul élément non nul à la colonne j correspondant au coefficient de la variable q_j dans l'équation de la contrainte.

La résolution de ces équations de contraintes est triviale. En pratique, les valeurs des coordonnées associées aux articulations commandées peuvent être déterminées directement et indépendamment de toutes les autres coordonnées. Nous ne les considérons donc pas comme des coordonnées *dépendantes*, mais comme des coordonnées *indépendantes* pour la résolution des contraintes scléronomes présentée dans la section 2.4.

2.2.5 Autres contraintes imposées par l'utilisateur

Les trois cas de coupure présentés ci-dessus ainsi que la commande cinématique de certaines articulations ne suffisent pas toujours pour modéliser le comportement d'un système ou de certains éléments.

Un exemple simple est le cas de la présence d'une articulation hélicoïdale (ex : une vis) entre deux corps. Une telle articulation ne permet qu'un seul degré

de liberté relatif mais présente deux mouvements combinés : une translation et une rotation selon le même axe. Elle peut être modélisée par deux articulations prismatique et rotoïde élémentaires dont les coordonnées articulaires respectives $z(t)$ et $\theta(t)$ doivent être soumises à une relation algébrique linéaire du type $k.\theta(t) - z(t) = 0$ où la constante k représente le pas de la vis. Dans ce cas, la contrainte prend la forme d'une combinaison linéaire de coordonnées et peut être générée symboliquement par ROBOTRAN.

Tous les exemples ne sont pas aussi simples et il est parfois utile voire nécessaire d'imposer des contraintes très particulières dont la forme sera généralement non linéaire. Les expressions de ces contraintes ne seront alors pas calculées symboliquement par ROBOTRAN mais bien implémentées par l'utilisateur dans une fonction externe $fctext(q)$. Nous imposons toutefois que ces contraintes particulières soient scléronomes, comme dans le cas des coupures.

L'équation suivante sera générée par ROBOTRAN

$$hJ_{usr} = fctext(q)$$

où le tableau hJ_{usr} renvoyé par la fonction $fctext(q)$ contient à la fois la valeur de la contrainte et la valeur des termes non nuls de la ligne correspondante de la Jacobienne.

L'utilisateur devra également implémenter une fonction $d2fctext(q, \dot{q})$ séparée pour le calcul des termes $\dot{J}\dot{q}$ qui interviennent dans la dérivée seconde de sa contrainte.

$$Jdqd_{usr} = d2fctext(q, \dot{q})$$

Afin de permettre un traitement symbolique optimal des contraintes et de leur résolution, l'utilisateur doit fournir la liste de toutes les coordonnées articulaires impliquées dans sa contrainte de telle sorte que ROBOTRAN puisse déterminer quels sont, a priori, les termes nuls et non nuls de la ligne correspondante de la Jacobienne.

2.3 Partitionnement des coordonnées

Comme nous l'avons déjà dit, un système articulé soumis à des contraintes possède un nombre de degrés de liberté inférieur au nombre d'articulations élémentaires présentes dans sa structure.

En appelant

- n, le nombre d'articulations élémentaires de la structure arborescente après coupure des boucles,
- m, le nombre d'équations de contraintes scléronomes scalaires indépendantes,
- c, le nombre d'articulations commandées dont la cinématique est imposée (voir 2.2.4) et donc connue a priori,

on peut déterminer le nombre f de degrés de liberté du système par la relation suivante :

$$f = n - m - c \tag{2.19}$$

Nous nous intéressons dans ce travail au cas où ce nombre de degrés de liberté f est positif. Dans le cas particulier où $f = 0$, le système peut encore avoir une mobilité si certaines articulations sont commandées, c'est-à-dire si $c > 0$, et faire l'objet d'un analyse cinématique. Si $f = c = 0$, le système est totalement contraint et sa structure est alors statique, toutes les coordonnées généralisées q ont une valeur constante et donc $\ddot{q} = \dot{q} = 0$.

La résolution des m contraintes scléronomes permet alors de déterminer la valeur de m coordonnées généralisées, que l'on note q_v et que l'on appelle coordonnées *dépendantes*, en fonction des $n - m$ autres coordonnées *indépendantes* notées q_u.

Cette technique de *partitionnement des coordonnées* a été proposée par Wehage et Haug [37] en 1982 et est largement utilisée.

La matrice Jacobienne des contraintes joue un rôle essentiel dans la résolution des équations de contraintes, et en particulier la sous-matrice carrée construite avec les colonnes qui correspondent aux coordonnées généralisées dépendantes q_v. On la note J_v.

$$J_v(q) \triangleq \frac{\partial h(q)}{\partial {q_v}^T}$$

Il est indispensable que la matrice J_v soit régulière et bien conditionnée. Pour cela, il faut que toutes les contraintes soient indépendantes et que le jeu des m coordonnées q_v soit choisi de telle sorte que J_v soit de plein rang.

L'ensemble des $n - m$ coordonnées indépendantes q_u est alors composé de :

- c coordonnées commandées q_c dont les évolutions temporelles sont imposées et qui sont appelées coordonnées *commandées*,
- f coordonnées q_f qui correspondent aux degrés de liberté du système et sont appelées coordonnées *libres*.

2.3.1 Approche numérique

Habituellement, dans le contexte de la génération symbolique, le choix des variables *indépendantes* et *dépendantes* est laissé au soin de l'utilisateur. La procédure classique consiste à générer tout d'abord les équations de contraintes et à procéder à une analyse numérique préliminaire de la matrice Jacobienne.

Outre les coordonnées associées aux articulations commandées, on placera nécessairement dans la partition des variables indépendantes les coordonnées qui n'interviennent pas dans les contraintes, car les colonnes correspondantes de la Jacobienne sont nulles.

La sous-matrice constituée des colonnes de J correspondant aux autres coordonnées, doit ensuite faire l'objet de l'analyse numérique pour en dégager la matrice J_v et ainsi les m coordonnées dépendantes.

Une telle analyse peut consister en une recherche successive de m pivots de valeur absolue maximale. Cette recherche est systématique lors de l'utilisation d'une méthode numérique de factorisation LU avec pivotement total ou même partiel. Ces m pivots appartiennent à autant de colonnes qui seront utilisées pour définir la matrice J_v.

Cette méthode directe et rapide fournit systématiquement une et une seule partition.

Toutefois, il existe bien souvent plusieurs partitions valides du point de vue du conditionnement numérique de J_v.

Il faut savoir que la partition des coordonnées a une influence importante sur la quantité et l'efficacité des équations générées pour la modélisation d'un système contenant des boucles cinématiques. En effet, comme nous le montrons dans les paragraphes suivants, le choix des coordonnées dépendantes a une influence sur la structure de J_v et sur le découplage des contraintes, ce qui peut entraîner une importante réduction des calculs nécessaires à leur résolution.

2.3.2 Structure de J_v et découplage des contraintes

Considérons le mécanisme représenté sur la figure 2.6. Ce mécanisme plan formé des sept corps assemblés de manière à former trois boucles cinématiques ne possède qu'un seul degré de liberté.

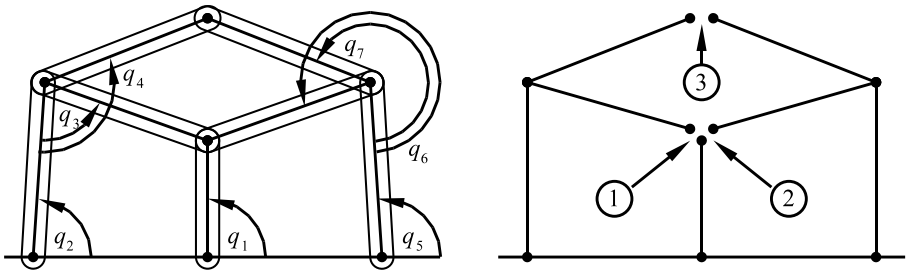


FIG. 2.6 – Mécanisme plan à un degré de liberté

Les trois boucles sont ouvertes par des coupures d'articulations comme indiqué sur la partie droite de la figure 2.6. Deux contraintes de translation sont générées pour chaque coupure, afin d'assurer la coïncidence de deux points dans le plan. Il reste sept articulations dans le système arborescent.

La Jacobienne des contraintes a la structure suivante où les $\otimes$ représentent des termes non nuls :

$$q = \begin{pmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 \end{pmatrix}$$

$$J = \left(\begin{array}{ccccccc} \otimes & \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \otimes & \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \hline \otimes & \cdot & \cdot & \cdot & \otimes & \otimes & \cdot \\ \otimes & \cdot & \cdot & \cdot & \otimes & \otimes & \cdot \\ \hline \cdot & \otimes & \cdot & \otimes & \otimes & \cdot & \otimes \\ \cdot & \otimes & \cdot & \otimes & \otimes & \cdot & \otimes \end{array}\right) = \left(\begin{array}{c} \partial h_{1x}/\partial q \\ \partial h_{1y}/\partial q \\ \hline \partial h_{2x}/\partial q \\ \partial h_{2y}/\partial q \\ \hline \partial h_{3x}/\partial q \\ \partial h_{3y}/\partial q \end{array}\right)$$

Les lignes horizontales mettent en évidence les trois paires de lignes de la matrice J qui correspondent aux trois coupures.

Nous allons utiliser ce petit mécanisme pour illustrer l'influence du choix de la partition sur la structure de la Jacobienne J_v et le découplage de la résolution des contraintes.

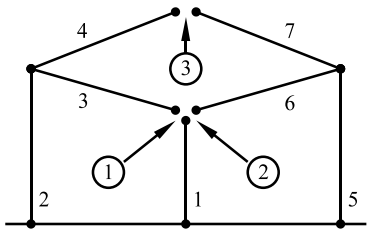
Nous considérons trois partitions différentes, en considérant successivement les coordonnées q_1, q_2 et q_4 comme variables indépendantes. Les coordonnées dépendantes sont réordonnées pour assurer que les termes de la diagonale ne soient pas nuls. La raison de cette permutation des colonnes de J_v est donnée dans la section suivante.

Partition 1 :

Choisissons la coordonnée q_1 comme indépendante.
$q_u = \{q_1\}$

$$q_v = \begin{pmatrix} q_3 & q_2 & q_5 & q_6 & q_4 & q_7 \end{pmatrix}$$

$$J_v = \left(\begin{array}{cc|cc|cc} \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \hline \cdot & \cdot & \otimes & \otimes & \cdot & \cdot \\ \cdot & \cdot & \otimes & \otimes & \cdot & \cdot \\ \hline \cdot & \otimes & \otimes & \cdot & \otimes & \otimes \\ \cdot & \otimes & \otimes & \cdot & \otimes & \otimes \end{array}\right)$$



Avec cette première partition, on constate que les deux variables dépendantes qui interviennent dans les deux premières contraintes, relatives à la première coupure, n'interviennent pas dans les deux contraintes de la seconde coupure et vice versa. Ces deux paires de contraintes peuvent être résolues séparément, indépendamment l'une de l'autre.

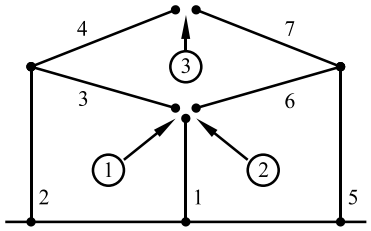
D'un point de vue mécanique, cela signifie que les deux parties inférieures gauche et droite du mécanisme peuvent être assemblées séparément sur le corps central, si celui-ci est fixe. Une fois résolues les contraintes relatives aux deux premières coupures, la troisième boucle peut être refermée à son tour.

Partition 2 :

Nous choisissons maintenant le coordonnée q_2 comme indépendante.
$q_u = \{q_2\}$

$$q_v = \{\ q_3 \quad q_1 \quad q_5 \quad q_6 \quad q_4 \quad q_7\ \}$$

$$J_v = \left(\begin{array}{cc|cc|cc} \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \hline \cdot & \otimes & \otimes & \otimes & \cdot & \cdot \\ \cdot & \otimes & \otimes & \otimes & \cdot & \cdot \\ \hline \cdot & \cdot & \otimes & \cdot & \otimes & \otimes \\ \cdot & \cdot & \otimes & \cdot & \otimes & \otimes \end{array}\right)$$



Avec cette seconde partition, on constate qu'il n'y a toujours que deux variables qui interviennent dans les deux équations de contraintes relatives à la première coupure. Ces équations peuvent donc être résolues indépendamment des autres équations de contraintes. En revanche, comme la variable q_1, qui correspond à la seconde colonne, intervient dans les équations relatives à la seconde coupure, celles-ci ne peuvent être résolues qu'après fermeture de la première boucle. Il en est de même pour la variable q_5 et les équations relatives à la troisième coupure.

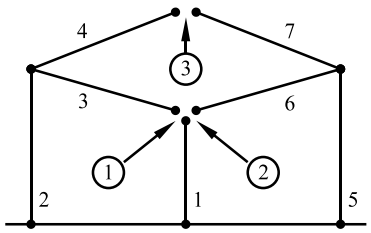
Cette possibilité de résoudre les boucles cinématiques en cascade ou même en parallèle, est exploitée dans [45, 46] et [47]. Elle est à la base de l'utilisation des "Kinematic Transformers" pour la représentation des systèmes composés de nombreuses boucles. Cette propriété est utilisable quelle que soit la méthode, analytique ou numérique, choisie pour résoudre les équations de contraintes.

Partition 3 :

Examinons cette fois, la conséquence du choix de q_4 comme coordonnée indépendante. $q_u = \{q_4\}$

$$q_v = \{\ q_3 \quad q_1 \quad q_6 \quad q_5 \quad q_2 \quad q_7\ \}$$

$$J_v = \left(\begin{array}{cc|cc|cc} \otimes & \otimes & \cdot & \cdot & \otimes & \cdot \\ \otimes & \otimes & \cdot & \cdot & \otimes & \cdot \\ \hline \cdot & \otimes & \otimes & \otimes & \cdot & \cdot \\ \cdot & \otimes & \otimes & \otimes & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot & \otimes & \otimes & \otimes \\ \cdot & \cdot & \cdot & \otimes & \otimes & \otimes \end{array}\right)$$



Cette troisième partition ne permet pas de résoudre certaines équations indépendamment des autres, à cause de la présence de termes non nuls dans les blocs situés hors de la diagonale de la matrice. Ces termes correspondent

à un couplage des équations relatives aux trois coupures. L'ensemble complet des équations de contraintes doit donc être résolu en un seul bloc.

Remarque Cette situation se présente souvent lorsque les variables dépendantes correspondent à des articulations situées dans des branches communes à différentes boucles cinématiques. C'est le cas des variables q_1, q_2 et q_5 dans notre exemple.

Cette observation devrait même être prise en considération lors de la modélisation du système. Il est en effet souvent possible d'utiliser des chaînes cinématiques fictives pour introduire des coordonnées articulaires indépendantes correspondant aux coordonnées absolues ou indirectes de certains corps du système, et permettre de découpler la résolution des boucles cinématiques.

Cette technique sera illustrée à la section 2.7 pour la modélisation efficace de robots ou de mécanismes à structure parallèle.

2.3.3 Factorisation bloc triangulaire de J_v

Nous avons vu dans la section précédente que pour certaines partitions des coordonnées, la matrice carrée J_v pouvait présenter une structure intéressante pour le découplage de la résolution des contraintes. Cette structure en bloc n'est toutefois pas connue a priori. Il est nécessaire de la rechercher pour savoir si elle existe, pour une partition donnée.

Afin de mettre cette structure en évidence, nous avons implémenté dans ROBOTRAN une méthode de *factorisation bloc triangulaire* de la matrice J_v. Une méthode plus générale applicable aux matrices rectangulaires est présentée en détails dans [106].

L'environnement MATLAB offre également une fonction `dmperm` qui permet d'effectuer une telle factorisation sur une matrice rectangulaire.

La première étape consiste en la recherche d'un "maximum matching" [106].

Cela consiste à faire correspondre chaque ligne de la matrice avec une de ses colonnes. On utilise pour cela une représentation de la matrice J_v sous forme de graphe bipartite, comme illustré sur la figure 2.7 dont les noeuds, illustrés par des (•), représentent les lignes et les colonnes, alors que les arcs (–) qui relient les noeuds correspondent aux éléments. Les paires de noeuds sont formées en commençant par associer entre eux ceux qui sont le moins connectés, c'est-à-dire en associant d'abord les lignes et colonnes possédant le moins d'éléments.

Lorsque la matrice est carrée et régulière, la procédure se termine lorsque chaque ligne est associée à une colonne. Il va sans dire que chaque ligne ne peut être associée qu'une seule fois avec une colonne et vice versa.

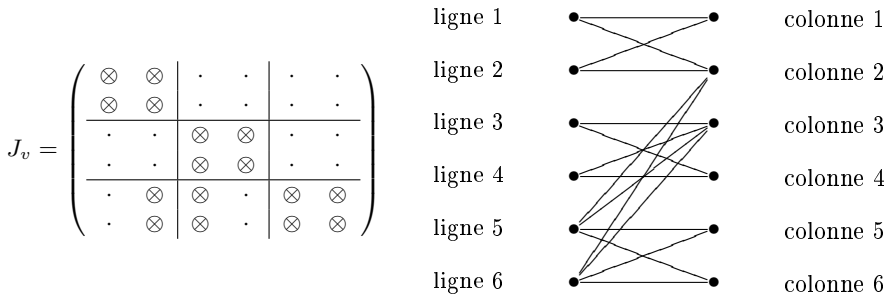


FIG. 2.7 – Graphe bipartite d'une matrice

Factorisation

La mise en évidence de la structure bloc triangulaire, nécessite de poursuivre l'analyse du graphe bipartite de la matrice J_v, pour trouver des chemins cycliques. On ne retient que les cycles qui relient des noeuds de même indice. Les arcs qui relient ces noeuds de même indice correspondent à des éléments diagonaux. L'ensemble des arcs appartenant à des cycles disjoints correspondent alors aux éléments qui formeront les différents blocs diagonaux de la structure bloc triangulaire.

Lorsque tous les éléments diagonaux ont été associés dans des blocs, il ne reste plus qu'à permuter les lignes et les colonnes de la matrice de manière à regrouper les éléments qui appartiennent à chaque bloc et à ranger les éléments hors blocs diagonaux du même côté de la diagonale de la matrice. On obtient ainsi la structure bloc triangulaire.

L'algorithme complet est décrit en détail dans [106].

Remarques Cette méthode de factorisation n'utilise pas la valeur numérique des éléments. Elle peut donc être parfaitement appliquée dans le contexte de la génération symbolique et a été implémentée dans le logiciel ROBOTRAN afin de profiter au maximum des avantages d'une telle structure pour la matrice J_v.

Le caractère triangulaire de la structure en bloc de J_v qu'il est possible de mettre en évidence dépend du partitionnement des coordonnées.

Pour chaque bloc diagonal, il est toujours possible de permuter les lignes et les colonnes sans altérer la structure globale de la matrice. De telles permutations peuvent viser à placer les meilleurs pivots de ces blocs sur la diagonale, de manière à optimiser le comportement numérique de la méthode de résolution des contraintes.

Actuellement, les permutations des lignes et des colonnes de J_v sont effectuées symboliquement par ROBOTRAN pour effectuer la factorisation bloc

triangulaire. Ces permutations sont déterminées une fois pour toutes au moment de la génération du modèle.

On peut néanmoins envisager de laisser à l'utilisateur la possibilité de spécifier ces permutations et de désactiver la recherche automatique d'une factorisation bloc triangulaire par ROBOTRAN. Seule la vérification de la présence d'éléments symboliques non nuls sur la diagonale sera maintenue pour des raisons évidentes.

2.3.4 Méthode de partitionnement symbolique

La partitionnement des coordonnées par une approche numérique ne garantit pas que la matrice J_v correspondante possède une structure bloc triangulaire intéressante. Par intéressante, nous entendons que les blocs diagonaux soient petits, et que les éléments non nuls situés hors de ces blocs soient peu nombreux. La structure la plus intéressante est évidemment celle d'une matrice purement diagonale.

Au lieu d'utiliser une procédure numérique, on pourrait envisager d'appliquer une méthode de factorisation bloc triangulaire à la matrice Jacobienne complète et sélectionner les colonnes de la partie triangulaire pour définir la partition des coordonnées. Cependant, des tests effectués à l'aide de la fonction `dmperm` disponible dans l'environnement MATLAB montrent que cette méthode ne permet pas systématiquement de mettre en évidence une structure bloc triangulaire intéressante.

Ceci s'explique par l'existence d'une multitude de solutions au problème du "maximum matching", c'est-à-dire de la recherche des m paires de lignes et de colonnes de J ayant un élément non nul en commun. Or c'est précisément cette étape de la méthode qui conditionne la décomposition de la matrice et donc la partition obtenue.

Nous avons développé une procédure symbolique pour déterminer une partition qui conduit à une structure bloc triangulaire intéressante pour J_v. La différence entre la procédure décrite dans [106] et celle que nous présentons ici est que nous tenons compte du nombre d'éléments présents dans les colonnes de J, pour la mise en correspondance de lignes et des colonnes.

Algorithme

- On définit pour chaque colonne j,
  - N^j, le nombre total d'éléments de cette colonne, soit le nombre total de lignes i qui ont un élément en commun avec cette colonne j
  - L_i^j le nombre de lignes dont l'indice est supérieur ou égal à i et qui ont un élément en commun avec cette colonne j,
- On définit pour chaque élément E_{ij} appartenant à la ligne i et à la colonne j, un nombre de connexions S_{ij}. Il s'agit du nombre d'éléments non nuls

E_{kj} de la matrice J, tels que $E_{kl} \neq 0$ et $E_{il} \neq 0$ avec $i \neq k$ et $j \neq l$.

On va associer à chaque ligne i, de la première à la dernière, une colonne j telle que

- l'articulation j n'est pas commandée $q_j \notin \{q_c\}$,
- l'élément E_{ij} est non nul,
- $N^j \leq N^k$ avec $L_i^j > 0$ et $S_{ij} \leq S_{ik}$. Si $S_{ij} = S_{ik}$ on choisit $j : j < k$.

L'ensemble des indices j des colonnes affectées successivement à chaque ligne i constitue l'ensemble des indices des coordonnées dépendantes.

Remarques La méthode proposée se révèle être efficace dans la plupart des cas. Un exemple d'application fructueuse de cette méthode est présenté à la section 2.8.2, où elle a permis de déterminer une excellente partition, en ce qui concerne le nombre d'équations du modèle généré.

Toutefois, cette méthode ne se base que sur des considérations de nature symbolique. Aucun test numérique n'est effectué. Elle n'offre donc a priori aucune garantie que la matrice J_v obtenue est bien conditionnée.

Il est toujours indispensable de procéder à une analyse numérique pour valider la partition engendrée. Le conditionnement de la matrice J_v est un facteur sensible pour la bonne convergence de la méthode de Newton-Raphson utilisée pour la résolution des contraintes, et constituera donc toujours un élément clé dans l'appréciation d'une partition.

2.3.5 Synthèse

Nous résumons ici les différents éléments relatifs au partitionnement des coordonnées préalable à la génération symbolique du modèle dynamique complet.

Nous distinguons :

1. le partitionnement des coordonnées, qui définit l'ensemble des colonnes de la Jacobienne qui constituent la sous-matrice J_v ainsi que l'ensemble des coordonnées dépendantes,
2. les permutations adéquates des lignes et des colonnes de J_v qui garantissent le bon comportement numérique de la méthode itérative de Newton-Raphson utilisée pour la résolution des contraintes,
3. les permutations des lignes et des colonnes de J_v qui permettent de mettre en évidence une structure bloc triangulaire favorable à la réduction du nombre d'opérations du modèle.

En ce qui concerne le partitionnement des coordonnées, plusieurs approches sont envisageables :

1. on peut utiliser une méthode de partitionnement numérique par pivotement de la Jacobienne. La recherche de m pivots de valeurs maximales définit à la fois la partition et les permutations de lignes et des colonnes de J_v, en fonction de la séquence de découverte des pivots,

2. nous proposons une méthode symbolique qui fournit la partition des coordonnées en ne se basant que sur un examen de la structure de la Jacobienne,
3. pour des systèmes simples, on peut effectuer la partition "manuellement" en se basant sur l'expérience acquise.

Les permutations des lignes et des colonnes peuvent également être déterminées de plusieurs façons différentes :

1. en conservant celles qui sont déterminées par la méthode de partitionnement par pivotement appliquée à la Jacobienne complète,
2. en appliquant une technique de pivotement à la sous-matrice J_v, lorsqu'on a utilisé une méthode de partitionnement symbolique ou manuelle,
3. en cherchant à mettre en évidence une structure bloc triangulaire pour J_v.

Dans ce dernier cas, il est encore possible d'effectuer des permutations des lignes et des colonnes au sein des blocs diagonaux, tout en conservant la structure bloc triangulaire.

Il est vivement conseillé de procéder à un pivotement des sous-matrices qui correspondent aux blocs diagonaux pour déterminer les permutations optimales des lignes et de colonnes de J_v.

Ces permutations des lignes et des colonnes peuvent être spécifiées à ROBOTRAN et seront alors figées dans le modèle généré une fois pour toutes.

Nous recommandons l'utilisation de la procédure de partitionnement symbolique et la factorisation bloc triangulaire de J_v suivie du pivotement des blocs diagonaux.

Cette solution permet de combiner les avantages

- de la structure intéressante de J_v et
- d'un bon comportement numérique de la méthode de résolution,

pour la réduction du nombre d'opérations à effectuer lors de l'utilisation du modèle du système.

2.4 Résolution des équations de contraintes

La plupart des m équations de contraintes proviennent de l'ouverture de chaînes cinématiques fermées et sont généralement non linéaires par rapport aux variables articulaires. Une des conséquence de cette non linéarité est qu'il n'existe pas toujours de solution analytique à ces équations. Dans ces cas là, leur résolution implique l'utilisation d'une méthode numérique itérative, telle que la méthode de Newton-Raphson.

2.4.1 Calcul des coordonnées généralisées dépendantes

La méthode de résolution présentée ici est en fait une implémentation symbolique de la méthode de Newton-Raphson. Cette méthode ne fournit pas une solution directe analytique des équations de contraintes (2.1), mais son utilisation permet de trouver la valeur numérique de la solution. Elle nécessite une bonne estimation initiale de la valeur des coordonnées dépendantes. Leurs valeurs précises sont alors calculées de façon itérative par la formule suivante :

$$q_v^{k+1} = q_v^k - \left(\frac{\partial h(q)}{\partial {q_v}^T} \right)^{-1} \cdot h(q) \tag{2.20}$$

Nous proposons de générer symboliquement tous les éléments nécessaires à l'exécution itérative de l'équation (2.20). Les contraintes et la Jacobienne provenant des coupures des boucles cinématiques sont générées en utilisant les formalismes présentés dans la section 1.2. Les équations nécessaires à l'évaluation du terme $J_v(q)^{-1} \cdot h(q)$ sont également générées symboliquement.

En définissant le terme Δq_v comme ceci :

$$\Delta q_v \stackrel{\Delta}{=} -J_v(q)^{-1} \cdot h(q) \tag{2.21}$$

L'équation itérative (2.20) est alors récrite :

$$q_v^{k+1} = q_v^k + \Delta q_v \tag{2.22}$$

Le contrôle de la convergence du processus itératif de résolution est basé sur l'évaluation de la norme des contraintes $\|h\|$. En pratique nous utilisons le carré de la norme dont l'expression est générée symboliquement en effectuant le produit scalaire $h^T\, h$.

Calcul de Δq_v

Nous développons dans ce paragraphe le calcul symbolique du terme Δq_v qui apparaît dans l'équation (2.22). Nous n'utilisons pas la définition (2.21) de ce terme pour son calcul. Nous évitons ainsi de calculer l'inverse de la matrice Jacobienne J_v. Pour obtenir Δq_v, nous résolvons le système linéaire suivant :

$$J_v(q)\ \Delta q_v = -h(q) \tag{2.23}$$

La résolution de ce système d'équations (2.23) est basée sur une factorisation LU de la matrice J_v. L'algorithme de la factorisation LU qui a une complexité $O(N^3/3)$ [18], est repris ci-après. Il permet de calculer les matrices L et U correspondant à une matrice A non symétrique, telles que $A = LU$. Les éléments a_{ij} sont écrasés par l_{ij} si $i > j$ ou par u_{ij} sinon. La méthode tombe en défaut si le terme a_{kk} s'avère être nul.

For $k = 1 : n$
  For $i = k + 1 : n$
    $a_{ik} = a_{ik}/a_{kk}$
    For $j = k + 1 : n$
      $a_{ij} = a_{ij} - a_{kj}\, a_{ik}$
    end
  end
end

L'utilisation de cet algorithme nécessite que tous les termes a_{kk} situés sur la diagonale de la matrice J_v soient différents de zéro.

$$h = \begin{pmatrix} h^I(q_u, q_v^I) \\ h^{II}(q_u, q_v^I, q_v^{II}) \\ h^{III}(q_u, q_v^I, q_v^{II}, q_v^{III}) \\ \end{pmatrix} \qquad J_v = \begin{pmatrix} J^I & \overset{1}{\searrow} & \cdot & \cdot \\ \otimes & J^{II} & \overset{2}{\searrow} & \cdot \\ \otimes & \otimes & J^{III} & \overset{3}{\searrow} \\ \otimes & \otimes & \otimes & \end{pmatrix}$$

Chaque ligne de la matrice J_v correspond à une équation de contrainte h et chaque colonne à une coordonnée dépendante q_v.

On note J^I, J^{II}, etc. les différents blocs diagonaux de la Jacobienne. Dès lors, h^I, h^{II}, etc. sont les équations de contraintes correspondant aux blocs I, II, etc. De façon similaire, q_v^I, q_v^{II} etc. sont les ensembles des coordonnées dépendantes correspondant aux différents blocs.

La structure bloc triangulaire de la matrice J_v, permet de résoudre les équations de contraintes $h(q) = 0$ en n'utilisant que les éléments de J_v situés dans les blocs diagonaux.

$$J_v^{diag} = \begin{pmatrix} J^I & \cdot & \cdot & \cdot \\ \cdot & J^{II} & \cdot & \cdot \\ \cdot & \cdot & J^{III} & \cdot \\ \cdot & \cdot & \cdot & \end{pmatrix}$$

En effet, il est possible de résoudre les groupes de contraintes h^I, h^{II}, etc. correspondant aux blocs J^I, J^{II}, etc. de la Jacobienne, de façon récursive, bloc par bloc.

La résolution itérative des équations de contraintes h^I va fournir les valeurs des coordonnées q^I. Ces coordonnées sont alors considérées comme indépendantes pour la résolution des équations de contraintes qui se rapportent aux blocs suivants.

On peut alors procéder à la résolution itérative des équations de contraintes h^{II} afin d'obtenir les valeurs des coordonnées q^{II} et ainsi de suite jusqu'à ce que toutes les équations de contraintes soient résolues et toutes les valeurs des coordonnées dépendantes soient connues.

La conséquence intéressante de cette observation est que seuls les termes situés dans les blocs diagonaux de J_v sont nécessaires pour le calcul des Δq_v. On peut résoudre le système d'équations (2.23) bloc par bloc et générer les expressions symboliques des $\Delta q_v^I, \Delta q_v^{II}, \cdots$ en résolvant symboliquement des systèmes d'équations plus petits pour chaque bloc de la matrice, ce qui constitue une économie de calculs évidente.

$$\begin{aligned} & J^I(q_u, q_v^I) \cdot \Delta q_v^I = -h^I(q_u, q_v^I) \\ \hookrightarrow \quad & J^{II}(q_u, q_v^I, q_v^{II}) \cdot \Delta q_v^{II} = -h^{II}(q_u, q_v^I, q_v^{II}) \\ \hookrightarrow \quad & J^{III}(q_u, q_v^I, q_v^{II}, q_v^{III}) \cdot \Delta q_v^{III} = -h^{III}(q_u, q_v^I, q_v^{II}, q_v^{III}) \\ & \vdots \end{aligned}$$

Ce principe de résolution récursive concerne l'évaluation numérique des équations de résolution générées symboliquement. La structure proposée pour l'implémentation de la procédure numérique de résolution itérative des contraintes est illustrée sur la figure 2.8.

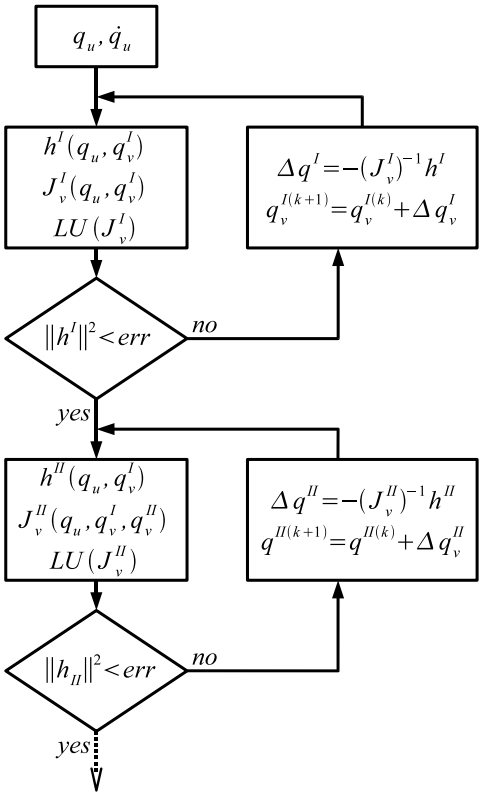


FIG. 2.8 – Organigramme de résolution des équations de contraintes

Génération symbolique

On remarquera toutefois que pour pouvoir en effectuer l'analyse et la factorisation, la matrice J_v a du être préalablement générée symboliquement. La séquence de génération symbolique des équations de contrainte ainsi que des termes de J_v ne correspond donc pas a priori à la séquence d'évaluation déterminée a posteriori.

Afin de pouvoir identifier et trier les expressions relatives aux différents blocs, les équations de contraintes, les termes de la Jacobienne et les autres éléments utilisés dans la résolution des contraintes sont marqués par un code d'identification. Ceci est indispensable pour pouvoir imprimer les équations dans une séquence correcte et insérer les instructions de contrôle d'exécution appropriées du langage choisi pour exporter les équations.

2.4.2 Calcul des vitesses généralisées dépendantes

La résolution des dérivées premières des équations de contraintes consiste en la résolution d'un système d'équations linéaires dont les inconnues sont les vitesses généralisées dépendantes $\dot{q}_v$.

Les valeurs des coordonnées dépendantes doivent être connues avant de pouvoir évaluer les expressions générées pour le calcul des vitesses.

L'équation (2.2) peut être récrite comme ceci en considérant le partitionnement $\{q_u; q_v\}$ des coordonnées :

$$J(q)\ \dot{q} = J_u(q)\ \dot{q}_u + J_v(q)\ \dot{q}_v = 0 \tag{2.24}$$

J_u est la matrice constituée des $n-m$ colonnes de la matrice Jacobienne des contraintes correspondant aux $n-m$ coordonnées indépendantes q_u.

Le système linéaire à résoudre est alors le suivant :

$$J_v(q)\ \dot{q}_v = -J_u(q)\ \dot{q}_u \tag{2.25}$$

Et les valeurs des vitesses dépendantes sont obtenues par l'évaluation de l'expression

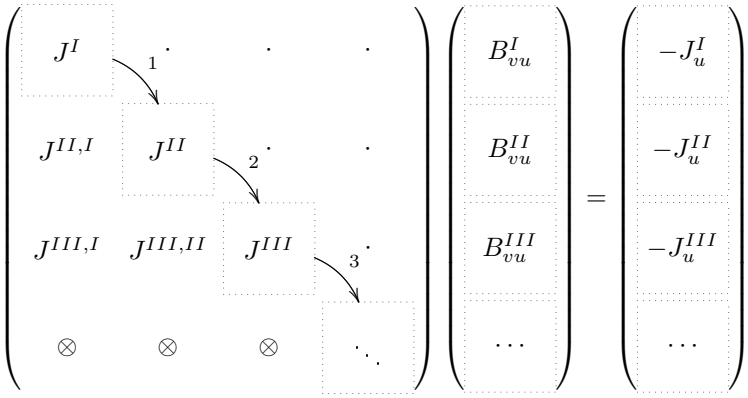
$$\dot{q}_v = B_{vu}\ \dot{q}_u \tag{2.26}$$

$$\text{où} \quad B_{vu} \triangleq -J_v^{-1}(q)\ J_u(q) \tag{2.27}$$

B_{vu} est appelée la matrice de couplage des vitesses [13]. Cette matrice joue un rôle essentiel dans la formulation réduite des équations du mouvement. En pratique, elle est obtenue colonne par colonne en résolvant $n-m$ systèmes linéaires d'équations dont les termes indépendants sont les colonnes de la matrice J_u :

$$J_v(q)\ B_{vu} = -J_u(q) \tag{2.28}$$

Nous allons à nouveau utiliser la structure bloc triangulaire de la matrice J_v et calculer les termes de B_{vu} de façon récursive, bloc par bloc. Toutefois, dans ce cas, nous ne pouvons pas négliger les termes situés hors des blocs diagonaux de J_v. On notera que les lignes de J_u doivent subir les mêmes permutations que celles qui permettent d'obtenir la structure bloc de J_v.



Le système d'équations (2.28) peut alors être résolu bloc par bloc comme ceci :

$$\begin{aligned} & J^I \cdot B^I_{vu} = -J^I_u \\ \hookrightarrow \quad & J^{II} \cdot B^{II}_{vu} = -J^{II}_u - J^{II,I} \cdot B^I_{vu} \\ \hookrightarrow \quad & J^{III} \cdot B^{III}_{vu} = -J^{III}_u - J^{III,I} \cdot B^I_{vu} - J^{III,II} \cdot B^{II}_{vu} \\ & \vdots \end{aligned}$$

Les vitesses dépendantes peuvent alors être calculées par groupe à l'aide des équations suivantes :

$$\begin{aligned} \dot{q}^I_v &= B^I_{vu}\ \dot{q}_u \\ \dot{q}^{II}_v &= B^{II}_{vu}\ \dot{q}_u \\ \dot{q}^{III}_v &= B^{III}_{vu}\ \dot{q}_u \\ &\vdots \end{aligned}$$

Cette méthode de calcul des vitesses n'est pas la plus économique, mais comme nous l'avons mentionné plus haut, elle est justifiée par la nécessité de disposer de la matrice de couplage des vitesses pour effectuer ultérieurement la réduction des équations du mouvement. Dans un contexte différent, où on ne désirerait pas obtenir explicitement la matrice B_{vu}, on notera que les vitesses dépendantes q_v peuvent être obtenue en résolvant l'équation (2.25) bloc par

bloc comme ceci :

$$\begin{aligned} & J^I \cdot \dot{q}_v^I = -J_u^I \, \dot{q}_u \\ \hookrightarrow \quad & J^{II} \cdot \dot{q}_v^{II} = -J_u^{II} \, \dot{q}_u - J^{II,I} \cdot \dot{q}_v^I \\ \hookrightarrow \quad & J^{III} \cdot \dot{q}_v^{III} = -J_u^{III} \, \dot{q}_u - J^{III,I} \cdot \dot{q}_v^I - J^{III,II} \cdot \dot{q}_v^{II} \\ & \vdots \end{aligned}$$

2.4.3 Calcul des accélérations généralisées dépendantes

La résolution des dérivées secondes des équations de contraintes consiste également en la résolution d'un système d'équations linéaires dont les inconnues sont cette fois les accélérations généralisées dépendantes $\ddot{q}_v$.

Les valeurs des coordonnées et des vitesses dépendantes doivent être connues avant de pouvoir évaluer les expressions générées pour le calcul des accélérations.

L'équation (2.3) peut être récrite comme ceci en considérant le partitionnement $\{q_u; q_v\}$ des coordonnées :

$$J(q) \; \ddot{q} + \dot{J}(q, \dot{q})\dot{q} = J_u(q) \; \ddot{q}_u + J_v(q) \; \ddot{q}_v + \dot{J}(q, \dot{q})\dot{q} = 0 \tag{2.29}$$

Le système linéaire à résoudre est donc le suivant :

$$J_v(q) \; \ddot{q}_v = -J_u(q) \; \ddot{q}_u - \dot{J}(q, \dot{q})\dot{q} \tag{2.30}$$

Et les valeurs des accélérations dépendantes peuvent alors être obtenues par l'évaluation de l'expression

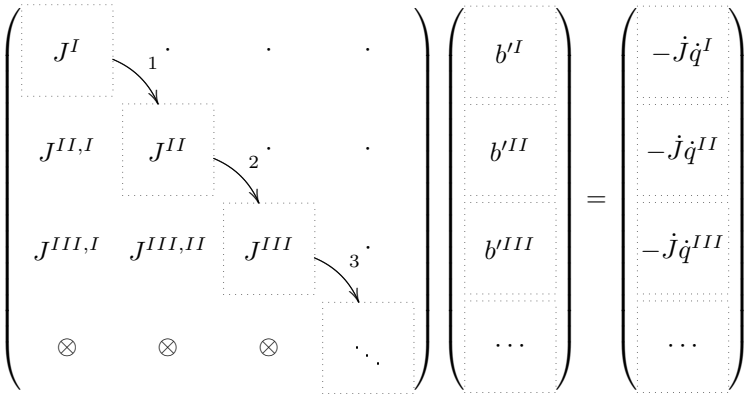
$$\ddot{q}_v = B_{vu} \; \ddot{q}_u + b' \tag{2.31}$$

$$\text{où} \quad b' \triangleq -J_v^{-1}(q) \; \dot{J}(q, \dot{q})\dot{q} \tag{2.32}$$

Le vecteur b' est obtenu en résolvant le système linéaire d'équations suivant :

$$J_v(q) \; b' = -\dot{J}(q, \dot{q})\dot{q} \tag{2.33}$$

Nous utilisons à nouveau la structure bloc triangulaire de la matrice J_v pour calculer les termes de b' de façon récursive, bloc par bloc. Les termes de $\dot{J}(q, \dot{q})\dot{q}$ doivent subir les mêmes permutations que celles qui permettent d'obtenir la structure bloc de J_v.



Le système d'équations (2.33) est alors résolu bloc par bloc comme ceci :

$$
\begin{aligned}
& J^I \cdot b'^I = -\dot{J}\dot{q}^I \\
\hookrightarrow \quad & J^{II} \cdot b'^{II} = -\dot{J}\dot{q}^{II} - J^{II,I} \cdot b'^I \\
\hookrightarrow \quad & J^{III} \cdot b'^{III} = -\dot{J}\dot{q}^{III} - J^{III,I} \cdot b'^I - J^{III,II} \cdot b'^{II} \\
& \vdots
\end{aligned}
$$

Les accélérations dépendantes peuvent être calculées à l'aide des expressions suivantes :

$$
\begin{aligned}
\ddot{q}_v^I &= B_{vu}^I \; \ddot{q}_u + b'^I \\
\ddot{q}_v^{II} &= B_{vu}^{II} \; \ddot{q}_u + b'^{II} \\
\ddot{q}_v^{III} &= B_{vu}^{III} \; \ddot{q}_u + b'^{III} \\
&\vdots
\end{aligned}
$$

2.5 Génération symbolique des équations du mouvement

Dans cette section nous allons montrer comment on peut générer symboliquement une *formulation réduite* [6] des équations du mouvement pour des

[6] Cette formulation réduite des équations du mouvement est parfois appelée "*embedded formulation*" en anglais

systèmes contenant des boucles cinématiques. L'obtention des équations réduites repose sur le partitionnement des coordonnées [37] et sur l'élimination des multiplicateurs de Lagrange λ qui sont présents dans les équations (2.4).

Cette réduction de l'ensemble des $n+m$ équations différentielles-algébriques fournit $n-m$ équations purement différentielles qui peuvent être intégrées en utilisant des méthodes numériques classiques.

Nous présentons ici une formulation réduite où les contraintes sont résolues non seulement en accélération, mais également en vitesse **et** en position [13]. Nous utilisons pour cela la méthode présentée dans la section précédente.

La formulation réduite proposée par d'autres équipes [53, 6] qui utilisent également l'approche symbolique, n'incluent pas systématiquement cette résolution complète des contraintes. Dans ces formulations réduites, les valeurs des vitesses et des positions dépendantes sont obtenues par une double intégration des accélérations dépendantes, lesquelles sont calculées explicitement à l'aide des dérivées seconde des contraintes. Ceci peut poser des problèmes de divergence et d'ouverture des boucles lorsque les modèles sont intégrés pendant des temps relativement longs. Il peut alors s'avérer nécessaire d'introduire des termes de stabilisation dans les expressions utilisées pour le calcul des accélérations dépendantes. Nous évitons totalement ce problème en calculant explicitement les positions et les vitesses dépendantes par résolution des équations de contraintes et de leurs dérivées premières.

Une *formulation augmentée* est parfois utilisée également. Elle est proposée ici essentiellement dans un but de comparaison. Dans cette formulation augmentée, les contraintes ne sont pas résolues mais intégrées conjointement aux équations du mouvement, ce qui fournit les valeurs des coordonnées dépendantes. L'intégration de la formulation augmentée nécessite l'utilisation d'une méthode de stabilisation telle que la méthode de "Baumgarte" [48], afin d'éviter la divergence due à l'accumulation d'erreur sur les coordonnées dépendantes et leur dérivées.

Nous présentons pour commencer le cas de la génération d'une formulation réduite implicite aussi appelée modèle dynamique inverse. Bien que cette formulation puisse également être utilisée pour la simulation de systèmes, en utilisant des méthodes d'intégration implicites [44], elle est utilisée, à la base, pour calculer les forces généralisées développées par les actionneurs présents dans le système.

2.5.1 Formulation implicite

L'équation (2.4) peut être obtenue sous forme implicite comme cela a été montré dans le chapitre précédent. Pour un système soumis à des contraintes, elle s'écrit de la façon suivante :

$$\Phi(q, \dot{q}, \ddot{q}, ...) = Q + J^T \lambda \tag{2.34}$$

En tenant compte de la partition $\{q_u; q_v\}$ des coordonnées généralisées, on peut écrire (2.34) comme ceci :

$$\begin{pmatrix} \Phi_u \\ \Phi_v \end{pmatrix} = \begin{pmatrix} Q_u \\ Q_v \end{pmatrix} + \begin{pmatrix} {J_u}^T \\ {J_v}^T \end{pmatrix} \lambda$$

En éliminant les multiplicateurs de Lagrange λ on obtient l'équation suivante :

$$\Phi_u + {B_{vu}}^T \Phi_v = Q_u + B_{vu}^T Q_v$$

Dans le cas de la dynamique inverse, les inconnues sont les efforts à fournir par les actionneurs du système de telle sorte que celui-ci évolue conformément à la trajectoire décrite par l'évolution temporelle des coordonnées généralisées et de leurs dérivées $\{q, \dot{q}, \ddot{q}\}$

Les forces généralisées articulaires Q comprennent les efforts Q_a développés par les actionneurs ainsi que les forces généralisées passives Q_p provenant d'éléments élastiques ou dissipatifs présents dans les articulations. On définit l'ensemble q_a des articulations actionnées.

Pour l'articulation j on aura alors :

$$Q_j = \begin{cases} Q_{jp} + Q_{ja} \text{ si } j \in q_a \\ Q_{jp} + 0 \text{ sinon.} \end{cases} \tag{2.35}$$

On remarquera que la partition $\{q_u : q_v\}$ des coordonnées est indépendante de la répartition des actionneurs dans le système. On utilise deux sous divisions des coordonnées :

- q_{ua} et q_{va} correspondent aux articulations actionnées,
- q_{una} et q_{vna} correspondent aux articulations non actionnées.

En utilisant la définition suivante :

$$\Psi = (\Phi_u - Q_{up}) + B_{vu}^T (\Phi_v - Q_{vp}) \tag{2.36}$$

On peut récrire les équations dynamiques implicites comme ceci :

$$\begin{pmatrix} \Psi_{ua} \\ \Psi_{una} \end{pmatrix} = G \begin{pmatrix} Q_{ua} \\ Q_{va} \end{pmatrix} \tag{2.37}$$

$$\text{avec } G = \left(\begin{array}{c|c} E_{ua} & \\ \hline 0 & B_{vua}^T \end{array} \right) \tag{2.38}$$

où E_{ua} est une matrice identité et B_{vua} est la matrice obtenue en sélectionnant les lignes de la matrice B_{vu} qui correspondent aux coordonnées dépendantes relatives aux articulations actionnées.

La taille de la matrice G dépend du nombre d'actionneurs présents dans le système. Si le nombre d'actionneurs est égal au nombre de degrés de liberté, alors la matrice G est carrée[7].

On dit que le système est *sur-actionné* si le nombre d'actionneurs est supérieur au nombre de degrés de liberté. Dans ce cas, la matrice G possède plus de colonnes que de lignes et est donc rectangulaire horizontale.

Nous pouvons générer les expressions symboliques des efforts articulaires Q_a développés par les actionneurs, même si le nombre d'actionneurs est supérieur au nombre de degrés de liberté.

Dans le cas inverse, le système est *sous-actionné* et la matrice G possède moins de colonnes que de lignes.

Les expressions des efforts des actionneurs ne sont pas calculés explicitement dans ce cas, seuls les termes de Ψ et de la matrice G sont générés symboliquement.

Systèmes non suractionnés

Lorsque le nombre d'actionneurs est égal au nombre de degrés de liberté, il est possible de résoudre le système d'équations (2.37) en fonction des forces généralisées développées par les actionneurs, à condition que la matrice G soit de plein rang.

$$Q_a = G^{-1}\Psi \tag{2.39}$$

Les forces généralisées Q_a sont obtenues par factorisation LU symbolique de G et élimination de Gauss, comme dans le cas de la résolution des équations du mouvements des systèmes simples présentés dans le chapitre précédent.

Cas des sytèmes sur-actionnés

Le cas des systèmes sur-actionnés peut également être traité symboliquement. Nous utilisons la *pseudo-inverse* qui correspond à la solution de norme Frobenius minimale. Si le rang de la matrice G est égal au nombre de lignes, c'est-à-dire au nombre de variables indépendantes, on peut écrire l'équation suivante qui fournit la solution pour les forces généralisées Q_a :

$$Q_a = G^T(G\ G^T)^{-1}\Psi \tag{2.40}$$

Cette solution est obtenue en deux étapes :

[7] L'égalité du nombre d'actionneurs et du nombre de degrés de liberté ne garantit toutefois pas que la matrice G soit de plein rang. En effet, selon la répartition des actionneurs, le système peut être localement sur-actionné et néanmoins globalement sous-actionné.

1. Tout d'abord, on résout symboliquement le système d'équations suivant en les inconnues x
$$Ax = \Psi \text{ avec } A = G\ G^T \tag{2.41}$$
On remarquera que la matrice intermédiaire A est symétrique et peut donc être factorisée en utilisant la méthode LDL^T. La valeur de x est obtenue ensuite par élimination de Gauss.
2. Ensuite on obtient les expressions de Q_a en évaluant l'expression algébrique suivante
$$Q_a = G^T x \tag{2.42}$$

2.5.2 Formulation explicite

Cette formulation permet d'obtenir une forme réduite des équations du mouvement. Cette forme réduite est utilisée pour la simulation des systèmes contenant des boucles cinématiques. Les équations obtenues sont purement différentielles et correspondent aux équations d'état non-linéaires du système.

Pour obtenir cette forme réduite, nous commençons par générer la matrice de masse M et le vecteur non linéaire c en utilisant le formalisme Newton-Euler Récursif présenté dans le chapitre précédent, en se basant sur la structure arborescente obtenue après coupure des boucles. Les contributions des forces généralisées Q et des forces extérieures sont calculées comme pour les systèmes arborescents. Nous procédons ensuite au calcul de la matrice de masse réduite $\mathcal{M}$ et du vecteur réduit $\mathcal{F}$ définis par les équations (2.45) et (2.46) écrites ci-après. Finalement, les expressions des accélérations sont obtenues par résolution symbolique du système d'équations réduites (2.48).

En fin de section, nous mettons en évidence l'intérêt de l'approche symbolique par rapport à l'approche numérique pour la génération de la formulation réduite des équations du mouvement.

Génération de la forme réduite des équations du mouvement

En tenant compte de la partition $\{q_u; q_v\}$ des coordonnées généralisées, on peut écrire le système d'équations du mouvement sous la forme :

$$\begin{pmatrix} M_{uu} & M_{uv} \\ M_{vu} & M_{vv} \end{pmatrix} \begin{pmatrix} \ddot{q}_u \\ \ddot{q}_v \end{pmatrix} + \begin{pmatrix} c_u \\ c_v \end{pmatrix} = \begin{pmatrix} Q_u \\ Q_v \end{pmatrix} + \begin{pmatrix} {J_u}^T \\ {J_v}^T \end{pmatrix} \lambda$$

Les multiplicateurs de Lagrange λ peuvent être exprimés comme ceci

$$\lambda = J_v^{-t} \left[\begin{pmatrix} M_{vu} & M_{vv} \end{pmatrix} \begin{pmatrix} \ddot{q}_u \\ \ddot{q}_v \end{pmatrix} + c_v - Q_v \right] \tag{2.43}$$

et éliminés pour obtenir l'équation suivante où réapparaît la matrice de couplage $B_{vu} = -J_v^{-1} J_u$:

$$\begin{pmatrix} M_{uu} & M_{uv} \end{pmatrix} \begin{pmatrix} \ddot{q}_u \\ \ddot{q}_v \end{pmatrix} + {B_{vu}}^T \begin{pmatrix} M_{vu} & M_{vv} \end{pmatrix} \begin{pmatrix} \ddot{q}_u \\ \ddot{q}_v \end{pmatrix} + (c_u - Q_u) + {B_{vu}}^T (c_v - Q_v) = 0$$

Ensuite, en introduisant les expressions des accélérations dépendantes $\ddot{q}_v = B_{vu}\ddot{q}_u + b'$ obtenues par résolutions des dérivées secondes des contraintes, on obtient le système suivant :

$$\left(M_{uu} + M_{uv}B_{vu} + {B_{vu}}^T M_{vu} + {B_{vu}}^T M_{vv} B_{vu}\right) \ddot{q}_u + \left(M_{uv} + {B_{vu}}^T M_{vv}\right) b' + (c_u - Q_u) + {B_{vu}}^T (c_v - Q_v) = 0$$

que nous écrivons de façon plus compacte comme ceci :

$$\mathcal{M}(u)\ \ddot{q_u} + \mathcal{F}(\dot{u}, u) = 0 \tag{2.44}$$

avec :

$$\mathcal{M} \triangleq M_{uu} + M_{uv}B_{vu} + {B_{vu}}^T (M_{vu} + M_{vv}B_{vu}) \tag{2.45}$$

$$\mathcal{F} \triangleq (c_u - Q_u) + M_{uv}\, b' + {B_{vu}}^T (c_v - Q_v + M_{vv}\, b') \tag{2.46}$$

Parmi les coordonnées indépendantes q_u se trouvent les coordonnées libres q_f et les coordonnées commandées q_c. $\mathcal{M}$ et $\mathcal{F}$ peuvent être récris en conséquence :

$$\begin{pmatrix} \mathcal{M}_{ff} & \mathcal{M}_{fc} \\ \mathcal{M}_{cf} & \mathcal{M}_{cc} \end{pmatrix} \begin{pmatrix} \ddot{q}_f \\ \ddot{q}_c \end{pmatrix} + \begin{pmatrix} \mathcal{F}_f \\ \mathcal{F}_c \end{pmatrix} = 0 \tag{2.47}$$

La valeur des accélérations des coordonnées commandées sont connues car elles sont imposées. Les équations du mouvement peuvent donc finalement s'écrire :

$$\mathcal{M}_r\ \ddot{q_f} + \mathcal{F}_r = 0 \tag{2.48}$$

où

$$\mathcal{M}_r = \mathcal{M}_{ff} \tag{2.49}$$

$$\mathcal{F}_r = \mathcal{F}_f + \mathcal{M}_{fc}\ddot{q}_c \tag{2.50}$$

On utilise la même technique de factorisation de Choleski ou LDL^T que pour la résolution du système d'équations du mouvement des systèmes simples présentée dans la section 1.3.4.

Après résolution du système d'équation (2.48), on obtient alors explicitement les accélérations libres $\ddot{q_f}$:

$$\ddot{q}_f = \phi(q, \dot{q}, F_{ext}, L_{ext}, g) \tag{2.51}$$

Les accélérations dépendantes $\ddot{q}_v$ peuvent être obtenues en résolvant les équations (2.31) des dérivées secondes des contraintes comme montré dans la section précédente.

Les expressions des multiplicateurs de Lagrange peuvent être générées en utilisant l'équation (2.43) dans laquelle on a substitué $\ddot{q}_v$ en fonction de $\ddot{q}_u$:

$$J_v^T \lambda = (M_{vu} + M_{vv} B_{vu}) \ddot{q}_u + (c_v - Q_v + M_{vv}\, b') \tag{2.52}$$

Cette formulation est un peu plus économique car elle réutilise des termes déjà générés précédemment pour le calcul des termes de $\mathcal{M}$ et de $\mathcal{F}$ et ne nécessite pas la génération explicite des accélérations dépendantes.

Diagramme général

La figure 2.9 illustre l'organisation des différentes parties d'un modèle dynamique direct réduit tel que nous venons de le présenter.

Intégration numérique

La simulation du mouvement d'un système multicorps peut être effectuée par intégration temporelle des équations (2.51). Toutefois, comme ces équations sont des équations différentielles du second ordre, il est habituel de les récrire sous la forme d'une équation d'état directement utilisable par une méthode d'intégration classique :

$$\dot{x} = f(x) \tag{2.53}$$

où le vecteur d'état x est défini comme ceci :

$$x = \begin{pmatrix} \dot{q}_u \\ q_u \end{pmatrix} \tag{2.54}$$

et

$$f(x) = \begin{pmatrix} \phi(q, \dot{q}, F_{ext}, L_{ext}, g) \\ \dot{q}_u \end{pmatrix} \tag{2.55}$$

Une attention particulière doit être apportée au fait que les valeurs des coordonnées dépendantes sont obtenues par résolution itérative des équations de contraintes en utilisant la méthode de Newton-Raphson. Bien que cette méthode soit évidemment moins efficace que l'évaluation d'une solution analytique telle qu'envisagée dans [45], elle offre une très bonne performance en simulation car :

1. les éléments symboliques représentant la plus grande quantité de calcul, à savoir la matrice J_v et sa forme LU, sont de toute façon nécessaires par ailleurs pour le calcul d'autres éléments utilisés pour la réduction des équations,

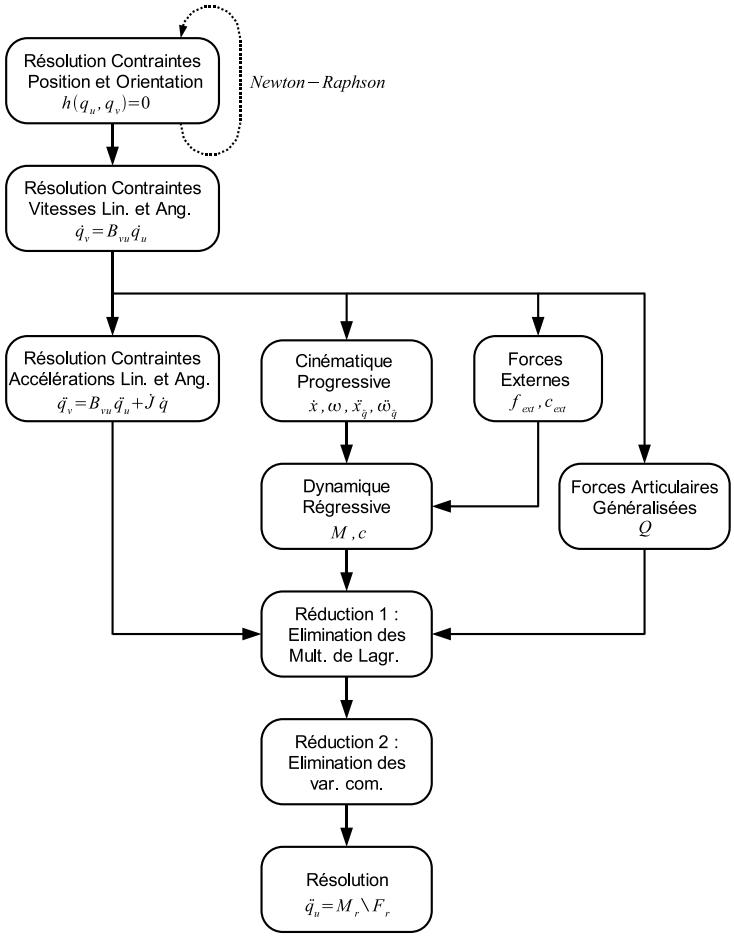


FIG. 2.9 – Diagramme du modèle dynamique direct réduit

2. le nombre d'itérations nécessaire pour atteindre une précision correcte de l'ordre de 10^{-10} sur la norme des contraintes, est relativement faible : une ou deux itérations sont généralement suffisantes, car les valeurs initiales utilisées pour la méthode de Newton-Raphson proviennent du pas de temps précédent $q_v^{t-\Delta t}$.

Néanmoins, dans le cas de la simulation de systèmes où les valeurs de vitesses articulaires dépendantes $\dot{q}_v$ sont importantes, il est utile d'ajouter également ces vitesses dépendantes dans le vecteurs d'état x présenté ci-dessus :

$$x = \begin{pmatrix} \dot{q}_u \\ q \end{pmatrix} \tag{2.56}$$

L'intégration des vitesses dépendantes $\dot{q}_v$ fournit alors une excellente estimation q_v^{*t} des valeurs des coordonnées dépendantes. Pour la méthode itérative de Newton-Raphson, ces valeurs q_v^{*t} constituent au pas de temps t, des conditions initiales qui sont bien meilleures que les valeurs de $q_v^{t-\Delta t}$ calculées lors du dernier pas de temps d'intégration. Les calculs supplémentaires requis pour l'intégration de ces variables sont largement compensés par l'économie des itérations supplémentaires nécessaires pour obtenir les valeurs précises des q_v lorsque les valeurs du pas de temps précédent sont utilisées comme estimations initiales. Ces deux méthodes sont illustrées sur la figure 2.10.

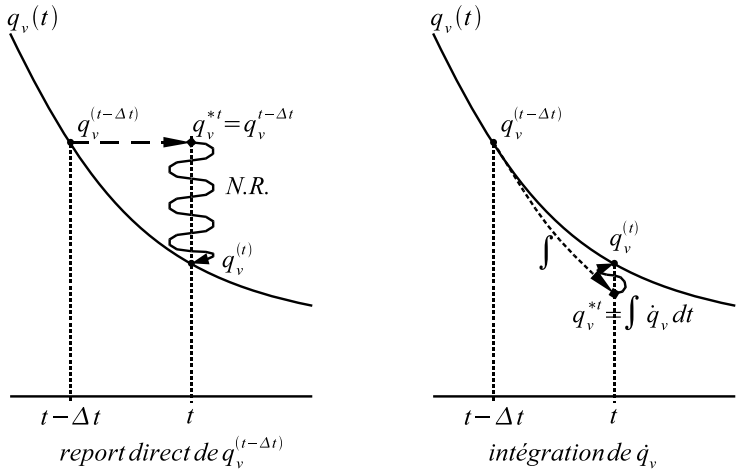


FIG. 2.10 – Estimation de q_v par report direct ou par intégration

Remarque Pour la simulation de systèmes dont les vitesses articulaires dépendantes $\dot{q}_v$ varient faiblement, les valeurs des coordonnées dépendantes ob-

tenues par intégration des vitesses permettent souvent de satisfaire les critères de précision imposés pour la fermeture des boucles dans ce cas. On se retrouve alors au même niveau de performance que celui des modèles réduits où les équations de contraintes ne sont pas explicitement résolues, tout en gardant l'avantage du contrôle de la précision.

2.5.3 Formulation augmentée

Cette formulation consiste à ajouter les dérivées secondes des équations de contraintes aux équations du mouvement et à les intégrer conjointement. Ceci conduit à résoudre un système de $n+m-c$ équations au lieu de $n-m-c$ dans le cas de la formulation réduite.

Les équations (2.4) sont récrites sous la forme suivante :

$$\begin{pmatrix} M & -J^T \\ J & 0 \end{pmatrix} \begin{pmatrix} \ddot{q} \\ \lambda \end{pmatrix} = \begin{pmatrix} Q-c \\ -\dot{J}\dot{q} \end{pmatrix}$$

Les inconnues sont donc les n coordonnées généralisées q et les m multiplicateurs de Lagrange λ. Comme dans le cas de la formulation réduite, les accélérations des variables commandées sont déjà connues et il est donc nécessaire d'enlever les lignes et les colonnes de la matrice de masse M ainsi que les colonnes de la matrice Jacobienne J qui correspondent aux articulations commandées. Le membre de droite est également modifié en conséquence.

On notera M_{aug} la matrice augmentée du système

$$M_{aug} = \begin{pmatrix} M_{ff} & M_{fv} & -J_f^T \\ M_{vf} & M_{vv} & -J_v^T \\ J_f & J_v & 0 \end{pmatrix}$$

et F_{aug} le vecteur des termes indépendants

$$F_{aug} = \begin{pmatrix} Q_f - c_f - M_{fc}\ddot{q}_c \\ Q_v - c_v - M_{vc}\ddot{q}_c \\ -\dot{J}\dot{q} - J_c\ddot{q}_c \end{pmatrix}$$

Ce qui permet d'écrire les équations augmentées du sytèmes :

$$M_{aug} \begin{pmatrix} \ddot{q}_{nc} \\ \ddot{q}_v \\ \lambda \end{pmatrix} = F_{aug} \quad \text{avec} \quad q_{nc} = \begin{pmatrix} q_f \\ q_v \end{pmatrix} \tag{2.57}$$

On notera que la matrice M_{aug} peut ne pas être régulière si certains corps terminaux ont une masse ou une inertie nulle[8], ou si toutes les contraintes ne

[8] Ceci peut se produire, par exemple, lorsqu'on utilise des coupures de corps pour ouvrir des boucles.

sont pas indépendantes. Pour la suite, nous faisons l'hypothèse que M_{aug} est bien régulière.

Ces équations peuvent être résolues telles quelles pour obtenir les expressions des accélérations articulaires $\ddot{q}$ et des multiplicateurs de Lagrange λ. Les accélérations peuvent alors être intégrées directement en utilisant une méthode numérique classique, en introduisant des termes de stabilisation qui font intervenir les contraintes ainsi que leurs dérivées premières dans les équations des dérivées secondes [38, 48].

Dans le cas de l'utilisation de la méthode de stabilisation de Baumgarte [38], cela donne :

$$M_{aug} \begin{pmatrix} \ddot{q}_{nc} \\ \lambda \end{pmatrix} = F_{aug} - \begin{pmatrix} 0 \\ 2\alpha\dot{h} + \beta^2 h(q) \end{pmatrix} \tag{2.58}$$

où les constantes $\alpha > 0$ et β doivent être choisies de façon adéquate, ce qui représente un inconvénient de la méthode. Il est en effet difficile de trouver des indications pour effectuer les choix des valeurs numériques de ces constantes.

Les valeurs de $\ddot{q}$ et de λ peuvent être obtenue par résolution du système linéaire d'équation (2.58). Bien que la matrice augmentée M_{aug} soit symétrique, il n'est pas possible de résoudre le système en utilisant une méthode de factorisation de type Choleski ou LDL^T car la partie inférieure de la diagonale est nulle.

Étant donnée sa structure, le pré-conditionnement requis pour la factorisation nécessite d'effectuer des permutations différentes des lignes et des colonnes. Dans notre implémentation, des permutations des colonnes sont effectuées, ce qui a pour conséquence de rompre cette propriété de symétrie. Nous utilisons alors une factorisation LU pour la résolution du système.

$$\begin{pmatrix} \ddot{q}_{nc} \\ \lambda \end{pmatrix} = \Phi_{aug}(q, \dot{q}, F_{ext}, L_{ext}, g) \tag{2.59}$$

Tout comme pour l'intégration des expressions des accélérations indépendantes obtenues par la formulation réduite présentée ci-dessus, les équations différentielles du second ordre (2.59) sont récrites sous la forme d'équations d'état :

$$\dot{x} = f_{nc}(x)$$

où le vecteur d'état x est défini comme ceci :

$$x = \begin{pmatrix} \dot{q}_{nc} \\ q_{nc} \end{pmatrix} \tag{2.60}$$

et

$$f_{nc}(x) = \begin{pmatrix} \Phi_{aug\{\ddot{q}_{nc}\}}(q, \dot{q}, F_{ext}, L_{ext}, g) \\ \dot{q}_{nc} \end{pmatrix} \tag{2.61}$$

Compte tenu de la structure de la matrice M_{aug}, il n'est pas possible d'éviter de calculer les multiplicateurs de Lagrange λ lors de la résolution du système d'équations (2.58) qui fournit les valeurs des accélérations $\ddot{q}_{nc}$.

2.6 Génération symbolique vs numérique

Dans les sections précédentes, nous avons présenté les différentes étapes permettant de traiter le cas des systèmes multicorps soumis à des contraintes provenant des boucles cinématiques présentes dans leur structure. La méthode du partitionnement de coordonnées qui permet d'obtenir une forme purement différentielle des équations du mouvement[9] est une technique bien connue et généralement utilisée dans le cas de la génération numérique des équations.

Nous l'avons implémentée dans le logiciel ROBOTRAN pour la génération symbolique des modèles dynamiques directs et inverses de systèmes à topologie bouclée. L'approche symbolique offre des avantages bien connus en termes de simplifications et de réduction du nombre d'opérations arithmétiques dans le cas de la génération, par le biais de formalismes récursifs ou non, des équations du mouvement pour des systèmes arborescent. Elle se montre tout aussi intéressante, sinon plus, dans le contexte des systèmes à topologie complexe.

Nous en résumons les raisons principales relatives à la résolution des équations de contrainte et à la réduction des équations du mouvement dans les sections suivantes.

2.6.1 Traitement des contraintes

Dans la section relative à la résolution des contraintes, nous avons montré que la matrice J_v pouvait être factorisée sous forme bloc triangulaire. Cette structure offre deux avantages intéressants :

1. seuls les blocs diagonaux de J_v sont triangularisés par factorisation LU et non plus la matrice J_v toute entière, ce qui constitue une économie de calculs considérable étant donné la complexité $O(N^3/3)$ de la factorisation LU.
2. les m équations de contraintes peuvent être résolues séparément, groupe par groupe, selon une séquence déterminée par la structure de J_v. Cette séparation permet de réduire le nombre total d'évaluations des équations de contraintes lors du processus itératif de résolution.

La génération symbolique permet d'effectuer cette analyse a priori et d'effectuer les manipulations nécessaires des équations pour les agencer de telle sorte que leur évaluation se fasse selon une séquence optimale. Ce type d'opération est effectué une fois pour toutes lors de la génération, ce qui convient

[9] par élimination des multiplicateurs de Lagrange associés aux contraintes

parfaitement à l'approche symbolique, dans laquelle les phases de génération symbolique et d'évaluation numérique des équations sont distinctes. Bien que ce ne soit pas impossible, il serait relativement plus délicat de mettre en oeuvre un processus similaire dans un logiciel purement numérique. En effet, dans le cas de l'approche numérique, les phases de génération et d'évaluation des équations sont identiques.

2.6.2 Réduction des équations du mouvement

L'utilisation de la méthode du partitionnement des coordonnées et l'élimination des multiplicateurs de Lagrange pour la réduction des équations du mouvement conduit à effectuer un nombre important d'opérations matricielles comme le montrent les équations (2.45) et (2.46).

Les matrices impliquées sont essentiellement la matrice de masse globale M et la matrice de couplage B_{vu}. Dans le cas de l'analyse des systèmes contenant des boucles cinématiques, ces matrices sont généralement creuses. La génération symbolique des termes des matrices réduites $\mathcal{M}_r$ et $\mathcal{F}_r$ définies par les équations (2.45), (2.46) et (2.50) permet de réaliser une économie substantielle de calculs, grâce à l'élimination des opérations relatives aux éléments nuls.

Système	#Artic.	#D.d.L.	#Contr.	Num.	Symb.	Gain
Robot Delta	12	3	9	900	60	15
Plate forme Stewart	24	6	18	6840	330	20
Voiture Audi A6	54	14	40	78036	1897	41
Bogie articulé	53	29	24	146257	1075	136

TAB. 2.1 – Nombre d'opérations pour effectuer la réduction

La table 2.1 reprend des valeurs obtenues pour quelques systèmes. Les nombres d'articulations (#Artic.), de degrés de liberté (#D.d.L.) et de contraintes (#Contr.) sont indiqués pour chaque système. Les nombres d'opérations pour un processus numérique (Num.) sont estimés en fonction des tailles des matrices correspondant aux caractéristiques respectives des systèmes. Les nombres d'opérations relatives à la réduction des modèles générés symboliquement (Symb.) par ROBOTRAN sont également repris et comparés en terme de gain relatif.

On remarque que la génération symbolique permet, en moyenne, de réduire d'un facteur supérieur à 20, le nombre d'opérations arithmétiques en virgule flottante nécessaires à la réduction des équations du mouvement. On remarque que pour certains systèmes plus complexes tels que le bogie articulé, un modèle généré par l'approche numérique peut impliquer jusqu'à 136 fois plus d'opérations qu'un modèle généré par l'approche symbolique.

Cette constatation montre de façon évidente l'intérêt de l'approche symbolique pour la génération de modèles dynamiques utilisés pour la simulation de systèmes multicorps. On notera que l'application de la technique du partitionnement des coordonnées et de l'élimination des multiplicateurs de Lagrange associés aux contraintes pour la génération des équations du mouvement sous forme purement différentielle représente un coût calcul relativement moins important dans les modèles obtenus par génération symbolique que dans le cas de la génération numérique. On n'observe habituellement pas une telle différence entre l'approche symbolique et l'approche numérique pour les autres parties du modèle, comme par exemple pour la génération récursive des équations cinématiques et dynamiques (voir figure 2.9).

Ceci justifie pleinement notre contribution qui vise à générer des modèles symboliques complets, par rapport aux formulations mixtes précédentes [12].

2.7 Illustrations : un mécanisme parallèle plan

Dans cette section, nous allons illustrer l'influence de l'utilisation des coupures et du choix des coordonnées indépendantes, c'est-à-dire du partitionnement des coordonnées, sur les équations générées pour un modèle dynamique direct réduit.

Nous comparons les modèles réduits correspondant à différentes partitions, avec le modèle augmenté où les expressions des contraintes et de leurs dérivées premières et secondes sont incluses pour la stabilisation du modèle selon la méthode de Baumgarte. Dans le cas du modèle augmenté, seule importe la topologie du système ouvert. En effet le partitionnement des coordonnées n'intervient pas dans la génération des équations de cette formulation.

Dans la section suivante, nous présenterons également d'autres systèmes plus réalistes et représentatifs d'applications réelles.

Le mécanisme représenté à la figure 2.11 est un robot manipulateur parallèle plan à 3 degrés de liberté. Il est constitué de trois bras reliant la plate-forme centrale, représentée par un carré, à la base inertielle. Chaque bras est constitué de deux parties rigides. Les articulations sont toutes de type rotoïde. La topologie du système comporte deux boucles cinématiques.

Bien que ce mécanisme plan soit relativement élémentaire, il permet de bien visualiser et comprendre les concepts présentés, ce qui n'est pas toujours évident avec un exemple spatial plus complexe.

2.7.1 Modèles et partitions

Nous présentons dans les paragraphes qui suivent, différentes manières de modéliser ce mécanisme, selon le type de coupures utilisées et leurs emplacements. Pour chaque modèle, on mesure le nombre d'opérations obtenues en

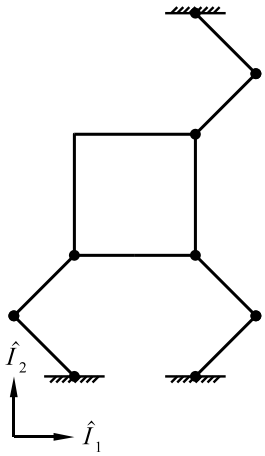


FIG. 2.11 – Un manipulateur parallèle plan

utilisant différentes partitions de coordonnées.

Modèle 1

Un premier modèle arborescent de ce système est illustré sur la figure 2.12. Les boucles sont ouvertes à l'aide de coupures faites au niveau des supports de deux des trois bras.

Les points de coïncidence des repères des corps 6 et 9 avec le corps de référence, sont les positions absolues des articulations 6 et 9.

Cette procédure conserve chacune des neuf articulations et génère deux contraintes de translation et une contrainte de rotation pour chaque coupure : soit six contraintes au total. Il reste donc trois degrés de liberté.

La matrice Jacobienne a la structure suivante :

$$q = \left\{ \begin{array}{ccccccccc} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 & q_9 \end{array} \right\}$$

$$Jac = \left(\begin{array}{ccccccccc} \otimes & \otimes & \otimes & \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \otimes & \otimes & \otimes & \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \otimes & \otimes & \otimes & \otimes & \otimes & \otimes & \cdot & \cdot & \cdot \\ \hline \otimes & \otimes & \otimes & \cdot & \cdot & \cdot & \otimes & \otimes & \cdot \\ \otimes & \otimes & \otimes & \cdot & \cdot & \cdot & \otimes & \otimes & \cdot \\ \otimes & \otimes & \otimes & \cdot & \cdot & \cdot & \otimes & \otimes & \otimes \end{array} \right)$$

Nous allons envisager trois partitions différentes pour ce modèle en utilisant la numérotation des articulations de la figure 2.12.

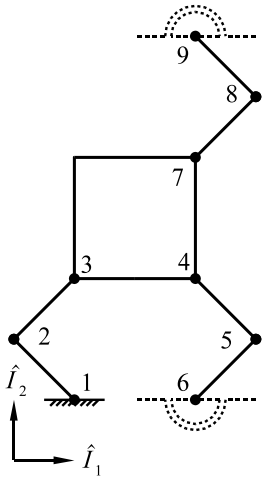


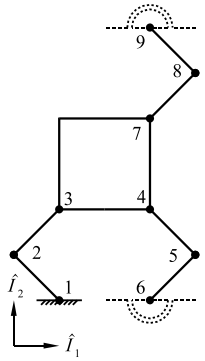
FIG. 2.12 – Coupure de deux supports des articulations de base des bras

Partition 1 On choisit les trois articulations 1, 6 et 9 de base des bras comme variables indépendantes. Les coordonnées dépendantes sont permutées pour obtenir une diagonale non nulle de la matrice J_v.

$$q_u = \{\ q_1 \quad q_6 \quad q_9\ \}$$

$$q_v = \{\ q_4 \quad q_5 \quad q_2 \quad q_3 \quad q_7 \quad q_8\ \}$$

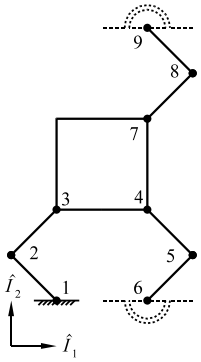
$$J_v = \begin{pmatrix} \otimes & \otimes & \otimes & \otimes & \cdot & \cdot \\ \otimes & \otimes & \otimes & \otimes & \cdot & \cdot \\ \otimes & \otimes & \otimes & \otimes & \cdot & \cdot \\ \cdot & \cdot & \otimes & \otimes & \otimes & \otimes \\ \cdot & \cdot & \otimes & \otimes & \otimes & \otimes \\ \cdot & \cdot & \otimes & \otimes & \otimes & \otimes \end{pmatrix}$$



On constate que la matrice J_v est relativement pleine et ne possède pas de structure bloc intéressante. La matrice de couplage des vitesses B_{vu} est pleine, ce qui implique qu'aucune simplification triviale ne sera faite dans le processus de réduction du modèle. Cette partition conduit à un modèle réduit dont le nombre d'opérations est de l'ordre de $\boxed{1704}$ opérations en virgule flottante.

Partition 2 Cette seconde partition correspond au choix des coordonnées des trois articulations 3, 4 et 7 de la plate-forme comme coordonnées indépendantes.

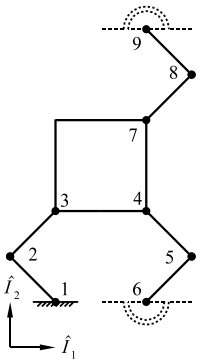
$$q_u = \{\ q_3 \quad q_4 \quad q_7\ \}$$
$$q_v = \{\ q_5 \quad q_1 \quad q_8 \quad q_2 \quad q_6 \quad q_9\ \}$$
$$J_v = \left(\begin{array}{cccc|cc} \otimes & \otimes & \cdot & \otimes & \cdot & \cdot \\ \otimes & \otimes & \cdot & \otimes & \cdot & \cdot \\ \cdot & \otimes & \otimes & \otimes & \cdot & \cdot \\ \cdot & \otimes & \otimes & \otimes & \cdot & \cdot \\ \hline \otimes & \otimes & \cdot & \otimes & \otimes & \cdot \\ \cdot & \otimes & \otimes & \otimes & \cdot & \otimes \end{array}\right) \qquad h_p = \left\{\begin{array}{c} h_1 \\ h_2 \\ h_4 \\ h_5 \\ h_3 \\ h_6 \end{array}\right\}$$



Dans ce cas on observe que la Jacobienne possède une structure bloc. La mise en évidence de cette structure bloc implique une permutation de l'ordre des contraintes. Cette permutation est donnée par la liste h_p. Les quatre contraintes de translation sont résolues ensemble, ensuite les deux contraintes de rotations peuvent être résolues indépendamment l'une de l'autre. Ceci provient du fait que les variables associées aux articulations 6 et 9 sont dépendantes et que les points de référence pris pour faire correspondre les corps 6 et 9 sont précisément les positions de ces articulations. Cette partition conduit à un modèle comportant environ $\boxed{1508}$ opérations.

Partition 3 Cette dernière partition place les trois premières articulations dans l'ensemble des coordonnées indépendantes.

$$q_u = \{\ q_1 \quad q_2 \quad q_3\ \}$$
$$q_v = \{\ q_4 \quad q_5 \quad q_6 \quad q_7 \quad q_8 \quad q_9\ \}$$
$$J_v = \begin{pmatrix} \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \otimes & \otimes & \otimes & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \otimes & \otimes & \cdot \\ \cdot & \cdot & \cdot & \otimes & \otimes & \cdot \\ \cdot & \cdot & \cdot & \otimes & \otimes & \otimes \end{pmatrix} \qquad h_p = \left\{\begin{array}{c} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \end{array}\right\}$$



L'examen de la Jacobienne des contraintes Jac montre que ces trois coordonnées qui appartiennent à la branche commune au deux boucles, sont im-

pliquées dans toutes les contraintes, contrairement aux coordonnées des deux autres bras. Cette partition a l'avantage de découpler complètement les équations de contrainte relative aux deux coupures, comme on peut le constater en examinant la structure de la matrice J_v. Celle-ci contient également moins d'éléments ce qui a un effet favorable sur le nombre d'opérations nécessaires pour traiter les contraintes. Le modèle réduit comporte $\boxed{1447}$ opérations.

Modèle augmenté Le modèle augmenté généré symboliquement comporte $\boxed{1917}$ opérations, soit plus que le modèle réduit obtenu avec la "plus mauvaise" partition. Ceci peut être expliqué par le fait que la formulation augmentée conduit à la résolution d'un système linéaire de 15 équations pour extraire les accélérations généralisées et les multiplicateurs de Lagrange, alors que la formulation réduite conduit à résoudre un système de 3 équations pour obtenir les accélérations indépendantes.

Modèle 2

Le second modèle considéré ne diffère du premier que par le corps qui fait l'objet des coupures et la numérotation des articulations. Dans ce second cas, on découpe la plate-forme autour des articulations qui la relie à deux des trois bras. Toutes les articulations sont également conservées et les points de coïncidence des corps originaux et secondaires sont précisément les positions respectives des articulations 6 et 9 sur la plate-forme. L'intérêt étant à nouveau de découpler les équations de translation et de rotation des coupures.

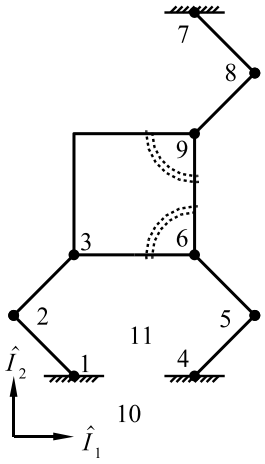


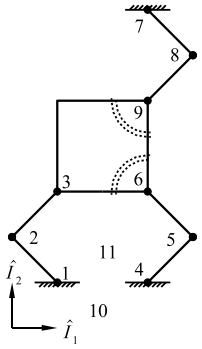
FIG. 2.13 – Modèle 2

La matrice Jacobienne des contraintes possède exactement la même structure que celle du modèle précédent.

Nous envisageons également trois partitions pour ce modèle en utilisant la numérotation des articulations indiquée sur la figure 2.13.

Partition 1 Cette première partition correspond au choix des coordonnées des trois articulations 1, 4 et 7 de base comme coordonnées indépendantes.

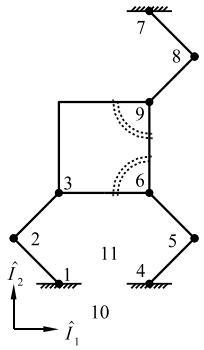
$$q_u = \{\ q_1 \quad q_4 \quad q_7\ \}$$
$$q_v = \{\ q_5 \quad q_2 \quad q_8 \quad q_3 \quad q_6 \quad q_9\ \}$$
$$J_v = \left(\begin{array}{cccc|cc} \otimes & \otimes & \cdot & \otimes & \cdot & \cdot \\ \otimes & \otimes & \cdot & \otimes & \cdot & \cdot \\ \cdot & \otimes & \otimes & \otimes & \cdot & \cdot \\ \cdot & \otimes & \otimes & \otimes & \cdot & \cdot \\ \hline \otimes & \otimes & \cdot & \otimes & \otimes & \cdot \\ \cdot & \otimes & \otimes & \otimes & \cdot & \otimes \end{array}\right) \qquad h_p = \left\{\begin{array}{c} h_1 \\ h_2 \\ h_4 \\ h_5 \\ h_3 \\ h_6 \end{array}\right\}$$



Ce choix est le même que celui de la première partition envisagée pour le modèle précédent, toutefois, étant donné la différence des deux modèles au niveau du placement des coupures, on obtient une situation équivalente à la seconde partition du modèle précédent, avec une structure de matrice J_v identique. Le modèle comporte $\boxed{918}$ opérations avec ce choix.

Partition 2 Cette seconde partition place les articulations 3, 6, et 9 de la plate-forme dans l'ensemble des coordonnées indépendantes.

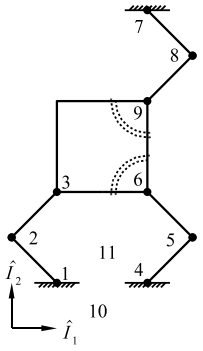
$$q_u = \{\ q_3 \quad q_6 \quad q_9\ \}$$
$$q_v = \{\ q_4 \quad q_5 \quad q_1 \quad q_2 \quad q_7 \quad q_8\ \}$$
$$J_v = \left(\begin{array}{cccccc} \otimes & \otimes & \otimes & \otimes & \cdot & \cdot \\ \otimes & \otimes & \otimes & \otimes & \cdot & \cdot \\ \otimes & \otimes & \otimes & \otimes & \cdot & \cdot \\ \cdot & \cdot & \otimes & \otimes & \otimes & \otimes \\ \cdot & \cdot & \otimes & \otimes & \otimes & \otimes \\ \cdot & \cdot & \otimes & \otimes & \otimes & \otimes \end{array}\right)$$



On se retrouve dans la même situation que dans le cas de la première partition du modèle précédent avec à nouveau une structure de matrice J_v identique. Ce choix de partition produit un modèle qui contient $\boxed{1076}$ opérations.

Partition 3 Cette dernière partition est identique à la troisième partition proposée pour le modèle précédent, ce qui conduit à la même structure de matrice J_v.

$$q_u = \{\ q_1 \quad q_2 \quad q_3\ \}$$
$$q_v = \{\ q_4 \quad q_5 \quad q_6 \quad q_7 \quad q_8 \quad q_9\ \}$$
$$J_v = \begin{pmatrix} \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \otimes & \otimes & \otimes & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \otimes & \otimes & \cdot \\ \cdot & \cdot & \cdot & \otimes & \otimes & \cdot \\ \cdot & \cdot & \cdot & \otimes & \otimes & \otimes \end{pmatrix}$$



Tout comme pour le premier modèle, cette troisième partition conduit à une Jacobienne qui contient moins d'éléments, ce qui a un effet favorable sur le nombre d'opérations nécessaires pour traiter les contraintes. Le modèle réduit comporte cette fois 861 opérations.

Modèle augmenté Le modèle augmenté comporte 1202 opérations, soit toujours plus que le modèle réduit obtenu avec la "plus mauvaise" partition. Les raisons sont les mêmes que dans le cas du modèle précédent.

Observation Tous les modèles produits avec cette seconde disposition des coupures contiennent invariablement moins d'opérations que dans le cas précédent. Ceci s'explique par la différence de longueur des chaînes cinématiques de ce second modèle arborescent. Le nombre d'opérations nécessaires pour la génération récursive des équations cinématiques et dynamiques est en effet d'une complexité $O(n^2)$ où n est la longueur des chaînes. On a donc toujours intérêt à effectuer les coupures de manière à générer une arborescence à branches courtes.

Modèle 3

Avec ce troisième modèle arborescent nous allons montrer l'intérêt de l'utilisation de coordonnées mixtes, c'est-à-dire l'utilisation conjointe de coordonnées relatives et absolues. Nous ajoutons au système une chaîne cinématique fictive qui correspond aux coordonnées absolues de la plate-forme. Nous ajoutons donc trois articulations fictives : 10 et 11 correspondent aux coordonnées cartésiennes

du centre de la plate-forme dans le repère $\{\hat{I}\}$ et 12 correspond à son angle absolu de rotation dans le plan. Cet ajout d'une branche fictive augmente donc le nombre d'articulations et de coordonnées généralisées du système.

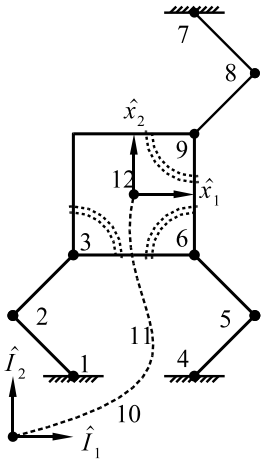


FIG. 2.14 – Modèle 3

La figure 2.14 illustre les trois coupures de la plate-forme et la topologie correspondante ainsi que l'insertion de la branche fictive, qui relie la base à la plate-forme.

Nous allons également utiliser ce modèle pour monter l'influence du choix du point de référence pour les coupures de corps. Dans ce modèle, nous effectuons trois coupures de la plate-forme autour des articulations des bras. Le corps original est celui qui est attaché à la branche fictive qui correspond aux coordonnées absolues introduites.

Nous considérons les deux cas où les points de coïncidence utilisés pour les coupures de la plate-forme sont :

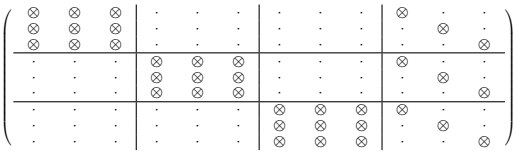
- soit le centre de la plate-forme,
- soit les points respectifs des articulations qui relient la plate-forme aux trois bras.

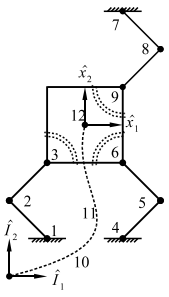
Le choix d'un point différent de la position des articulations, en l'occurrence le centre de la plate-forme, va entraîner un surcoût de calcul, à cause du couplage des contraintes de translation et de rotation pour chaque coupure.

Les Jacobiennes des contraintes obtenues pour les deux choix de points de référence sont les suivantes :

point de référence commun au centre

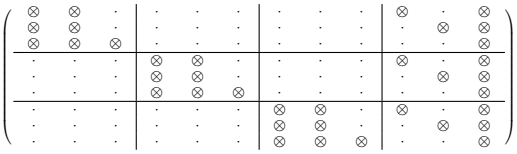
$Jac =$





points de références sur chaque articulation

$Jac =$



On constate que le choix du point de référence au centre a pour conséquence que les trois coordonnées absolues apparaissent séparément dans chaque contrainte, comme le montre la structure des trois dernières colonnes de la matrice.

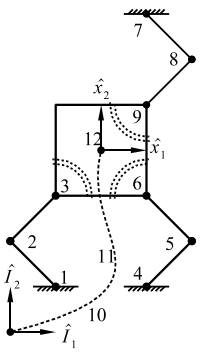
Dans le cas de choix des points de références placés sur chacune des articulations $3, 6, 9$, ce sont les contraintes de translation qui sont indépendantes des coordonnées de ces articulations, mais cette fois la coordonnée 12 de rotation absolue de la plate-forme apparaît dans toutes les contraintes.

Globalement, on observe que la sous-matrice constituée des neuf premières colonnes possède une structure bloc diagonale dans les deux cas. Nous ne considérons donc qu'une seule partition dans laquelle les neuf premières coordonnées sont considérées comme dépendantes, ce qui correspond à utiliser les coordonnées absolues introduites comme indépendantes.

Partition 1 Cette partition est le choix logique après avoir inséré la chaîne cinématique fictive correspondant aux coordonnées absolues. On choisit donc les coordonnées absolues de la plate-forme comme coordonnées indépendantes.

point de référence commun au centre La structure bloc diagonale de la Jacobienne J_v signifie que les contraintes relatives aux trois coupures peuvent être résolues indépendamment les unes des autres.

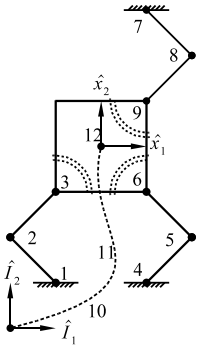
$$q_u = \{\ q_{10}\ q_{11}\ q_{12}\ \}$$
$$q_v = \{\ q_1\ q_2\ q_3\ q_4\ q_5\ q_6\ q_7\ q_8\ q_9\ \}$$
$$J_v = \begin{pmatrix} \otimes & \otimes & \otimes & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \otimes & \otimes & \otimes & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \otimes & \otimes & \otimes & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \otimes & \otimes & \otimes & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \otimes & \otimes & \otimes & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \otimes & \otimes & \otimes & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \otimes & \otimes & \otimes \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \otimes & \otimes & \otimes \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \otimes & \otimes & \otimes \end{pmatrix}$$



Les trois boucles cinématiques sont indépendantes dès lors que la branche commune, à savoir la chaîne cinématique fictive ajoutée est considérée comme figée. Le modèle obtenu dans ce cas contient toutefois encore $\boxed{986}$ opérations, ce qui est supérieur à ce qui a été obtenu avec la partition précédente en plaçant les points de référence des trois coupures sur les articulations $3, 6, 9$.

points de références sur chaque articulation Le placement judicieux des points de référence permet le découplage des équations de contraintes de translation et de rotation pour chaque coupure. On constate ici que les blocs diagonaux sont au nombre de 6.

$$q_u = \{\ q_{10}\ q_{11}\ q_{12}\ \}$$
$$q_v = \{\ q_1\ q_2\ q_3\ q_4\ q_5\ q_6\ q_7\ q_8\ q_9\ \}$$
$$J_v = \begin{pmatrix} \otimes & \otimes & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \otimes & \otimes & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \otimes & \otimes & \otimes & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \otimes & \otimes & \otimes & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \otimes & \otimes & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \otimes & \otimes & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \otimes & \otimes & \otimes \end{pmatrix}$$



Nous obtenons ici le modèle le plus compact qui ne comporte plus que $\boxed{845}$ opérations.

Modèle augmenté Le modèle augmenté comporte $\boxed{1232}$ ou $\boxed{1303}$ opérations selon le choix des points de référence pour la coupure de la plate-forme.

Ceci est un peu plus important que dans le cas du modèle précédent et s'explique probablement par la taille plus élevée de la matrice de masse augmentée, consécutive à l'ajout des trois coordonnées absolues et d'une coupure supplémentaire. Néanmoins on constate à nouveau que le modèle augmenté est plus lourd que le modèle réduit obtenu avec la "plus mauvaise" partition.

Observations L'introduction de la chaîne cinématique supplémentaire permet donc d'obtenir des modèles plus compacts, magré l'ajout des coordonnées et de contraintes supplémentaires. Toutefois, cette technique nécessite d'effectuer un choix de partition de variables cohérent avec l'introduction des nouvelles coordonnées, qu'elles soient absolues - dans ce cas-ci - ou intermédiaires dans d'autres cas. On a également vu que les contraintes relatives à une même coupure peuvent être découplées pour un choix judicieux du point de référence commun pour les corps original et secondaire obtenus par la coupure d'un corps.

Modèle 4

Avant de conclure l'analyse de ce système, nous proposons un dernier modèle obtenu en coupant les articulations qui relient les bras à la plate-forme.

En effet, lors de la modélisation d'un système, il n'est pas toujours nécessaire de connaître les valeurs de toutes les coordonnées articulaires et donc pas forcément nécessaire de conserver toutes les articulations dans le modèle.

Le système ouvert contient donc moins de coordonnées articulaires et moins de contraintes. Les coupures d'articulations ne génèrent chacune que deux contraintes de translation, dans le plan. Il est donc évident à priori que cette démarche va produire un modèle dynamique plus économique.

Le modèle est illustré sur la figure 2.15, ainsi que la numérotation des articulations. On conserve l'utilisation de la chaîne fictive pour le découplage des contraintes.

La matrice Jacobienne des contraintes possède alors la structure suivante, similaire à celle du modèle précédent.

$$Jac = \left(\begin{array}{cc|cc|cc|ccc} \otimes & \otimes & \cdot & \cdot & \cdot & \cdot & \otimes & \cdot & \otimes \\ \otimes & \otimes & \cdot & \cdot & \cdot & \cdot & \cdot & \otimes & \otimes \\ \hline \cdot & \cdot & \otimes & \otimes & \cdot & \cdot & \otimes & \cdot & \otimes \\ \cdot & \cdot & \otimes & \otimes & \cdot & \cdot & \cdot & \otimes & \otimes \\ \hline \cdot & \cdot & \cdot & \cdot & \otimes & \otimes & \otimes & \cdot & \otimes \\ \cdot & \cdot & \cdot & \cdot & \otimes & \otimes & \cdot & \otimes & \otimes \end{array}\right)$$

Partition 1 On choisit les coordonnées absolues de la plate-forme comme coordonnées indépendantes. On obtient alors la matrice J_v suivante :

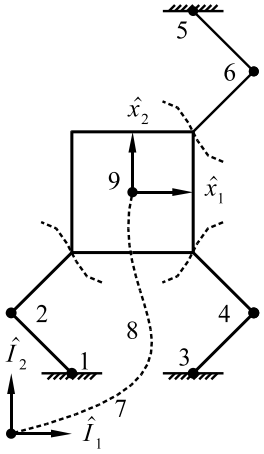
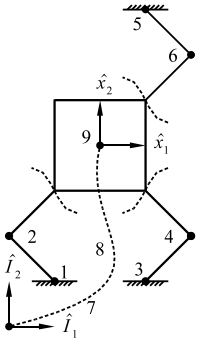


FIG. 2.15 – Modèle 4

$$q_u = \{\ q_7\ \ q_8\ \ q_9\ \}$$
$$q_v = \{\ q_1\ \ q_2\ \ q_3\ \ q_4\ \ q_5\ \ q_6\ \}$$
$$J_v = \begin{pmatrix} \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \otimes & \otimes & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \otimes & \otimes & \cdot & \cdot \\ \cdot & \cdot & \otimes & \otimes & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \otimes & \otimes \\ \cdot & \cdot & \cdot & \cdot & \otimes & \otimes \end{pmatrix}$$



On retrouve une structure bloc diagonale qui indique le découplage des équations de contraintes relatives aux trois coupures. Le modèle obtenu ne contient alors plus que $\boxed{772}$ opérations.

Modèle augmenté Le modèle augmenté compte toujours $\boxed{900}$ opérations.

2.7.2 Conclusions

L'analyse de la modélisation de ce système relativement simple révèle les nombreuses possibilités offertes pour la génération d'un modèle de simulation, et leurs conséquences sur la taille du modèle dynamique direct obtenu. On a observé une grande différence entre le meilleur modèle réduit présenté, qui

comptait 772 opérations et le plus mauvais, qui en comptait 1704.

En réalité, il est possible de réduire le nombre d'opérations à moins de 500 en utilisant toutes les possibilités offertes par le logiciel ROBOTRAN en termes de conventions, de formalismes et de manipulations symboliques.

Une bonne analyse du système et l'utilisation judicieuse des conventions de modélisation et des formalismes proposés sont les facteurs les plus important pour obtenir le modèle de simulation le plus efficace.

Ceci implique l'ingénieur qui modélise le système, car lui seul connaît ses besoins et est en position de faire le meilleur choix parmi toutes les possibilités de modélisation du système et de génération de modèles qui sont offertes par son logiciel d'analyse.

Au travers des différents modèles présentés, nous avons pu mettre en évidence quelques indices pour obtenir un modèle le plus compact possible :

1. placer les coupures des boucles cinématiques de manière à minimiser la longueur des branches du système arborescent ainsi obtenu,
2. introduire des chaînes cinématiques fictives de manière à découpler les contraintes, même si cela correspond à ajouter des coordonnées absolues et des contraintes supplémentaires,
3. couper des articulations qui ne sont pas indispensables plutôt que des corps pour limiter le nombre de contraintes,

Ces conseils sont importants : si une variation d'un facteur 3 ou 4 n'est pas problématique pour un petit modèle plan qui ne compte que quelques centaines d'opérations, cela peut devenir un élément clé pour l'analyse d'un système complexe qui peut comporter quelques dizaines de milliers d'opérations, notamment lorsqu'on désire utiliser le modèle dans un contexte de simulation en temps réel. Le logiciel ROBOTRAN peut alors se révéler un outil très performant qui permet à l'utilisateur averti d'obtenir des modèles vraiment très efficaces.

2.8 Applications

2.8.1 Une voiture : l'Audi A6

Cet exemple de système multicorps est représentatif d'une catégorie de systèmes correspondant aux véhicules automobiles actuels. On trouve dans la littérature récente, des comptes-rendus de travaux [107, 108, 109] portant sur des techniques d'élaboration de modèles et d'intégration numérique de ces modèles dans le but de développer des systèmes d'assistance à la conduite, par exemple.

Les véhicules modélisés dans ces rapports sont similaires à l'exemple que nous présentons ici. Toutefois, les modèles utilisés pour la simulation en temps réel ne sont pas toujours équivalents. Ces équipes d'ingénieurs ont généralement tendance à utiliser des modèles simplifiés pour la simulation dans le contexte

temps réels [109]. Des modèles complets et plus fidèles, n'offrant pas la possibilité de simulation en temps réel, sont alors utilisés pour la validation de modèles simplifiés. Certains atteignent des performances compatibles avec la simulation en temps réel avec des modèles complets lorsque le véhicule est pourvu de suspensions dont la structure est relativement simple [108].

Caractéristiques du modèle de l'Audi A6

Cette voiture est équipée de suspensions multi-points comme illustré sur la figure 2.16. Le porteur de roue est relié au châssis par quatre barres de suspension. Une cinquième barre sert pour la direction à l'avant et pour le maintien de l'orientation de la roue à l'arrière.

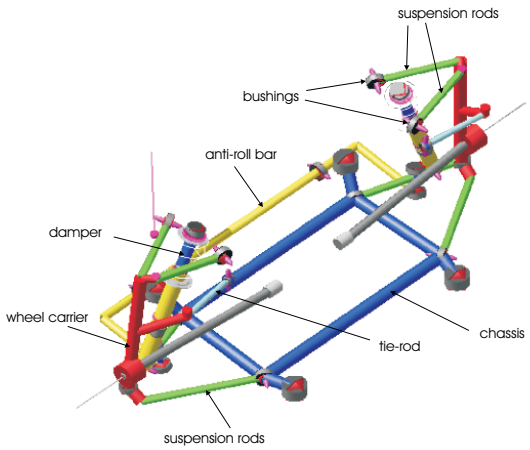


FIG. 2.16 – Suspensions de l'Audi A6

Chaque suspension comporte 12 articulations y compris celles de rotation de la roue. Quatre coupures permettent de transformer la structure bouclée de chaque suspension en structure arborescente. Nous utilisons trois coupures d'articulations sphériques et une coupure de type bielle pour la barre de direction, ce qui correspond à retirer ce corps du modèle. Les arbres de transmission pour les roues avant ainsi que les dispositifs anti-roulis ne sont pas inclus dans le modèle. Les efforts appliqués par ces éléments sur les différentes parties du véhicule sont toutefois pris en compte. Dix équations de contraintes sont alors générées pour chaque suspension.

En tenant compte des 6 degrés de liberté du châssis, notre modèle comporte alors

- 54 variables généralisées,

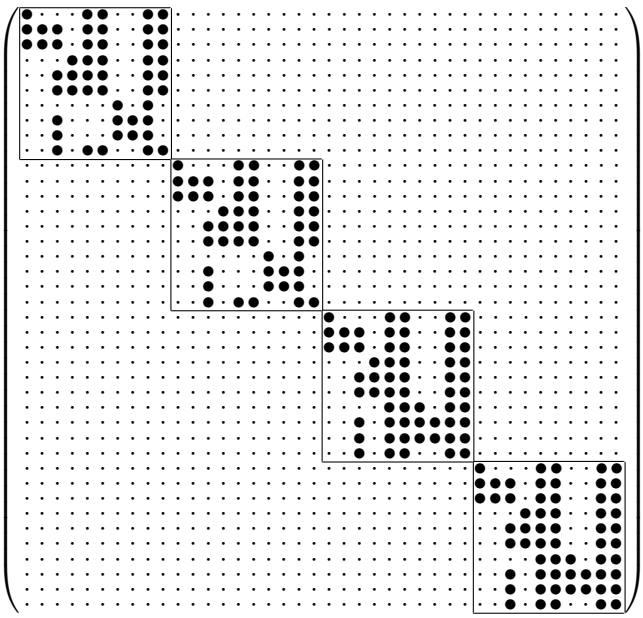
- 14 degrés de liberté (6 pour le châssis + 4 pour les suspensions + 4 pour la rotation des roues)
- 40 contraintes scalaires indépendantes et donc 40 coordonnées dépendantes.

Trente coordonnées relatives supplémentaires, bloquées à des valeurs constantes selon les conventions de modélisation de ROBOTRAN [12, 13], sont utilisées pour tenir compte des orientations des axes des articulations des différents éléments de suspension.

Le partitionnement des coordonnées n'a pas d'influence décisive sur la structure de la Jacobienne des contraintes J_v associées aux variables dépendantes. En effet, quelle que soit la partition envisagée, on obtient toujours une matrice J_v constituée de quatre blocs diagonaux, correspondant aux contraintes des quatre suspensions. Le caractère diagonal de la structure s'explique par l'absence de couplage cinématique entre les différentes suspensions du véhicule.

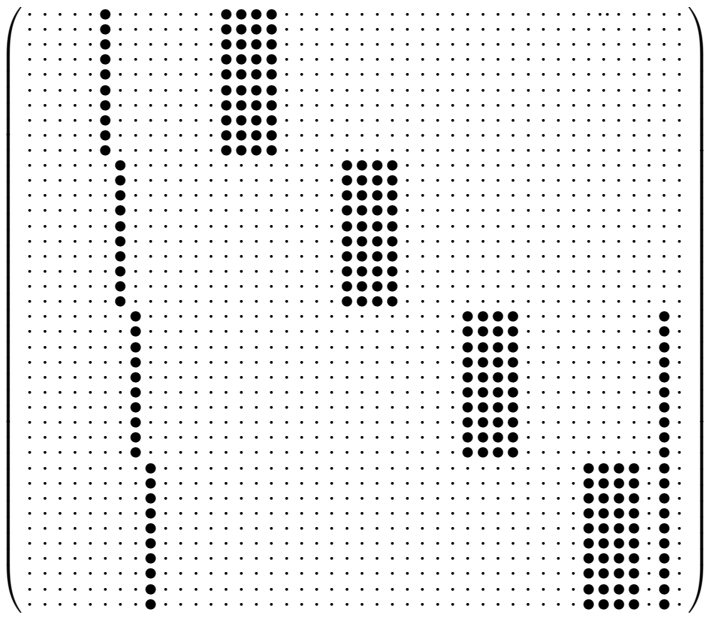
Nous n'avons pas pu découpler la résolution des contraintes au sein de chaque suspension[10].

Pour un tel système, le partitionnement des coordonnées peut être effectué manuellement sans difficultés. L'analyse du conditionnement numérique des sous-blocs confirme que les coordonnées qui correspondent aux angles d'élévation des barres peuvent être choisies indifféremment comme coordonnées indépendantes pour chaque suspension. Ces choix sont d'ailleurs assez intuitifs.

$J_v =$ 

[10] ce qui est caractéristique d'une suspension multi-points

On retrouve cette structure bloc diagonale de J_v pour la plupart des modèles de voiture à suspensions indépendantes. Elle a une influence intéressante sur la matrice de couplage des vitesses B_{vu} qui se trouve être très creuse.

$B_{vu} =$ 

Ceci permet d'expliquer l'économie de calcul importante réalisée dans l'étape de réduction des équations du mouvement. Le gain, repris dans la table 2.1, est estimé à 41 par rapport à une approche purement numérique qui ne tiendrait pas compte de la structure creuse des matrices de couplage B_{vu} et de masse M.

Le modèle réduit complet compte de l'ordre de $\boxed{35500}$ opérations. Par comparaison, un modèle augmenté tel que présenté dans la section 2.5.3 comporte environ $\boxed{109000}$ opérations, soit trois fois plus. Ceci s'explique facilement en considérant la différence de taille entre les matrices augmentée et réduite.

La génération du modèle symbolique réduit est effectuée en 7 secondes[11] par ROBOTRAN sur un ordinateur personnel standard équipé d'un processeur Intel Pentium 4 centrino cadencé à 1.6 GHz.

Les nombres d'opérations relatives aux différentes parties du modèle sont repris dans la table suivante.

[11] Ce qui est une performance appréciable en comparaison de celles des outils de génération basés sur des logiciels symboliques commerciaux

Groupe	Equations	Opérations
1	Résolution des contraintes en position	2973
2	Résolution des contraintes en vitesse	538
3	Résolution des contraintes en accélération	1857
4	Calcul de la cinématique et des forces dans les suspensions	438
5	Calcul de la cinématique et des forces de contact roues/sol	1707
6	Calcul récursif direct de la cinématique absolue des corps	9215
7	Calcul récursif inverse de la dynamique arborescente	15666
8	Réduction des équations du mouvement par élimination des λ	1897
9	Résolution du système d'équations différentielles ordinaires	1185
Total		35555

Chaque groupe d'équations correspond à un bloc du schéma de la figure 2.9. On constate que les équations récursives, groupes 6 et 7, qui servent à calculer les termes de la matrice de masse M et du vecteur c représentent la plus grosse partie du modèle généré par calcul symbolique, pour ce système.

Le nombre d'opérations du modèle réduit peut être diminué encore en invoquant les procédure d'optimisations symboliques poussées présentées dans la section 1.6. La réduction obtenue est de l'ordre de 17%, ce qui amène le nombre d'opérations du modèle à un peu moins de 30000.

Le temps nécessaire à cette optimisation poussée est alors d'environ 6 minutes, soit dix fois plus que pour une génération classique. Toutefois, ce temps est encore négligeable dans le processus complet de modélisation et d'analyse numérique d'un système multicorps tel que celui-ci.

L'utilisation d'un jeu minimal de paramètres barycentriques [21] pour la génération récursive de la matrice de masse permettrait de réduire encore cette valeur de près de 4000 opérations.

Simulation du modèle

La génération symbolique des équations du mouvement de systèmes multicorps permet d'obtenir les modèles sous forme de sous-routines implémentées en un langage de programmation standard, C, FORTRAN, MATLAB, JAVA, etc. Ceci permet d'exploiter le modèle dans n'importe quel environnement d'analyse numérique.

Dans notre laboratoire, nous utilisons l'environnement MATLAB pour les premières analyses. Le modèle désiré, est alors généré sous forme de fonction

dans la syntaxe Matlab et peut être utilisé pour en effectuer l'analyse préliminaire au partitionnement des coordonnées par exemple.

Pour les analyses un peu plus intensives du point de vue calcul, nous générons le modèle, la dynamique directe par exemple, sous forme de fonction en langage C que nous compilons pour obtenir une librairie qui pourra être utilisée comme *S-function*[12] dans SIMULINK, un outil de l'environnement MATLAB dédicacé à la simulation des systèmes et très largement utilisé en ingénierie.

La figure 2.17 illustre un tel schéma de simulation sous SIMULINK. Le modèle dynamique direct de notre voiture Audi A6, qui a été généré symboliquement par ROBOTRAN puis compilé, est contenu dans la boite centrale. Nous avons effectué la simulation d'un double changement de bande. En utilisant une méthode d'intégration de type Runge-Kutta d'ordre 4-5 à pas variable[13], le temps de calcul d'une manoeuvre de 20 secondes prend environ 3 secondes sur un ordinateur personnel standard équipé d'un processeur Intel Pentium 4 centrino cadencé à 1.6 GHz. Bien que le contexte de simulation ne soit pas temps réel, on remarque que le temps de calcul est six fois moindre que le temps simulé.

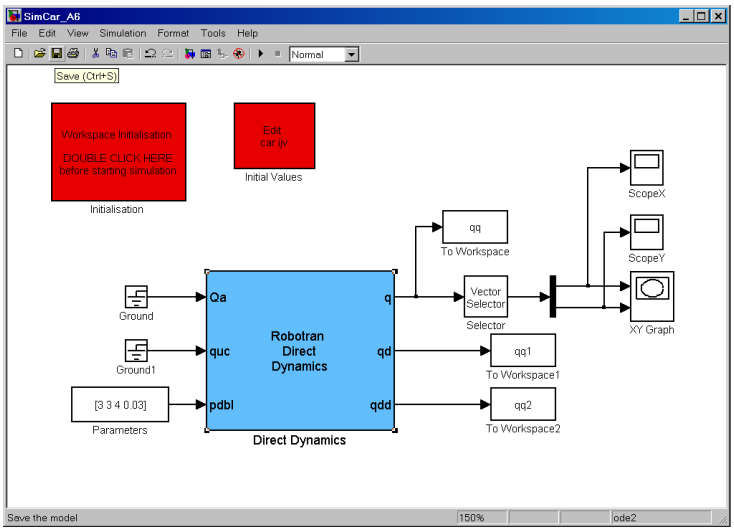


FIG. 2.17 – Schéma de simulation sous MATLAB/SIMULINK

Pour effectuer des analyses vraiment très exigeantes en terme de temps calcul, telles que des optimisations de paramètres ou des simulations en temps réel, nous générons un code directement exécutable, sous Windows ou sous

[12] une des interfaces proposées pour incorporer des gros modèles de système dans un schéma de simulation sous SIMULINK

[13] Les pas de temps ont une valeur moyenne de l'ordre de 20 millisecondes.

Linux, à partir des modèles générés par ROBOTRAN en langage C. Nous obtenons ainsi des performances calcul supérieures à celles qui sont possibles dans l'environnement SIMULINK.

En utilisant une méthode d'intégration de type Runge-Kutta d'ordre 4 avec un pas de temps fixe de 1 milliseconde, notre simulation d'une manoeuvre de 20 secondes est calculée en 10 secondes[14] sur le même ordinateur. Cela représente 80000 évaluations du modèle, soit un temps d'exécution du modèle d'environ 0.125 millisecondes.

Ce résultat montre que l'approche symbolique utilisée confère à des modèles complets de véhicules complexes, les performances requises pour une utilisation dans un contexte temps réel où, jusqu'il y a peu, les ingénieurs utilisaient systématiquement des modèles simplifiés. Notre approche permet donc d'utiliser directement dans le contexte temps réel les modèles qui n'étaient habituellement utilisés que pour la validation des modèles simplifiés, ou simplement d'obtenir un gain significatif sur le temps calcul des simulations de la dynamique de systèmes multicorps.

2.8.2 Un bogie ferroviaire articulé : Caracas

Nous terminons cette section par un dernier exemple. Il s'agit d'un bogie ferroviaire utilisé dans [110] pour illustrer la modélisation d'un système électromécanique. Ce bogie est développé par la société Bombardier Transport pour le métro de la ville de Caracas au Vénézuela. Il est illustré sur la figure 2.18.

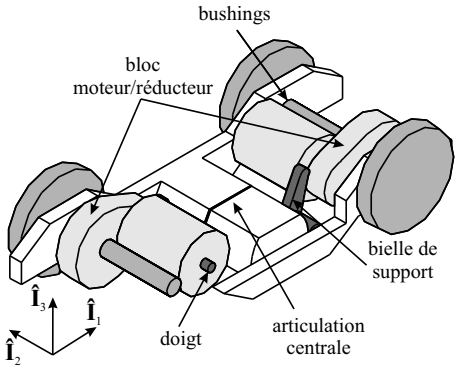


FIG. 2.18 – Schéma du bogie Caracas

Ce bogie a la particularité d'avoir un châssis articulé. Les parties gauche et droite peuvent pivoter l'une par rapport à l'autre autour d'un axe transversal

[14] Ce qui correspond à une vitesse de calcul près de six fois supérieure à celle atteinte sous SIMULINK, en tenant compte des valeurs des pas de temps respectifs

et ainsi présenter une inclinaison différente. Ceci permet aux roues de rester en contact avec les rails lorsque ceux-ci présentent d'importants défauts de planéité. Ce bogie est équipé de deux moteurs asynchrones munis d'un réducteur pour l'entraînement des essieux avant et arrière. L'ensemble est assez complexe comme le montre le schéma du système extrait de [110] et reproduit sur la figure 2.19.

On peut observer sur le schéma de la figure 2.19 que quatre coupures d'articulations sphériques, notées $C2$, ont été utilisées pour obtenir une topologie arborescente à partir de la structure bouclée du système original. En plus des 12 équations de contraintes générées par les coupures, six contraintes externes sont utilisées. Celles-ci sont implémentées dans des sous-routines par l'utilisateur.

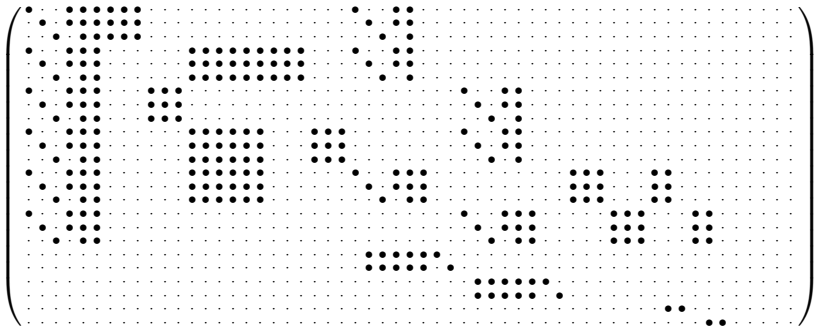
- quatre contraintes expriment les conditions géométriques de contact des roues avec les rails,
- deux contraintes expriment les rapports de réduction des boites de transmission avant et arrière.

Le système est donc soumis à 24 équations de contraintes algébriques. Outre ces contraintes scléronomes, on impose une valeur nulle à 4 articulations parmi les 57 qui sont utilisées pour décrire la topologie arborescente du système. Ces quatre articulations commandées correspondent aux rotations des roues par rapport à l'axe de l'essieu sur lequel elles sont montées. Comme il s'agit d'essieux rigides, les rotations relatives des roues sont bloquées à une valeur nulle.

Finalement, le système ne possède plus que 29 degrés de liberté.

Matrice Jacobienne

Avant toute permutation, la matrice Jacobienne J des 24 contraintes possède la structure suivante :

$J =$ 

Afin de résoudre ces 24 équations de contrainte, et de réduire les équations du mouvement, nous utilisons la technique du partitionnement des coordonnées. Comme nous l'avons mentionné dans la section 2.3, une approche numérique

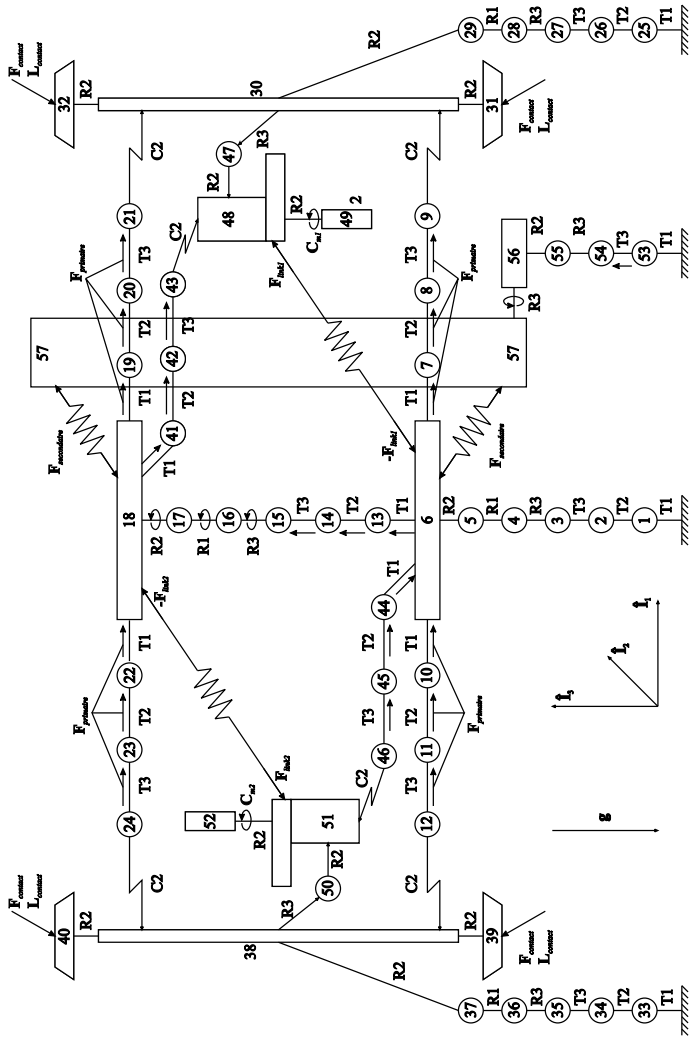


FIG. 2.19 – Schéma du modèle de la structure du bogie

peut être utilisée pour déterminer les coordonnées indépendantes et dépendantes. L'objectif numérique est de trouver une sous-matrice J_v carrée de plein rang avec le meilleur nombre condition possible. Les variables associées aux colonnes de J choisies pour constituer cette sous-matrice définissent alors la partition des variables dépendantes q_v.

Nous avons déjà montré l'influence de la structure de la sous-matrice J_v dans les exemples présentés ci-avant. Il peut donc s'avérer intéressant d'utiliser une méthode symbolique pour effectuer le partitionnement des coordonnées, tout en veillant, a posteriori, au bon conditionnement numérique de J_v.

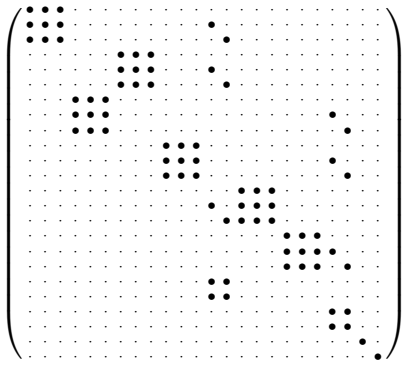
Partitions

Nous présentons ici deux partitions valides pour ce modèle. La première est obtenue à l'aide de la méthode présentée à la section 2.3.4. La seconde partition est obtenue en modifiant manuellement la première. Nous mettons en évidence qu'une petite différence entre deux partitions peut avoir une conséquence importante sur la quantité d'opérations des modèles générés.

Partition 1 Cette première partition est obtenue en appliquant la procédure proposée à la section 2.3.4. On sélectionne les coordonnées suivantes comme dépendantes :

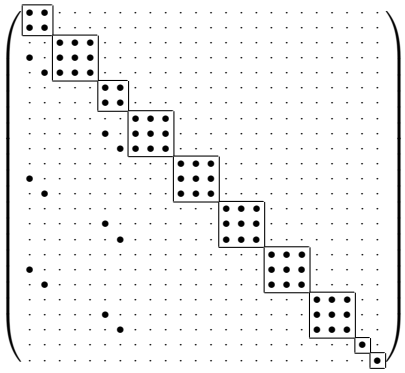
q_v = { 7, 8, 9, 10, 11, 12, 19, 20, 21, 22, 23, 24, 26, 27, 34, 35, 41, 42, 43, 44, 45, 46, 49, 52 }

La matrice Jacobienne J_v associée aux coordonnées dépendantes dont les indices sont donnés ci-dessus possède alors la structure que voici :

$J_v =$ 

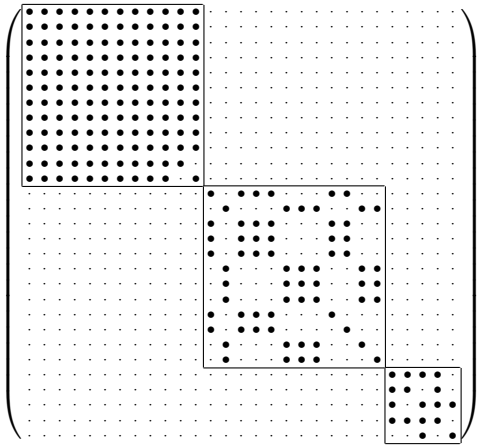
Après factorisation et permutations adéquates des lignes et des colonnes, on

peut observer la structure bloc triangulaire suivante :

$J_v^{btf} =$ 

On remarque que la matrice J_v^{btf} contient dix blocs diagonaux et qu'il y a très peu d'éléments non nuls, seulement 12, dans la partie inférieure.

En utilisant cette partition, la matrice de masse réduite M_r du système possède une structure bloc diagonale, ce qui permet d'économiser des opérations dans la phase de résolution également.

$\mathcal{M}_r =$ 

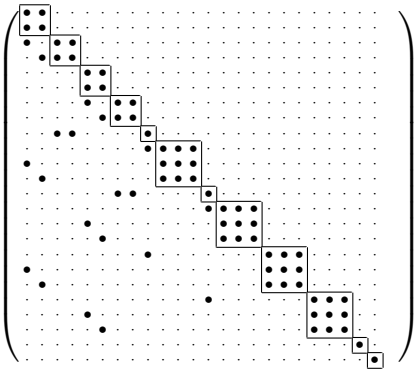
Le modèle généré avec cette partition comporte un peu plus de 12554 opérations en virgule flottante.

Partition 2 Une légère modification de la première partition va avoir un effet important et un peu paradoxal sur le nombre d'opérations du modèle.

Ainsi, en prenant l'ensemble suivant des coordonnées dépendantes

q_v = { 7, 8, 9, 10, 11, 12, 19, 20, 21, 22, 23, 24, 25, 26, 27, 33, 34, 35, 41, 42, 44, 45, 49, 52 } où les coordonnées 25 et 33 ont été choisies en lieu et place

des coordonnées 43 et 46 dans la première partition, on obtient une matrice factorisée J_v^{btf} qui possède cette fois 12 blocs diagonaux.

$$J_v^{btf} = \left(\right.$$

$$\left. \right)$$

Cela correspond à une décomposition plus fine de l'ensemble des équations de contraintes.

Cependant, la partie inférieure contient cette fois 20 éléments non nuls, soit tout de même près du double par rapport à la première partition.

Le nombre d'opérations total du modèle s'élève cette fois à plus de 20500.

D'autres partitions différentes peuvent conduire à des modèles contenant plus de 35000 opérations, soit près de trois fois plus que le nombre obtenu en utilisant la première partition ci-dessus.

2.9 Conclusions

Dans ce chapitre, nous avons présenté une méthode de modélisation des systèmes contenant des *boucles cinématiques*. Une *technique de coupure* est utilisée pour obtenir une *structure arborescente*, ce qui nous permet d'utiliser les formalismes récursifs présentés dans le chapitre précédent. Chaque coupure engendre l'écriture d'une ou plusieurs *équations de contraintes* qui expriment les *conditions géométriques* de fermeture des boucles. Les *forces de liaison* entre les corps séparés par les coupures sont prises en compte par le biais de la matrice Jacobienne et de *multiplicateurs de Lagrange* associés aux contraintes. Le système d'équations obtenu est alors composé d'*équations différentielles et algébriques*.

La technique du *partitionnement des coordonnées* est mise en oeuvre pour déterminer un ensemble de *coordonnées dépendantes* qui seront calculées en fonction des coordonnées indépendantes *par résolution des équations de contraintes*, ainsi que de leurs dérivées premières et secondes. Les multiplicateurs de Lagrange sont éliminés des équations à l'aide des valeurs des coordonnées dépendantes. Le système d'équations du mouvement est réduit en un *système d'équations purement différentielles*.

L'implémentation de cette technique dans le logiciel ROBOTRAN permet de réduire considérablement le nombre d'opérations nécessaire à l'évaluation des modèles lorsque ceux-ci sont obtenus par *génération symbolique*. Les simplifications automatiques effectuées lors de la génération symbolique des équations du modèle permettent de tirer profit de la structure creuse des matrices de masse et de couplage des vitesses, ce qui conduit à une *diminution conséquente du nombre d'opérations arithmétiques*, par rapport à une génération numérique, nécessaires à l'obtention des équations réduites.

Une analyse de la structure de la matrice Jacobienne et un partitionnement adéquat des coordonnées, permettent de mettre en évidence une *structure bloc triangulaire* dans la sous-matrice Jacobienne associée aux coordonnées dépendantes, ce qui rend possible un *découplage au niveau de la résolution des équations de contraintes*. Ce découplage est favorable à la résolution des équations de contraintes par une méthode itérative et permet d'obtenir une matrice de couplage des vitesses ayant une structure creuse.

Cependant, les avantages de l'utilisation de la technique du partitionnement des coordonnées pour la génération symbolique des équations du mouvement de systèmes contenant des boucles cinématiques, dépendent précisément du partitionnement des coordonnées.

Notre travail nous a permis de dégager les caractéristiques d'une bonne disposition des coupures ainsi que d'une bonne partition de coordonnées du point de vue de la génération symbolique. Nous proposons une procédure symbolique pour la sélection automatique des coordonnées dépendantes. Cette procédure est utilisée pour les systèmes complexes lorsque la partition ne peut pas être effectuée facilement "à la main". Les partitions obtenues manuellement ou à l'aide de cette procédure symbolique doivent être validées en vérifiant le conditionnement numérique de la matrice J_v engendrée. Il est également recommandé d'appliquer une technique de pivotement aux blocs diagonaux de J_v afin de déterminer des permutations correctes des lignes et des colonnes de ceux-ci avant la génération du modèle complet.

Le processus de coupure des boucles cinématiques conditionne également la matrice Jacobienne des contraintes. La mise en place d'une méthode automatique de coupure nous paraît également importante pour libérer l'utilisateur de cette étape décisive du processus de modélisation. Ceci requiert toutefois une modification des conventions de description des systèmes actuellement utilisées pour la modélisation avec ROBOTRAN.

Toutefois, nous n'avons pas cherché à développer une méthode automatique de coupure et de partitionnement, conciliant à la fois les critères numériques et symboliques. Il s'agit là d'un objectif important que nous jugeons nécessaire de poursuivre lors de recherches ultérieures.

Finalement, la génération symbolique nous permet d'obtenir les modèles des systèmes sous la forme de sous routines implémentées selon la syntaxe du langage de programmation choisi par l'utilisateur. Ces modèles sont indépendants

du logiciel ROBOTRAN et peuvent donc être exploités facilement dans n'importe quel environnement couramment utilisé par les ingénieurs pour l'analyse numérique du système multicorps étudié.

Deuxième partie

Méthodes de Parallélisation des modèles de systèmes multicorps

Dans cette seconde partie, nous allons présenter deux méthodes de parallélisation qui mettent à profit la connaissance de l'interdépendance des équations.

Le problème de la parallélisation d'un modèle dynamique de système articulé peut-être considéré à plusieurs niveaux, depuis l'exécution des opérations arithmétiques jusqu'à l'évaluation de modèles de sous-systèmes indépendants.

La première méthode proposée est originale et très différente de la parallélisation classique. Elle consiste à travailler au niveau de la représentation des équations générées ou plus précisément de la représentation des expressions symboliques. Il s'agit d'une méthode de parallélisation à grains fins, que nous appelons *vectorisation* dans ce texte. Cette méthode est présentée dans le chapitre 3. Elle a l'avantage de permettre d'extraire un taux de parallélisme important de n'importe quel ensemble d'équations générées pour n'importe quel système articulé, quelle qu'en soit la topologie, et quel que soit le formalisme utilisé. En revanche, pour exploiter efficacement ces modèles vectorisés, il est nécessaire de disposer d'un processeur ayant une architecture parallèle spécifique.

Dans le chapitre 4 nous présentons une méthode de parallélisation automatique qui ne nécessite pas l'intervention de l'utilisateur, ni au niveau de la génération des différentes parties du modèle, ni au niveau de leur assemblage. Elle ne nécessite en outre pas que l'algorithme de génération soit implémenté de manière parallèle.

Généralement, la topologie du système et la structure algorithmique du modèle sont à la base du procédé de parallélisation de celui-ci. Si le système est grand, complexe et composé de sous-systèmes, on peut générer les équations des sous-systèmes et évaluer celles-ci plus ou moins indépendamment selon les interactions présentes entre les sous-systèmes. Cette approche a été présentée dans [64] et se révèle très efficace, toutefois, elle nécessite que l'utilisateur décrive explicitement les différents sous-systèmes, et qu'il assemble manuellement les modèles obtenus séparément pour chacun d'entre eux.

La méthode que nous proposons utilise l'analyse topologique du système et la structure algébrique de la matrice jacobienne des contraintes pour séparer les équations. Elle n'a toutefois été appliquée que pour la parallélisation des modèles dynamiques directs utilisés pour la simulation.

Chapitre 3

Vectorisation des équations

Cette méthode originale est issue d'une observation, présentée brièvement dans [12] et [64], et relative à l'interdépendance des équations récursives générées symboliquement. On constate dans [12] qu'une équation récursive ne dépend pas toujours de l'équation qui la précède directement dans la liste. En d'autres termes, l'expression qui définit une variable intermédiaire, n'implique pas systématiquement la variable intermédiaire précédente de la liste, ce qui permet donc de calculer ces deux variables simultanément, en parallèle, comme illustré sur la figure 3.1.

$$\begin{array}{l} X = A + B \\ Y = C \cdot D \\ Z = X \cdot E \\ T = X + Y \end{array} \quad \longmapsto \quad \begin{array}{ll} X = A + B & ;\ Y = C \cdot D \\ Z = X \cdot E & ;\ T = X + Y \end{array}$$

FIG. 3.1 – Indépendance de certaines équations

Cette constatation est alors exploitée pour produire des schémas de calculs dans lesquels les équations indépendantes sont détectées et regroupées étapes par étapes en tenant compte des interdépendances récursives, pour être assemblées en une structure matricielle d'instructions. Chaque ligne contient alors un vecteur de calculs indépendants correspondant aux expressions constituant les membres de droite des équations indépendantes de la liste.

Un taux de parallélisme et une réduction du nombre d'étapes de calcul très importants sont observés : des valeurs très supérieures à dix sont avancées dans [64]. Néanmoins, d'une part, aucune application concrète n'est proposée pour tirer réellement profit du taux de parallélisme annoncé, et d'autre part, la

distribution des calculs est très inégale car les membres de droite des équations, qui ont des tailles et des complexités très différentes, sont utilisés tels quels comme "instructions élémentaires" dans le schéma de calcul vectoriel.

La motivation initiale à la base de la poursuite de ces investigations dans cette thèse, c'est-à-dire du développement de cette technique de parallélisation fine, ou *vectorisation*, des modèles de systèmes multicorps, est de tenter de tirer profit du taux de parallélisme élevé et prometteur d'une accélération importante des calculs.

Pour exploiter efficacement le taux de parallélisme important de ces schémas de calculs, il est nécessaire de disposer d'un processeur ayant une architecture particulière de type vectorielle par exemple.

Toutefois, le développement complet d'une telle architecture est un projet complexe qui aurait nécessité de s'y consacrer exclusivement, ce qui n'était pas compatible avec les autres objectifs de ce travail. C'est pourquoi nous avons utilisé un prototype d'architecture dataflow développé par Jean-Pierre David au cours de sa thèse de doctorat [59].

Les résultats de simulation sur FPGA du modèle dynamique inverse d'un robot manipulateur série, qui sont présentés dans ce texte, ont été publiés dans [58] et [59].

3.1 Traitements préliminaires des équations

Comme exposé brièvement ci-dessus, la démarche initiale présentée dans [64] consistait à aligner à chaque étape de calcul, toutes les équations qui pouvaient être évaluées indépendamment en utilisant les données et les variables auxiliaires déjà calculées, ce qui conduisait à un schéma de calcul vectoriel très déséquilibré et dont la longueur, le nombre d'étapes, ne correspondait à rien de très précis, vu la grande différence de complexité des équations récursives.

3.1.1 Décomposition

Afin d'éviter cet inconvénient, nous allons travailler au niveau élémentaire des opérations arithmétiques. Nous décomposons les membres de droite des équations en créant de nouvelles variables intermédiaires comme expliqué dans la section 1.6.2.

Ainsi, les équations que voici

```
BS91 = -qp(1)*qp(1);
ALPHA21 = g(3)*S1;
ALPHA31 = g(3)*C1;
ALPHA22 = C2*(ALPHA21-qpp(1)*D(3,2))+S2*(ALPHA31+BS91*D(3,2));
```

sont décomposées pour produire la série d'équations suivante :

```
1  BS91d = qp(1) * qp(1);
2  ALPHA21 = g(3) * S1;
3  ALPHA31 = g(3) * C1;
4  ALPHA22gdd = qpp(1) * D(3,2);
5  ALPHA22ddd = BS91d * D(3,2);
6  ALPHA22gd = ALPHA21 - ALPHA22gdd;
7  ALPHA22dd = ALPHA31 - ALPHA22ddd;
8  ALPHA22g = C2 * ALPHA22gd;
9  ALPHA22d = S2 * ALPHA22dd;
10 ALPHA22 = ALPHA22g + ALPHA22d;
...
```

où les membres de droite se réduisent aux opérations arithmétiques de base sous forme binaire.

Les équations décomposées sont alors analysées de manière à identifier puis éliminer les variables auxiliaires définies comme l'opposé d'une autre variable ou d'une donnée, ainsi que les *doublons*, c'est-à-dire les variables auxiliaires définies par des expressions binaires identiques, et donc redondantes. Ces deux traitements ont déjà été présentés dans la section 1.6.2 relative aux traitements symboliques effectués après la génération de l'ensemble complet des équations.

Remarques Comme nous l'avons présenté dans la section 1.6.1, les expressions symboliques générées peuvent contenir des appels de fonctions externes ou des opérations unaires, comme les fonctions trigonométriques. Ces fonctions impliquent évidemment des calculs plus complexes que les opérations arithmétiques.

Un indice de complexité peut-être associé à chaque opération pour tenir compte du nombre de cycles nécessaires à son exécution. Grâce aux extensions implantées dans ROBOTRAN, toute l'information sur le chaînage des expressions est disponible et permet donc d'effectuer l'analyse d'interdépendance nécessaire à l'application de la méthode de vectorisation.

Néanmoins, celle-ci a été développée historiquement au début de ce travail et ne s'appliquait alors qu'à des schémas d'équations ne contenant que des opérations arithmétiques de base. Elle n'a été validée que dans le contexte plus restreint, des modèles dynamiques directs ou inverses de systèmes robotiques simples à structure linéaire ou arborescente.

3.1.2 Indices d'ordre d'exécution

La figure 3.2 montre une représentation arborescente de l'évaluation de la série d'équations présentée ci-dessus. La notion de *chemin critique*, présentée ci-après y apparaît sous forme de tracé pointillé.

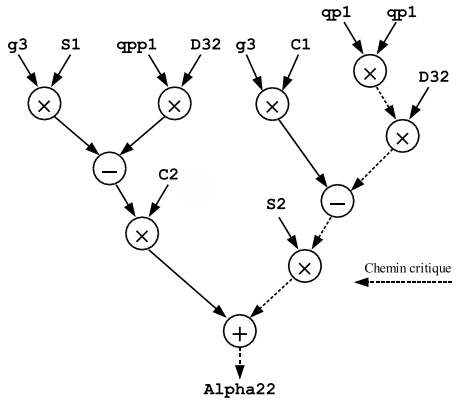


FIG. 3.2 – Calcul d'une série d'opérations

Chaque variable auxiliaire, chaque équation décomposée, reçoit deux attributs déterminé en fonction des interdépendances des équations. Ces attributs sont utilisés pour organiser l'ensemble des opérations à effectuer selon une grille d'exécution vectorielle.

Le premier attribut, appelé indice `asap`[1], correspond à l'étape d'exécution d'une opération de telle sorte que celle-ci soit calculée *le plus tôt possible*. Ainsi, une variable auxiliaire qui ne dépend que des données ou des coordonnées généralisées peut être calculée d'emblée, à la première étape vectorielle : on lui attribue un indice `asap` de valeur 1.

Si on considère que tous les paramètres du modèles, i.e. les données géométriques et dynamiques, les coordonnées généralisées indépendantes, etc., reçoivent un indice `asap` de valeur 0, on peut calculer récursivement la valeur de l'indice `asap` de chaque variable intermédiaire en incrémentant la valeur du plus grand indice `asap` des variables utilisées pour la calculer. L'attribution des valeurs des indices `asap` se fait en parcourant la liste des équations en commençant par la première et en progressant vers la dernière.

L'indice `asap` de valeur maximale fournit alors la *longueur du chemin critique*. Il s'agit du nombre minimal d'étapes nécessaires pour évaluer l'ensemble des opérations, en supposant que l'on dispose de suffisamment d'unités de calcul en parallèle à chaque étape, et que chaque opération peut être calculée en une seule étape.

Le second attribut, appelé indice `alap` [2] correspond à la dernière étape à laquelle l'opération pourra être effectuée, donc de telle sorte que chaque variable intermédiaire soit calculée *le plus tard possible*.

[1] As Soon As Possible
[2] As Late As Possible

Cet indice est déterminé en appliquant le raisonnement opposé : Les variables auxiliaires qui ne sont utilisées par aucune autre variable, en d'autres mots les résultats, peuvent être calculées à la dernière étape.

Les valeurs des indices `alap` des autres variables sont déterminées en parcourant la liste des équations décomposées en partant de la dernière et en remontant vers la première.

La figure 3.3 illustre cette possibilité d'effectuer des opérations le plus tôt ou le plus tard possible.

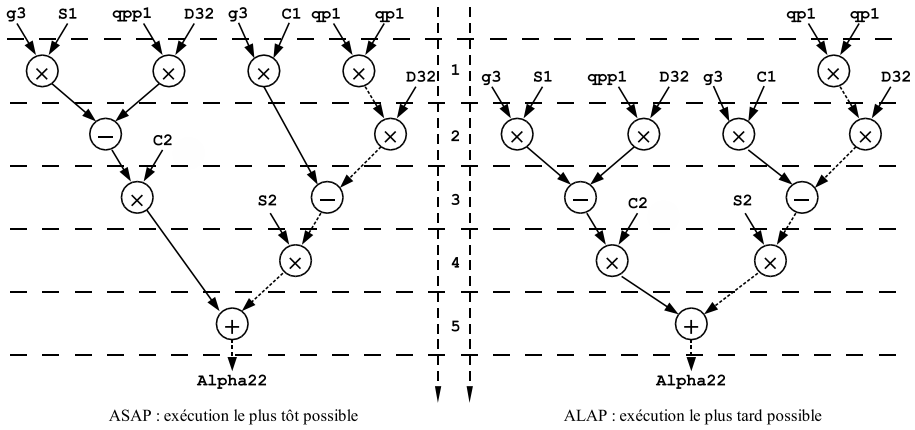


FIG. 3.3 – Représentations ASAP et ALAP d'un ensemble opérations

Remarque On notera que la complexité des différentes opérations peut être prise en compte à ce stade en utilisant une valeur différente de l'unité entre les valeurs des indices `asap` et `alap` des variables auxiliaires successives. Il est donc possible de faire intervenir directement le nombre de cycles nécessaires pour calculer les opérations. Ce nombre dépend naturellement de l'architecture et de l'implémentation du processeur qui sera utilisé pour exécuter les opérations.

3.2 Génération symbolique et taux de parallélisme

Nous attirons ici l'attention sur l'influence de certains choix effectuées au niveau de la génération symbolique, sur le parallélisme des équations obtenues.

3.2.1 Représentation des expressions

Examinons la façon dont les expressions symboliques sont construites. Nous avons expliqué précédemment que les expressions étaient stockées sous forme d'arbres binaires, car les opérateurs arithmétiques sont implémentés comme des fonctions à deux arguments.

Ainsi l'expression binaire $a + b$ est représentée en mémoire comme illustré sur la figure 3.4.

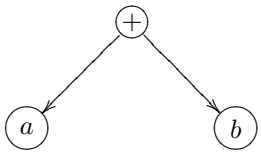


FIG. 3.4 – Expression binaire $a + b$

Pour une expression plus complexe, les possibilités de représentation sont plus nombreuses. Pour n'en considérer que deux, examinons les représentations de l'expression $a + b + c + d$, en *parallèle* ou en *cascade* qui sont illustrées sur la figure 3.5.

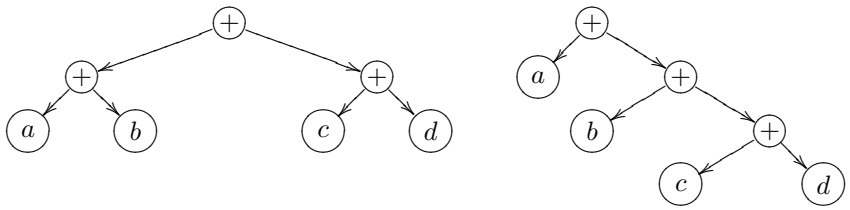


FIG. 3.5 – Arbres binaires

Bien que la représentation en *parallèle* $(a+b)+(c+d)$ soit indéniablement plus compacte en terme de longueur du chemin critique et contienne un taux de parallélisme intrinsèquement plus élevé, les conventions de simplifications de ROBOTRAN, produisent systématiquement des expressions à structure en *cascade* $a + (b + (c + d))$. Toutefois la juxtaposition, ou l'entrelacement de plusieurs structures en cascades conduit à un taux de parallélisme suffisamment élevé et à une répartition[3] plus intéressante des opérations dans les unités de calcul en terme de taux de remplissage, après ordonnancement.

[3] voir section 3.3.1

3.2.2 Adaptation des algorithmes

Le taux de parallélisme peut être augmenté par l'adaptation des algorithmes de générations d'équations.

Nous prenons comme exemple la génération explicite des accélérations pour un modèle dynamique direct. Nous nous concentrons ici sur la séquence d'opérations résultant de la génération des termes de la matrice de masse et de sa factorisation par la méthode LDL^T. Sans entrer à nouveau dans les détails du formalisme récursif utilisé pour obtenir les termes de la matrice de masse, rappelons que ces termes sont issus d'un parcours récursif inverse [17] dont la conséquence est que ce sont les termes du 'fond', c'est-à-dire les termes de la dernière ligne (et de la dernière colonne, vu la symétrie de la matrice de masse) qui sont obtenus en premier, puis ceux de l'avant dernière ligne, et ainsi de suite. Ce processus est illustré sur la figure 3.6.

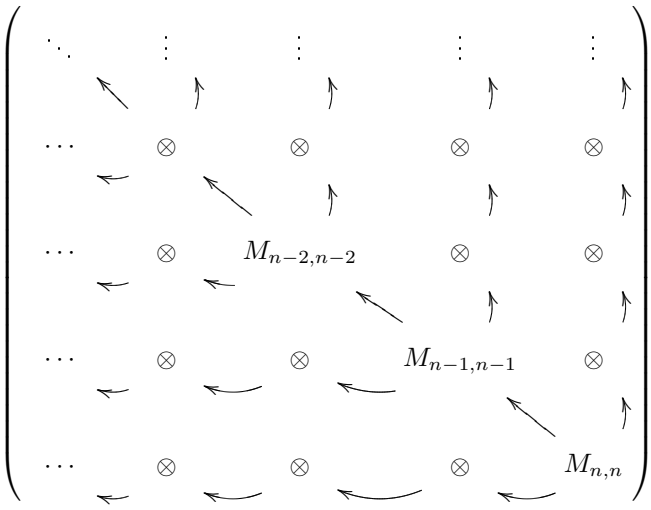


FIG. 3.6 – Génération récursive de la matrice de masse

Quant à la factorisation LDL^T, elle est habituellement effectuée en commençant par la première ligne et la première colonne de la matrice. En conséquence, aucun calcul relatif à la factorisation de la matrice ne peut commencer avant que tous les termes soient évalués.

On peut néanmoins imbriquer les deux procédures en inversant les séquences de l'algorithme de factorisation LDL^T et en commençant par la dernière ligne et la dernière colonne de la matrice, soit par le premier élément disponible. Ainsi la factorisation de la matrice peut commencer et se dérouler en même temps

que sa génération, ce qui permet d'obtenir une répartition[4] des opérations plus uniforme et un chemin critique un peu plus court.

3.3 Ordonnancement des opérations

Pour distribuer les opérations de la liste entre les différentes unités de calcul d'un processeur, nous utilisons une méthode d'ordonnancement simple qui est connue sous le nom générique de *list scheduling*.

Il s'agit d'une méthode heuristique dont la première étape consiste à trier les opérations par ordre croissant de valeur d'indice `alap` et par ordre croissant de valeur d'indice `asap` pour celles dont les indices `alap` ont la même valeur. La liste est alors parcourue du début vers la fin et les opérations sont affectées les unes après les autres aux différentes unités de calcul disponibles en tenant compte de l'interdépendance des opérations : par exemple, une addition qui dépend du résultat d'une multiplication exécutée à l'étape s ne pourra pas être exécutée à la même étape s, même si une unité de calcul d'addition est disponible.

Il est possible de faire intervenir la *complexité* des opérations lors de leur ordonnancement, ainsi que le délai de transfert du résultat d'une unité de calcul vers une autre. Ceci peut être critique lorsque ce délai de transfert n'est pas négligeable, ce qui dépend de l'architecture et du nombre d'unités d'exécution parallèles.

On incrémente alors d'une valeur supérieure à 1, l'étape s_i à laquelle est exécutée une opération op_i pour déterminer à quelle étape pourra être exécutée une opération op_j qui dépend du résultat de op_i. Cette valeur sera la somme de la complexité $O(op_i)$ de l'opération op_i ainsi que du délai $D(u_i, u_j)$ de transmission d'une donnée de l'unité de calcul u_i où est calculée op_i vers l'unité de calcul u_j disponible où pourrait être exécutée op_j. Il peut y avoir plusieurs possibilités pour placer l'opération op_j selon la disponibilité des unités de calcul à chaque étape et le nombre d'étapes nécessaires au transfert. On choisit la solution qui minimise s_j, avec $s_j = s_i + O(op_i) + D(u_i, u_j)$.

Le résultat de l'ordonnancement permet donc de connaître entre autres choses, le nombre d'étapes nécessaires pour évaluer un modèle sur une architecture donnée.

3.3.1 Exécution en temps minimal

Le problème réciproque de l'ordonnancement des opérations sur un processeur donné consiste à déterminer les caractéristiques de base d'une architecture vectorielle, essentiellement les nombres d'unités de calcul de chaque type, afin de pouvoir y exécuter un modèle donné en un nombre minimum d'étapes. Une

[4] voir section 3.3.1

variante consiste à chercher à obtenir une utilisation maximale des unités de calcul, tout en essayant de maintenir un temps d'exécution assez bas.

Estimation des ressources nécessaires

Nous ne disposons pas d'une méthode directe pour calculer les nombres exacts d'unités de calcul nécessaires. Une méthode itérative doit être utilisée. Pour cela, nous estimons les bornes inférieures et supérieures, puis nous recherchons la solution entre ces bornes.

Borne inférieure du nombre d'unités de calcul Afin d'obtenir une estimation du nombre d'unités de calculs nécessaires, on peut compter les nombres totaux d'opérations de chaque type et diviser ces nombres par la longueur du chemin critique. Les valeurs obtenues sont toujours des *sous-estimations* plus ou moins importantes des nombres d'unités nécessaires : cette estimation suppose en effet une utilisation de toutes les unités de calcul à chaque étape, ce qui n'est généralement pas possible en raison des interdépendances entre les opérations. Toutefois, pour les petits problèmes, les valeurs des moyennes globales, arrondies à l'unité supérieure constituent de très bonnes estimations.

Borne supérieure du nombre d'unités de calcul On peut compter les nombres d'opérations de chaque type pour chaque valeur des indices `asap` et retenir les valeurs maximales pour chaque type. On peut également inventorier les nombres d'opérations pour chaque valeur des indices `alap`.

Ces deux façons d'évaluer les bornes supérieures fournissent généralement des résultats différents selon que l'on utilise les indices `asap` ou `alap`. En outre, les quantités obtenues sont systématiquement bien supérieures aux nombres d'unités réellement nécessaires.

L'écart entre les valeurs obtenues pour les bornes inférieures et supérieures peut être assez grand. Il est donc nécessaire d'élaborer une méthode simple et directe qui fournit une meilleure sur-estimation du nombre d'unité de calcul nécessaires la solution. Pour cela, nous proposons une méthode de pondération qui tient compte des valeurs des indices `asap` et `alap` de chaque opération.

Méthode statistique d'estimation Considérons un tableau $U_{i,j}$ dont les colonnes j correspondent aux différents types d'opérations et dont les lignes i correspondent aux étapes d'exécution. Le nombre de lignes est alors égal à la longueur du chemin critique qui vaut $asap_{max}$.

On initialise tous les éléments $U_{i,j}$ à zéro.

Ensuite, pour chaque opération op, on calcule la plage rg_{op} dans laquelle elle peut être exécutée. Cette plage est la différence entre la valeur des indices `alap` et `asap`, plus un : $rg_{op} = 1 + alap_{op} - asap_{op}$. En considérant une distribution uniforme, il y a donc une probabilité $1/rg_{op}$ que l'opération op soit exécutée

à l'étape i pour $asap_{op} \leq i \leq alap_{op}$. La valeur $1/rg_{op}$ est alors ajoutée aux termes $U_{i,j}$ tels que $asap_{op} \leq i \leq alap_{op}$ et que j soit le type de l'opération op.

Lorsque toutes les opérations ont été passées en revue, la somme des éléments de chaque colonne doit alors correspondre au nombre d'opérations de chaque type. On peut également utiliser les valeurs des colonnes pour illustrer cette *estimation de la répartition* des opérations de chaque type à chaque étape, *avant ordonnancement*.

On pourrait alors rechercher pour chaque colonne l'élément de valeur maximum, et prendre sa valeur arrondie à l'unité supérieure comme estimation du nombre d'unités de calculs nécessaires pour le type d'opération correspondant à cette colonne. Bien que cette estimation soit assez proche de la solution, elle est encore très souvent un peu trop grande.

Par expérience, la meilleure estimation est obtenue en calculant, pour chaque colonne de U les moyennes partielles $moy(i,j)$

$$moy(i,j) = \sum_{k=1}^{i} U_{i,j} \tag{3.1}$$

pour $1 \leq i \leq asap_{max}$ et en prenant les valeurs maximales obtenues pour chaque colonne j, arrondies à l'unité supérieure. Ces valeurs fournissent de bonnes sur-estimations des nombres d'unités de calculs de chaque type, de manière à pouvoir exécuter l'ensemble des opérations en un nombre d'étapes égal à la longueur du chemin critique.

Méthode itérative Sur base des estimations des bornes inférieures, nous procédons alors à un ordonnancement des opérations pour vérifier le nombre d'étapes d'exécution. Afin de trouver une solution, nous examinons les combinaisons voisines des bornes inférieures en augmentant progressivement les nombres d'unités d'exécution de chaque type jusqu'à ce que le nombre d'étapes d'exécution soit égal à la longueur du chemin critique, après ordonnancement.

Il arrive parfois que plusieurs solutions existent. Elles sont différenciées en fonction du coût des unités de calcul. Par exemple, une solution $\{5, 5, \cdots\}$, qui requiert 5 unités de multiplication et 5 unités d'addition/soustraction sera préférée à une solution $\{6, 4, \cdots\}$, car une unité de multiplication est plus complexe et plus coûteuse dans un processeur.

D'autre part, en réduisant les nombres d'unités de calculs des différents types, on peut également obtenir une configuration qui présente un taux d'utilisation maximal. On définit le taux d'utilisation tx comme le rapport entre le produit du nombre d'unités arithmétiques par le nombre de cycles et le nombre d'opérations à effectuer : $tx = nb_{UA} \cdot nb_{cycle}/nb_{op}$. Naturellement, le nombre de cycles d'exécution à tendance à augmenter et devient supérieur à la valeur de la longueur du chemin critique, à chaque fois qu'on enlève une unité de

calcul, mais on peut toujours trouver une configuration qui maximise le taux d'utilisation.

3.3.2 Illustrations

Nous présentons ici les résultats d'estimations et d'ordonnancements obtenus en appliquant les méthodes présentées ci-dessus à la liste des opérations à effectuer pour calculer le modèle dynamique inverse d'un robot PUMA. Ce robot manipulateur à six degrés de liberté a été présenté dans la section 1.7.1. Pour simplifier le problème nous n'avons pas pris en compte le calcul des opérations trigonométriques *sinus* et *cosinus* relatives aux variables articulaires. Ce modèle dynamique inverse comporte 338 opérations arithmétiques réparties en 193 multiplications et 145 additions ou soustractions.

Situation 1

Dans ce premier cas on considère, de façon simpliste, que chaque unité de calcul peut évaluer une opération en un seul cycle, quelle que soit sa nature et que le résultat calculé dans une unité de calcul à la fin d'un cycle peut être transféré sans délai pour être utilisé au début du cycle suivant dans une autre unité de calcul.

La longueur du chemin critique, soit le nombre minimum de cycles nécessaires pour exécuter l'ensemble des opérations, est alors de 47. Les estimations des nombres minimaux d'unités de multiplication et d'addition valent respectivement 5 et 4.

Les figures 3.7 et 3.8 illustrent une estimation de la distribution des opérations basée sur les indices `asap` et `alap` comme expliqué ci-dessus. Les courbes en pointillé correspondent à une valeur moyenne arithmétique calculée à chaque cycle i sur l'ensemble des cycles précédents. Les estimations des bornes supérieures sont de 7 unités de multiplication et de 4 unités d'addition.

La table 3.1 montre les résultats d'ordonnancement des opérations de multiplication $\times$ et d'addition ou soustraction $\pm$ pour différents nombres d'unités de calcul en parallèle.

On peut constater deux résultats intéressants :

- on trouve une configuration la plus économique en terme d'unités d'exécutions pour pouvoir exécuter toutes les opérations en un nombre minimum de cycles correspondant à la longueur du chemin critique,
- on trouve également une configuration optimale en terme d'utilisation des unités d'exécution. On constate que pour cette solution optimale, les unités de calcul des différents types sont présentes dans les mêmes proportions, 4/3, que les nombres respectifs d'opérations.

Les figures 3.9 et 3.10 montrent la distribution des opérations (multiplications, additions et soustractions) après leur ordonnancement pour exécution

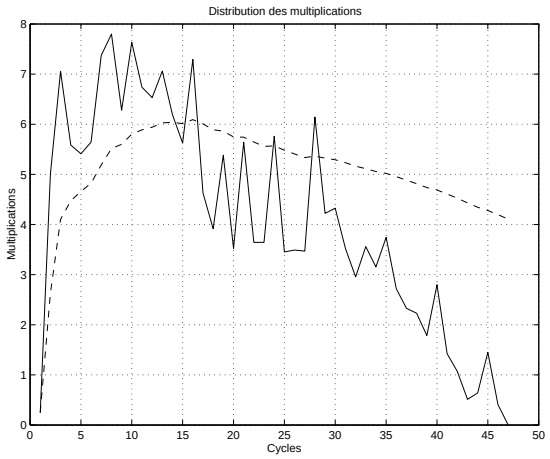


FIG. 3.7 – Dynamique inverse d'un robot PUMA : distribution des multiplications.

sur un processeur possédant 5 unités de multiplication et 4 unités d'addition/soustraction. On peut ainsi observer l'utilisation des unités de calcul à chaque cycle.

Situation 2

Considérons maintenant que les unités de multiplications ont une structure en pipeline à trois étage. Il y a donc un délai de trois cycles pour obtenir le résultat d'une multiplication. On considère toujours qu'une addition ou une soustraction sont calculées en un seul cycle

On considère en outre qu'il faut un cycle pour transférer le résultat d'une unité de calcul vers une autre unité de calcul.

Ces valeurs sont choisies de façon purement subjective pour illustrer une situation différente de la précédente.

Le nombre minimal de cycles nécessaire s'élève maintenant à 118. Les estimations des nombres minimaux d'unités de multiplication et d'addition sont alors égales à 4.

Les figures 3.11 et 3.12 montrent les graphes d'estimation des distributions des multiplications et des additions ou soustractions. On constate sur ces graphes que l'augmentation de cycles à pour conséquence de faire baisser le profil des courbes d'estimation du nombre d'unités de calcul. Les estimations des bornes supérieures sont de 3 unités de multiplication et de 2 unités d'addition.

×	±	cycles	usage
1	0	1	-
0	1	1	-
7	4	47	0.654
6	4	47	0.719
5	4	*47*	0.799
4	4	51	0.828
5	3	50	0.845
3	4	66	0.732
4	3	52	*0.929*
4	2	73	0.772

TAB. 3.1 – Résultats d'ordonnancement (1,1)

×	±	cycles	usage
1	0	3+1	-
0	1	1+1	-
3	2	118	0.573
2	2	*118*	0.716
1	2	194	0.581
2	1	145	0.777
1	1	194	*0.871*

TAB. 3.2 – Résultats d'ordonnancement (3+1, 1+1)

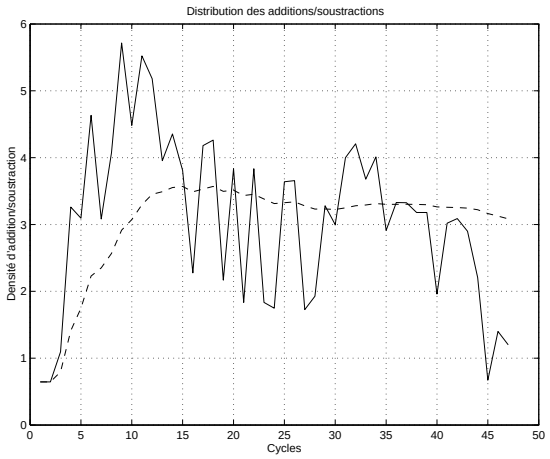


FIG. 3.8 – Dynamique inverse d'un robot PUMA : distribution des additions et des soustractions.

La table 3.2 contient les résultats obtenus après les ordonnancements effectués pour différentes combinaisons de nombres d'unités de calculs. On constate que la solution optimale en terme d'utilisation des unités de calcul, présente un taux moins élevé que dans la première situation, car le rapport entre les nombres d'unités de calcul différentes ne sont pas les même et ne correspondent plus aux proportions d'opérations de chaque type.

Les figures 3.13 et 3.14 illustrent la répartition des opérations après ordonnancement sur un processeur comportant 2 unités de multiplication et 2 unités d'addition ou soustraction. Il s'agit de la combinaison la plus économique qui permet d'exécuter l'ensemble des opérations en un nombre de cycles minimum. On constate sur la figure 3.14 que les unités d'addition ou soustraction sont moins utilisées qu'elles ne l'étaient dans le premier cas, et ce en raison de la quantité d'additions et de soustractions inférieure au potentiel de calcul disponible.

3.3.3 Analyse de différents systèmes multicorps

Nous présentons dans la table 3.3 des résultats d'ordonnancement des modèles dynamiques de différents systèmes. Nous avons limité l'analyse aux équations résultant de l'application des formalismes récursifs Newton-Euler fournissant les formes implicites, explicites et semi-explicites relatives à la topologie ouverte de ces systèmes. Bien que cette analyse, ne porte pas sur les jeux com-

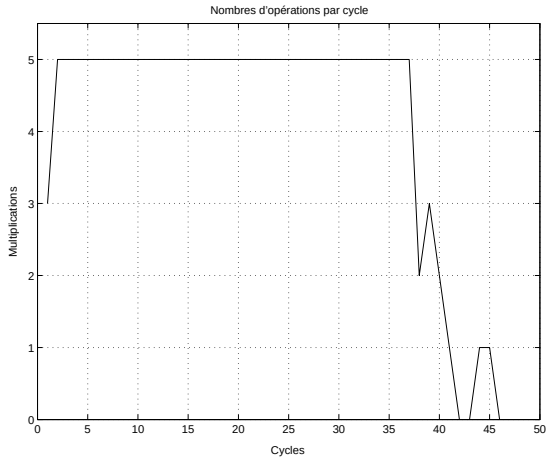


FIG. 3.9 – Dynamique inverse d'un robot PUMA : distribution des multiplications après ordonnancement

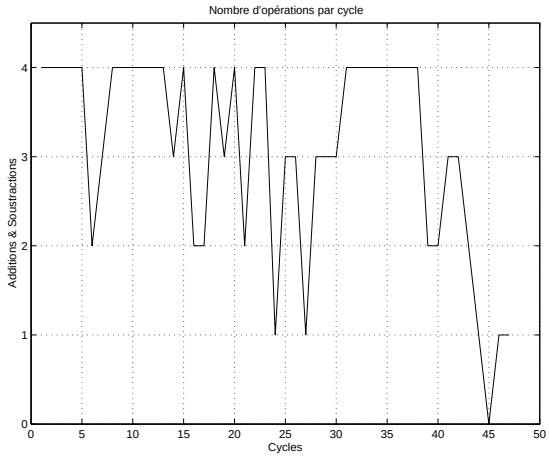


FIG. 3.10 – Dynamique inverse d'un robot PUMA : distribution des additions et des soustractions après ordonnancement

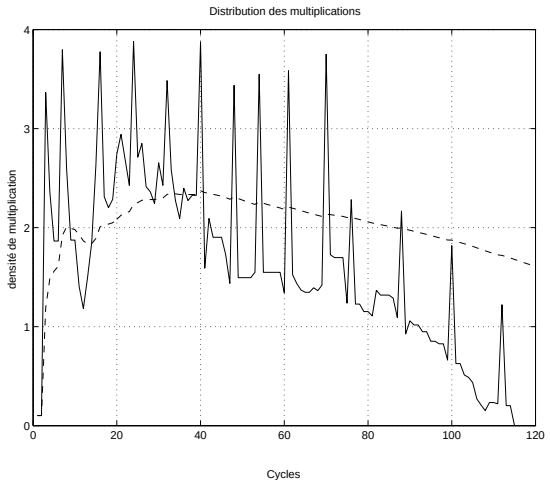


FIG. 3.11 – Estimation de la distribution des multiplications.

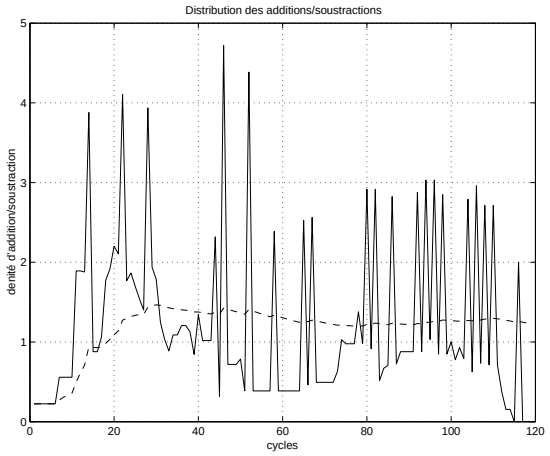


FIG. 3.12 – Estimation de la distribution des additions et des soustractions.

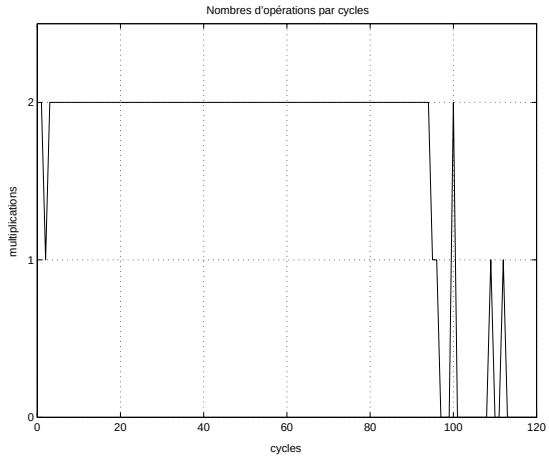


FIG. 3.13 – Ordonnancement des multiplications

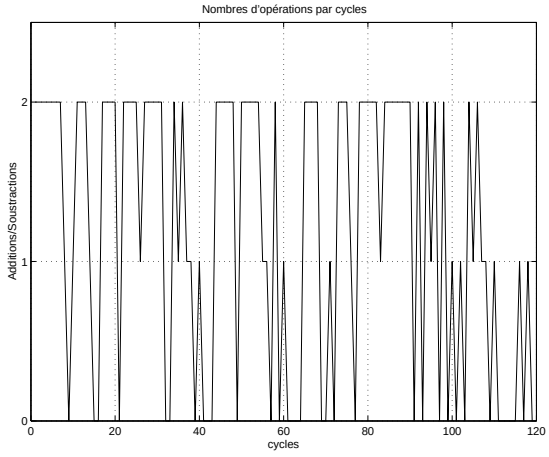


FIG. 3.14 – Ordonnancement des additions et soustractions

plets d'équations[5], elle permet néanmoins de se faire une idée de l'ordre de grandeur du taux de parallélisme présent dans les modèles.

Les systèmes analysés sont les suivants :

1. Un pendule plan constitué de cinq corps en série.
2. Un robot PUMA. C'est un bras manipulateur série à six degrés de liberté.
3. Une plateforme de Gough-Stewart. C'est une structure parallèle à six degrés de liberté. Elle est modélisée en utilisant 24 articulations réparties entre six branches parallèles et une chaîne cinématique fictive correspondant aux coordonnées absolues de la plateforme.
4. Un modèle simple de moto. Ce modèle présenté à la section 1.7.2 contient 14 articulations.
5. Un modèle simple de voiture, qui contient 16 articulations.
6. Un bogie ferroviaire articulé contenant 23 articulations.
7. Une voiture Audi A6. Le modèle comporte plus de 50 articulations.

Systèmes	Implicite				Explicite					Semi Explicite			
	×	±	cc	usage	×	±	÷	cc	usage	×	±	cc	usage
1	4	3	31	0.641	4	3	1	68	0.796	6	5	34	0.850
	3	2	36	0.772	4	3	1	68	0.796	5	4	39	0.906
2	5	4	47	0.799	6	5	1	86	0.805	10	7	44	0.874
	4	3	52	0.929	6	4	1	89	0.849	10	6	46	0.889
3	6	3	17	0.889	6	3	1	30	0.800	10	5	15	0.831
	5	3	19	0.895	5	3	1	33	0.808	9	4	17	0.846
4	8	7	53	0.805	10	7	1	194	0.885	23	19	53	0.879
	7	6	56	0.879	10	7	1	194	0.885	22	16	57	0.903
5	9	9	45	0.828	9	7	1	203	0.848	28	22	43	0.855
	8	8	48	0.874	9	6	1	207	0.884	28	22	43	0.855
6	14	11	58	0.806	18	14	1	164	0.904	44	36	57	0.881
	13	11	59	0.825	17	13	1	174	0.907	44	36	57	0.881
7	25	21	147	0.712	58	38	1	246	0.966	95	62	146	0.963
	23	18	156	0.753	58	38	1	246	0.966	95	62	146	0.963

TAB. 3.3 – Ordonnancement de modèles dynamiques de différents systèmes

Pour chaque système nous avons consigné les résultats sur deux lignes : celui du meilleur ordonnancement en un nombre de cycles minimal sur la première et un résultat qui maximise l'usage global des unités arithmétiques sur la seconde.

Comme annoncé précédemment, nous n'avons considéré que les quatre opérations arithmétiques de base. On notera que seul le formalisme explicite fait apparaître des divisions en raison de la factorisation de la matrice de masse et de la résolution du système linéaire qui permet d'obtenir les expressions des accélérations.

Observations Au vu des résultats repris dans la table 3.3, on pourrait croire que la formulation semi-explicite est celle qui présente le taux de parallélisme

[5] si le système comporte des boucles, ou est soumis à des forces extérieures, etc.

le plus élevé, ce qui est surprenant si on se rappelle que les opérations de la fomulation semi-explicite constituent un sous ensemble de celles de la formulation explicite. Bien que la formulation explicite implique un plus grand nombre d'opérations, il se produit en fait un étalement des opérations sur son chemin critique qui est plus long que celui de la formulation semi-explicite.

Ceci explique les nombres moins élevés d'unités arithmétiques nécessaires pour l'exécution en un minimum d'étapes et des taux d'utilisation des unités arithmétiques souvent supérieurs pour la formulation explicite.

Quant aux nombres d'unités arithmétiques exploitables, c'est à dire aux nombres opérations simultanées que l'on peut effectuer à chaque étape, on constate qu'ils sont voisins de 10 pour des petits systèmes. Pour des systèmes plus complexes, ils peuvent s'élever de 50 à 100 et cela avec des taux d'utilisation de près de 90%. Ces valeurs permettraient des réductions de temps de calcul impressionantes si on disposait d'une architecture parallèle adéquate.

Toutefois, il faut rappeler ici que ces valeurs ont été obtenues en considérant que chaque unité arithmétique effectue une opération en un seul cycle et que le transfert du résultat produit par une unité de calcul vers une autre se fait sans délai. Ces hypothèses sont évidemment un peu simplistes et la réalité des processeurs est un peu différente.

Néanmoins, même si on considère que certaines opérations comme les multiplications nécessitent plusieurs cycles pour être calculées et que les transferts des données prennent également plusieurs cycles, on remarque que le nombre d'unités arithmétiques parallèles diminue dans les mêmes proportions que l'augmentation du chemin critique, ce qui tend à maintenir le taux d'utilisation des ressources calculs à des valeurs qui se situent entre 80% et 90%.

3.4 Implantation sur un prototype d'architecture parallèle

Afin de démontrer l'intérêt réel de la méthode de vectorisation, nous avons procédé à l'implantation d'un modèle dynamique sur une architecture de calcul parallèle à grain fin.

N'étant pas des spécialistes de la conception de circuits digitaux, nous nous sommes adressé à nos collègues du laboratoire de micro électronique. Jean-Pierre David développait à cette époque une architecture synchronisée par les données [59] destinée au prototypage de systèmes de calcul digitaux.

Comme cette architecture originale présentait les caractéristiques nécessaires, un prototype de processeur parallèle capable d'exécuter des schémas de calcul issus de la modélisation de systèmes multicorps a été réalisé pour effectuer quelques tests.

3.4.1 Architecture synchronisée par les données

Cette architecture synchronisée par les données est composée d'un ensemble de ressources, ou de modules de base, que l'on peut connecter entre eux pour composer un système. Le module de base utilisé contient une unité arithmétique de multiplication, une unité arithmétique d'addition ou de soustraction, ainsi que des ressources de mémorisation et quelques autres éléments d'interconnexion. Un tel module est illustré sur la figure 3.15.

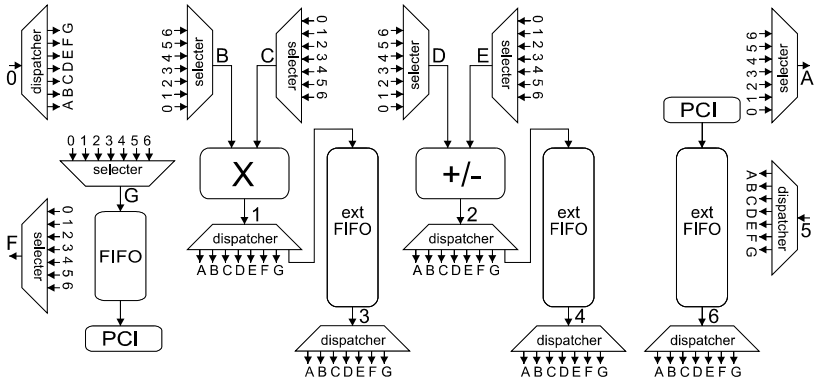


FIG. 3.15 – Module de base de l'architecture synchronisée par les données

Parmi les éléments d'interconnexion, on distinguera ceux qui reçoivent des données, ils sont désignés par des lettres, A..G, et ceux qui fournissent des données qui sont numérotés 0..6. Ainsi les sorties des deux unités de calculs, sources de données, sont numérotés 1 et 2, alors que leurs entrées sont désignées B, C, D et E.

Synchronisation par les données

La synchronisation par les données signifie que les transferts entre un élément source et un élément récepteur a lieu lorsque la donnée est disponible dans la source et que le récepteur est libre pour la recevoir. C'est la coïncidence de la disponibilité d'une donnée à destination d'un récepteur dans une source et de la disponibilité du récepteur envers cette source qui provoque le transfert. Le lecteur intéressé par des explications détaillées sur le fonctionnement de cette architecture trouvera toutes les informations dans [58] et [59].

Mise en oeuvre

Pratiquement, les informations à fournir pour exploiter cette architecture sont les séquences d'envoi et de réception des données pour chaque élément de connexion. Ainsi, pour une source, la séquence d'envoi sera constituée de la suite des adresses des récepteurs des données à transférer, et pour un récepteur, la séquence de réception sera constituée de la suite des adresses des sources des données. Naturellement, c'est la cohérence de ces séquences d'adresses qui garantit le fonctionnement correct du système.

Afin de générer ces séquences, nous utilisons la procédure d'ordonnancement décrite à la section 3.3. L'utilisateur choisit le nombre de modules de base qu'il veut utiliser et ROBOTRAN produit toutes les informations nécessaires à l'implantation du schéma de calcul d'un modèle sur cette architecture :

- la taille des mémoires tampons,
- l'affectation des opérations aux unités de calcul,
- l'affectation des variables temporaires dans les zones tampons.

En ce qui concerne la taille des mémoires tampons, pour un schéma de calcul donné, il est assez facile déterminer le temps de séjour d'une variable auxiliaire dans une mémoire tampon en fonction de l'étape à laquelle elle est calculée et de la dernière étape à laquelle elle est utilisée. On peut ainsi déterminer le nombre de variables à stocker dans chaque mémoire tampon à chaque étape et donc la taille minimale nécessaire pour chaque mémoire.

Implémentation d'un processeur parallèle

L'avantage principal de l'utilisation de cette architecture est précisément son côté modulaire. La possibilité d'interconnecter un nombre quelconque de modules est particulièrement intéressant pour notre application.

Cette architecture était totalement fonctionnelle et directement utilisable pour l'implémentation d'un processeur parallèle sur système de prototypage, également développé par Jean-Pierre David.

Ce système était composé de quatre FPGA[6] FLEX 10K100 montés sur une carte PCI installée dans un ordinateur personnel standard. Ce système est présenté en détails dans [57].

Sa capacité permettait la synthèse de quatre modules de base, soit une capacité de calcul de quatre multiplications et quatre additions ou soustractions en parallèle. Un schéma du processeur constitué de quatre modules est présenté sur la figure 3.16.

Chaque module est connecté à deux voisins, ce qui correspond à une configuration en anneau. Une telle structure présente une capacité de communication

[6] Field Programmable Gate Array : composant électronique reconfigurable destiné à la synthèse de systèmes logiques digitaux

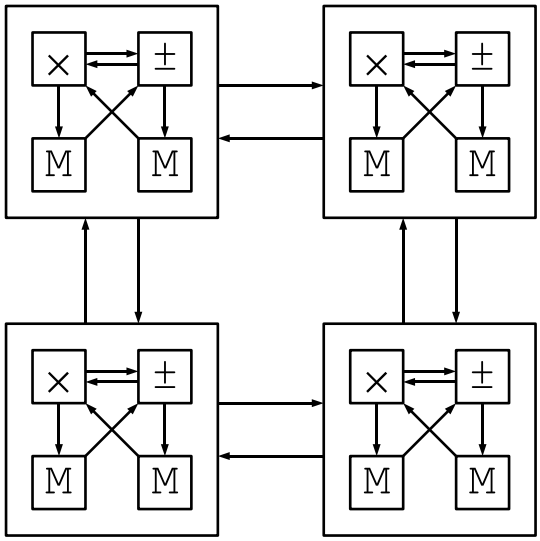


FIG. 3.16 – Schéma de la configuration du protype de processeur parallèle

plus limitée qu'une structure en étoile. Il serait possible de rajouter des éléments de communication pour améliorer cet aspect. On pourrait dédoubler les éléments de connexion vers l'extérieur - notés 0, 5, A, F sur la figure 3.15 - ce qui permettrait en outre d'implémenter une structure en treillis plus performante en terme de communication.

Une autre solution consisterait à placer plus d'unités arithmétiques dans un même module, au sein duquel tous les éléments sont connectés directement les uns aux autres selon une structure en étoile. Toutefois cette option implique de disposer de circuit de prototypage de plus grande capacité.

On remarque sur la figure 3.16 que chacun des quatre modules de base contient deux unités arithmétiques et deux mémoires tampons. Les autres éléments des modules ne sont pas représentés. Les quatre modules étaient distribués dans les quatre FPGA du système de prototypage, qui avaient juste la capacité suffisante pour contenir un module chacun.

On notera que le FPGA FLEX 10K100 utilisé pour la réalisation du système de prototypage était un classique en 1998. Il contenait 4992 cellules logiques et des cellules de mémoires d'un capacité totale de 24kbits, soit à peine 768 mots de 32 bits. Il était donc relativement difficile d'envisager d'implémenter des modules de base beaucoup plus complexes que ceux utilisés.

En 2004, la situation est bien différente, Altera propose un nouveau circuit, le Stratix II qui peut contenir jusqu'à l'équivalent de 180000 cellules logiques,

9Mbits de capacité mémoire et 392 unités câblées de multiplication entière de 18×18 bits qui peuvent servir à implémenter des fonctions de calcul diverses.

Il serait actuellement possible d'implanter tout un processeur dans un seul composant.

Ordonnancement des opérations

Plusieurs tests d'ordonnancement ont été effectués pour différents modèles de systèmes multicorps. Ils ont été présentés dans [58] et [59]. Nous ne présentons ici que les résultats relatifs au calcul du modèle dynamique inverse d'un robot PUMA.

Les opérations du modèle ont été ordonnancées en vue de l'utilisation de quatre unités de multiplication et quatre unités d'addition ou de soustraction, ce qui correspond à la répartition des opérations entre quatre modules de base. D'autres test ont également été effectués en utilisant seulement 1, 2 ou 3 modules.

Cette répartition a toutefois été faite sans tenir compte des capacités limitées de communication du prototype de processeur, ce qui provoque en fait un petit problème de performance : en effet la distribution des unités de calcul entre des modules différents ne permet pas d'effectuer tous les transferts en un seul cycle, comme il a été supposé lors de l'ordonnancement des opérations.

En effet, pour passer d'un module à l'autre, une donnée doit transiter par plusieurs éléments de connexion. La conséquence est évidemment une perte d'efficacité du système dont certaines unités de traitement se retrouvent alors en attente de données pendant quelques cycles, ce qui rallonge le temps de calcul global par rapport aux prévisions.

Il est possible de tenir compte de ces délais de communication lors de l'ordonnancement des opérations en faisant intervenir la provenance des opérandes pour privilégier l'exécution d'une opération dans un module de base ou un autre.

Néanmoins, la méthode d'ordonnancement n'a pas été modifiée, car l'objectif principal était de démontrer la faisabilité de l'implantation d'un schéma de calcul sur une architecture de calcul parallèle à grain fin.

Or cet objectif était déjà atteint lorsque l'analyse des résultats de simulation a révélé ces problèmes de performance, et nous avons ensuite préféré nous consacrer à d'autres développements.

3.4.2 Résultats de simulation sur FPGA

Les résultats de simulation sur la carte multi-FPGA et leur analyse sont présentés dans [58] et plus en détails dans [59]. Ces résultats souffrent un peu des limitations des FPGA de l'époque et de la simplicité de la méthode d'ordonnancement utilisée, laquelle ne tenait pas compte des particularités de la

distribution de l'architecture en module, du moins en ce qui concerne la capacité de communication limitée entre des unités de calcul localisées dans des modules différents.

Les résultats essentiels sont repris dans la table 3.4. Le schéma de calcul considéré correspond au modèle dynamique inverse du robot PUMA. Plusieurs tests ont été effectués, en utilisant de 1 à 4 modules de base répartis sur autant de FPGA.

Etant donné le problème de délai de transfert mentionné, nous avons également considéré que les opérations pouvaient prendre plusieurs cycles d'horloge, de 1 à 4, alors que les transferts via les éléments de connexion ne prennent toujours qu'un seul cycle. En augmentant le nombre de cycle d'horloge requis pour effectuer les opérations, on peut ainsi augmenter la capacité de communication relativement à la capacité de calcul et diminuer le coût relatif du transfert des données.

FPGA	1	2	3	4
Chem. Crit.	199	100	68	52
Cycles(1)	423	256	179	173
Coûts %	113	156	163	233
Gains //	1	1.65	2.40	2.44
Cycles(2)	499	286	211	196
Coûts %	25	43	55	88
Gains //	1	1.74	2.36	2.5
Cycles(3)	656	362	263	235
Coûts %	10	21	29	51
Gains //	1	1.81	2.49	2.79
Cycles(4)	849	451	329	283
Coûts %	7	13	21	36
Gains //	1	1.88	2.58	3

TAB. 3.4 – Résultats de simulation du modèle inverse d'un robot PUMA sur plusieurs FPGA

On reprend pour chaque situation :

- les nombres de **cycles** nécessaire à l'exécution de l'ensemble des opérations. Les nombres minimum correspondent aux longueurs des chemins critiques relatifs aux nombres de modules de base utilisés, et donc de FPGA, dans les conditions de l'époque,
- les **coûts** relatifs de transfert, calculés en tenant compte du nombre de cycles effectifs et de la longueur du chemin critique,
- les **gains** en vitesse d'exécution du à l'utilisation de plusieurs modules de

base.

On constate que :

- le nombre total de cycles augmente lorsque le nombre de cycles par opération augmente, ce qui est assez naturel. Ce nombre total diminue lorsque le nombre de ressources disponibles augmente, ce qui est évidemment le but de l'utilisation de plusieurs unités de calcul en parallèle,
- le coût relatif des transferts est un problème important. Il augmente considérablement avec le nombre de modules utilisés lorsque le nombre de cycles par opération est minimal. On constate alors que l'utilisation de plusieurs modules requiert impérativement l'augmentation de la capacité de communication de l'architecture. On notera toutefois que même dans le cas de l'utilisation d'un seul module, le nombre total de cycles est le double de la longueur du chemin critique. Cela est dû au délai de transfert des données d'un élément à l'autre à l'intérieur d'un même module. En effet, un donnée calculée pendant le cycle i n'est pas disponible pour un calcul qui a lieu au cycle $i+1$, car il faut un cycle pour effectuer un transfert, ce qui double le nombre total de cycles nécessaires par rapport à la longueur du chemin critique.
- les gains obtenus en fonction de l'utilisation de plusieurs modules de base sont relativement bons, si on considère la capacité de communication disponible entre les modules de base et le fait que la méthode d'ordonnancement et de génération des séquences de fonctionnement des éléments de transfert des données ne tenaient pas compte des caractéristiques particulières de l'architecture.

3.5 Autres architectures parallèles à grain fin

Bien que nous n'ayons utilisé qu'un prototype d'architecture parallèle implémenté sur des FPGAs pour effectuer des tests et montrer l'intérêt de nos développements, il nous semble également possible de tirer profit des capacités de calcul vectoriel de la plupart des architectures des processeurs disponibles actuellement pour augmenter la vitesse de calcul des modèles de systèmes multicorps.

La plupart des processeurs sont actuellement dotés d'unités de calcul vectoriel [111] capables d'effectuer simultanément jusqu'à quatre opérations arithmétiques en virgule flottante, en simple précision, comme illustré sur la figure 3.17. Le module SSE du processeur Pentium IV permet aussi d'exécuter deux opérations arithmétiques en virgule flottante en double précision.

Ces technologies portent des noms différents selon les constructeurs :

- Motorola l'a baptisée AltiVec. Le module AltiVec [112] est intégré dans les processeurs PowerPC depuis le G4 ainsi que dans les microcontrôleurs dont le noyau est semblable à celui du PowerPC à savoir les MPC5554,

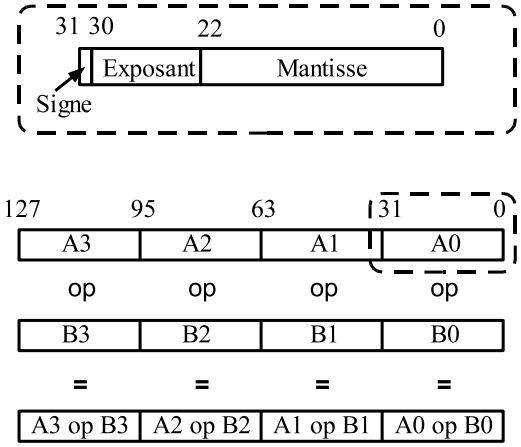


FIG. 3.17 – Unité de calcul vectoriel SIMD

MPC7xxx et supérieurs.

- Elle est connue sous l'acronyme SSE[7] pour les processeurs Intel [113] depuis le Pentium III et sous le nom 3Dnow pour les processeurs Athlon du constructeur AMD.

Le module vectoriel AltiVec disponible dans les processeurs PowerPC est le plus performant car il dispose de 32 registres vectoriels de 128 bits au lieu de 8 pour les processeurs Pentium. De plus le processeur PowerPC peut exécuter des instructions arithmétiques vectorielles et des instructions classiques simultanément ce qui n'est pas possible actuellement pour les autres processeurs.

Le fabriquant de DSP Texas Instrument propose actuellement un processeur ayant une architecture de type VLIW[8] appelée VelociTI [114, 115]. Celle-ci comporte six unités arithmétiques et logiques capables de traiter des nombres codés sur 32 bits et deux unités arithmétiques permettant de multiplier des nombres codés sur 16 bits. Dans la famille C6000, les modèles TMS320C67xx sont capables d'exécuter simultanément deux multiplications et deux additions en virgule flottante en simple précision. Toutefois ces opérations en virgules flottantes requièrent quatre cycles au lieu d'un seul pour les opérations sur des nombres entiers.

D'autre part, on peut également trouver des noyaux de DSP destinés à la synthèse de circuits dédicacés. C'est le cas du noyau VLIW mAgic [116] proposé par Atmel. Celui-ci permet d'effectuer en parallèle 15 opérations par cycle. Atmel annonce une performance de 1 Gflops à 100 Mhz. Les nombres en

[7] SSE : SIMD Streaming Extension
[8] Very Long Instruction Word

virgule flottante sont codés sur 40 bits au lieu de 32 bits en simple précision. Ce noyau est un peu plus de deux fois plus efficace en nombre de cycles que le noyau des DSP TMS320C67xx de Texas Instrument, qui sont parmi les plus performants à l'heure actuelle.

Au moment de la rédaction de ce texte, nous n'avions pas effectué de test sur ces architectures. Il nous semble toutefois que le problème de la distribution des données aux différentes unités de calculs serait bien plus difficile à gérer sur de telles architectures vectorielles ou VLIW que sur une architecture synchronisée par les données. Un nombre important de cycles devrait être dépensé pour réorganiser les données entre les différents registres et au sein de ceux-ci avant de pouvoir exécuter chaque opération arithmétique, ce qui limiterait le gain en performance.

En outre l'évolution récente des FPGA semble leur donner l'avantage par rapport aux processeurs DSP en terme de performance pour les applications demandeuses en calcul qu'il s'agisse de calcul en nombre entier ou en virgule flottante [117].

D'autre part la disponibilité de noyaux de DSP ou encore d'outils de synthèse automatique [118] de circuits sur FPGA à partir de schémas de simulation ou de collection de routines développés en MATLAB par exemple, renforce notre conviction que l'exploitation efficace de nos développements en matière de vectorisation des modèles de systèmes multicorps passe par la synthèse d'un circuit spécifique.

3.6 Conclusions

A ce stade, nous avons donc montré qu'il est possible de mettre en évidence un taux de parallélisme important, même dans les modèles dynamiques des systèmes multicorps à topologie linéaire, et de l'exploiter. Ces modèles dynamiques ont été générés en utilisant le formalisme Newton-Euler récursif présenté dans la chapitre 2. L'architecture synchronisée par les données s'est révélée être une solution intéressante pour le développement d'un système de calcul parallèle à grain fin.

Des modèles dynamiques inverses de robot contenant plusieurs centaines d'opérations ont pu être exécutés en parallèle sur une telle architecture. Les quelques problèmes liés aux particularités de cette architecture et à la limitation des dispositifs de prototypage sont à relativiser lorsque l'on considère l'évolution des caractéristiques des composants de prototypage actuels et la simplicité des méthodes utilisées pour l'ordonnancement des opérations, lors de nos tests.

Bien que cette méthode de *vectorisation* puisse être appliquée à tous les modèles générés par ROBOTRAN, quel que soit le formalisme utilisé et la topologie du système multicorps, il nous paraissait intéressant de disposer également d'une méthode permettant une segmentation des calculs à un niveau plus élevé

que celui des opérations. En procédant de cette manière, les modèles de systèmes de taille plus importante pourraient subir une première parallélisation plus grossière, ce qui permettrait de viser des architectures parallèles plus classique tels que des ordinateurs multiprocesseurs, et éventuellement de servir de solution au problème de la répartition des opérations dans des modules de base, pour des schémas de calcul contenant plusieurs dizaines de milliers d'opérations. Une telle méthode a été élaborée et est proposée dans le chapitre suivant.

Chapitre 4

Découpage automatique des modèles

Dans ce chapitre, nous présentons une méthode de découpage automatique des modèles qui tire profit de leur génération par l'approche symbolique.

Nous commençons par mettre en évidence les différentes possibilités de parallélisation des modèles de systèmes multicorps. Nous proposons ensuite de tirer profit à la fois de la topologie de la structure du système et de la structure algébrique de la Jacobienne des contraintes pour segmenter l'ensemble des équations d'un modèle en sous groupes d'équations suffisamment indépendantes.

Un objectif de cette méthode est de pouvoir retrouver des sous modèles de tailles relativement importantes et comparables à ceux obtenus par une modélisation en sous systèmes, sans qu'il soit nécessaire de segmenter le système au niveau de sa description comme dans [64].

4.1 Parallélisme dans les modèles de systèmes articulés

Lorsqu'on s'intéresse de près à la modélisation des systèmes articulés, on peut mettre évidence différents types de parallélisme, et cela à différents niveaux. Nous distinguons le parallélisme *algébrique*, présent dans les calculs, et le parallélisme *topologique*, présent dans la structure du système.

4.1.1 Parallélisme algébrique

Au niveau des opérations arithmétiques Au niveau le plus bas, celui des opérations arithmétiques scalaires, on peut observer un taux de parallélisme très

élevé dans les équations récursives générées pour l'étude d'un système multi-corps, même si leur topologie est linéaire. A ce niveau, on considère de façon identique toutes les opérations arithmétiques, qu'elles proviennent d'équations cinématiques, dynamiques ou encore algébriques, et quel que soit le formalisme utilisé pour générer ces équations. L'idée de base est la décomposition des expressions symboliques en opérations arithmétiques et l'ordonnancement de ces opérations en tenant compte de leur interdépendance. L'exploitation du parallélisme à ce niveau a fait l'objet du chapitre précédent consacré à ce que nous appelons la *vectorisation* des équations.

Au niveau des macro opérations ou opérations vectorielles [1]

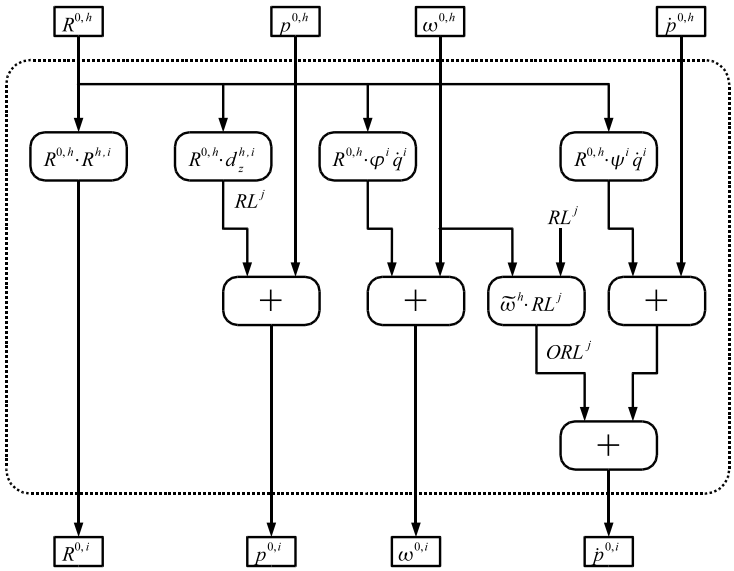


FIG. 4.1 – Une étape du calcul récursif de grandeurs cinématiques

On peut également s'intéresser au formalismes utilisés pour générer ces équations. Nous avons présenté dans le second chapitre une série de formalismes récursifs, permettant de calculer les expressions de vecteurs position, vitesse ou accélération, de matrice de rotation, Jacobienne, ou de masse, etc. Si on représente graphiquement, comme sur la figure 4.1, les calculs vectoriels effectués lors de l'application d'un de ces formalismes, par exemple pour le calcul récursif de quelques grandeurs cinématiques, on peut constater les différents groupes d'opérations et leur interdépendance qui permet de visualiser le

[1] au sens géométrique du terme : vecteur et tenseur de $\mathbf{R}^3$

parallélisme à ce niveau.

Si on considère l'écriture vectorielle d'un algorithme récursif, les opérations sont effectuées entre les vecteurs positions vitesses, etc. et les matrices de rotation, d'inertie, etc. Par exemple, le calcul récursif de la position de l'origine du repère d'un corps i s'écrit :

$$p^{0,i} = p^{0,h} + R^{0,h}\ d_z^{h,i}$$

Les données de base ne sont alors plus des scalaires, mais des vecteurs ou des tenseurs. Le nombre d'opérations vectorielles ou plutôt de macro opérations est alors plus restreint, ce qui allège le coût des procédures d'ordonnancement et permet de traiter des schémas de calculs plus volumineux. Toutefois, dans le contexte de la génération symbolique telle qu'elle est implémentée dans ROBOTRAN, ces macro opérations peuvent parfois se trouver très réduites en raison des simplifications symboliques qui ont lieu, ce qui rend très difficile l'estimation a priori du niveau de complexité de ces macro opérations.

Dès lors, il nous semble plus opportun de travailler directement au niveau des opérations scalaires, comme nous l'avons fait dans le chapitre précédent, afin d'éviter une trop grande disparité dans le coût des macro opérations. De plus, grâce à l'efficacité de ROBOTRAN, nous n'avons pas eu de problème pour gérer l'analyse et l'ordonnancement de gros modèles qui peuvent contenir plusieurs dizaines de milliers d'opérations.

Au niveau des opérations matricielles [2]

Dans les étapes de traitement des équations de contraintes et de réduction des équations du mouvement, nous trouvons un grand nombre de calculs impliquant des matrices : matrice Jacobienne, matrice de couplage des vitesses, matrice de masse. On peut en effet paralléliser le produit de deux matrices, en procédant ligne par ligne ou bloc par bloc.

En particulier, nous avons montré dans les sections relatives au traitement des équations de contraintes qu'il était souvent possible de mettre en évidence une structure bloc triangulaire au niveau de la matrice Jacobienne J_v. Cette structure bloc est due à la fois à la topologie et au partitionnement des coordonnées. Elle se retrouve au niveau de la matrice de couplage des vitesses, et il va également être possible de la mettre en évidence au niveau de la matrice de masse, étant donné son lien avec la topologie du système.

Au niveau de l'algorithme général du modèle Au delà des formalismes récursifs et des calculs matriciels, on peut également analyser la structure générale du modèle du système. La figure 4.2 est une illustration des différentes parties d'un modèle dynamique direct qui permet de simuler numériquement le comportement d'un système.

[2] au sens algébrique du terme : tableaux $m \times n$

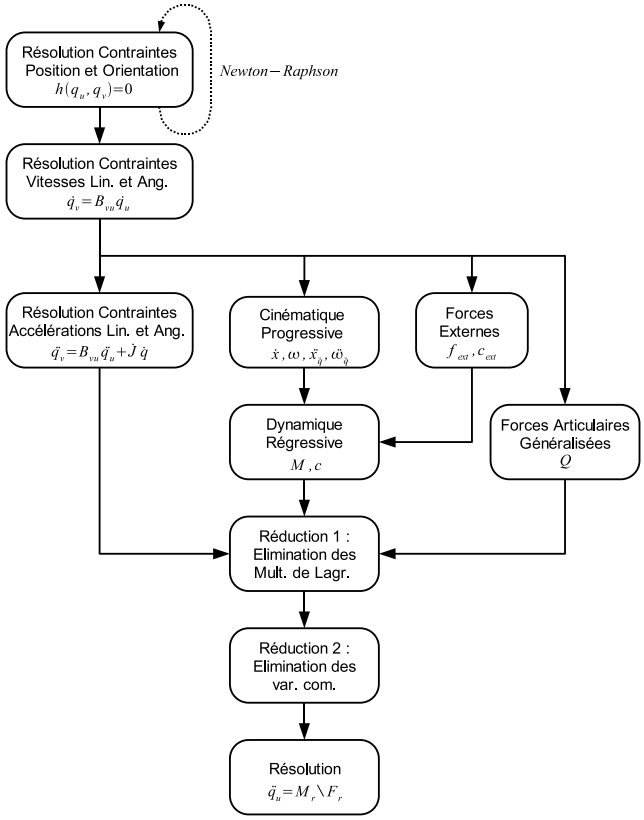


FIG. 4.2 – Diagramme d'un modèle dynamique direct complexe

On constate que la plus grande partie de ce schéma est constituée d'une séquence de blocs. Si on peut apercevoir quatre blocs indépendants qui pourraient éventuellement être exécutés en parallèle, il faut toutefois être attentif à la différence de complexité relative de ces blocs, selon les caractéristiques du système modélisé. En effet, la présence ou non de boucles ou d'interactions avec l'environnement a une influence directe sur la taille de ces blocs d'opérations. C'est pourquoi nous n'exploiterons pas cette possibilité dans notre méthode.

4.1.2 Parallélisme topologique

Outre le parallélisme de bas niveaux des calculs récursifs ou algébriques, il est intéressant de considérer le parallélisme exploitable au niveau de la structure du système. Ce parallélisme peut être considéré à deux niveaux.

Au niveau de l'arborescence Un système arborescent, ou rendu arborescent par diverses coupures, comporte plusieurs *branches*. Nous pouvons citer le cas des robots parallèles ou des suspensions de véhicules. Pour de tels systèmes, on observe une similitude entre la topologie de la structure et la topologie du graphe de dépendance des équations récursives générées pour chacun des corps du système. Cette similitude entre les deux graphes est illustrée sur la figure 4.3. L'orientation du graphe de dépendance des équations peut être identique au graphe de filiation des corps ou inversée, selon que parcours récursif est effectué dans le sens progressif ou régressif lors de la génération de ces équations.

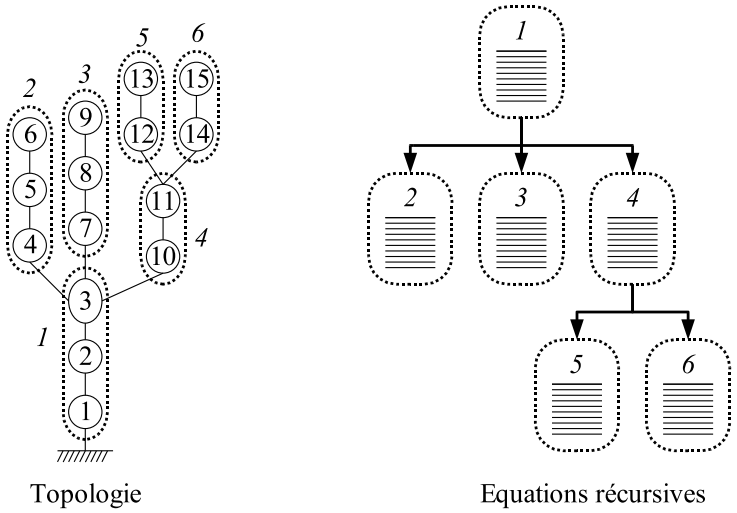


FIG. 4.3 – Relation entre topologie et dépendance des équations récursives

Pour rappel, une branche est constituée d'une série de corps, encadrés en pointillés sur la figure 4.3,

- dont le dernier possède plusieurs ou aucun enfants et
- dont le parent du premier est la base ou le dernier d'une autre branche.

Comme on le constate sur cette illustration, les blocs d'opérations se rapportant aux différentes branches du système ne sont pas complètement indépendants, si les branches différentes sont reliées à une branche commune intermédiaire. Ainsi les blocs d'équations relatives aux branches 2,3 et 4 sont indépendants, mais nécessitent que les équations relatives à la branche 1 soient évaluées d'abord[3]. On notera que la taille des blocs d'opérations peut être est très variable et dépend directement de la taille des branches.

[3]lorsqu'on adopte une approche récursive progressive

Au niveau de la composition du système D'autre part, certains systèmes sont composés ou peuvent être composé de *sous-systèmes*. Nous pouvons citer le cas des trains, composés de plusieurs caisses et bogies, de véhicules routiers, voitures ou camions, et de leur remorque, etc. La figure 4.4 illustre quelques systèmes composés de sous-systèmes.

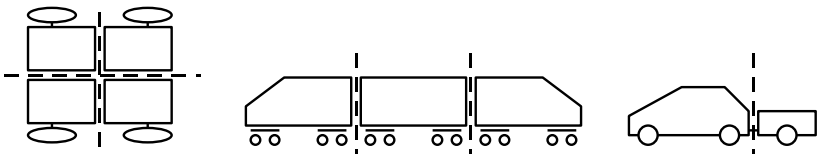


FIG. 4.4 – Exemples de sous-systèmes

Il ne s'agit alors pas de considérer seulement des branches parallèles mais bien des arborescences parallèles. Ces sous systèmes ne sont généralement pas totalement indépendants, ils sont souvent couplés par des forces de liaisons point-à-point ou par des contraintes d'assemblage. Ce couplage implique alors également un couplage au niveau des équations des sous-systèmes.

Toutefois, bien que l'exploitation de cette structure en sous-système [64] se révèle être une approche très efficace, notamment en raison de la taille importante des blocs d'équations correspondant aux sous-systèmes, elle nécessite que les sous-systèmes soient déclarés de façon *explicite* lors de la description du système. Or, dans certains contextes, en particulier dans celui de l'utilisation de coordonnées relatives articulaires, l'utilisateur d'un programme de modélisation n'aura pas forcément le réflexe, la volonté ou la capacité de découper son système de façon judicieuse en vue de sa modélisation.

On peut prendre l'exemple classique d'un véhicule routier ou tout-terrain, que l'on peut diviser en trois ou quatre parties, symétriques ou non, ou plus simplement celui d'une voiture tractant une remorque. Ce dernier exemple est particulièrement intéressant, car l'ingénieur habitué à l'utilisation de coordonnées relatives articulaires aura tendance à relier la remorque à la voiture par une articulation sphérique au lieu de modéliser voiture et remorque comme deux sous systèmes assemblés par une contrainte, ce qui permettrait de paralléliser le modèle.

Il est donc intéressant et utile de disposer d'une méthode un peu plus générale qui pourrait identifier différentes branches, ce qui est aisé, et les regrouper, ce qui l'est moins a priori, de manière à reconstituer l'équivalent de sous-systèmes.

4.2 Possibilités de découpage d'un modèle dynamique direct

Dans cette section, nous voulons montrer comment paralléliser la plupart des blocs d'un modèle dynamique direct tel que celui illustré sur la figure 4.2, en fonction de leurs caractéristiques intrinsèques et de celles du système modélisé. Pour certains blocs, il peut y avoir plusieurs façons d'effectuer leur parallélisation, en revanche certains autres sont difficilement sécables.

Nous désirons également découper chaque bloc selon la même règle de découpage avec pour objectif de parvenir, non seulement à découper chaque bloc en différentes parties, mais également à séparer le diagramme en différentes parties de manière à limiter le nombre de connexions entre les différents sous blocs. Nous expliciterons à nouveau cet objectif plus loin dans cette section.

4.2.1 Séparation des équations de contraintes

Sur le diagramme de la figure 4.2, on remarquera qu'il y a trois blocs relatifs au traitement des équations de contraintes. Comme nous l'avons présenté dans la section 2.4, nous proposons une résolution des contraintes à trois niveaux : en position, en vitesse et en accélération. Ceci implique de résoudre les équations de contraintes ainsi que leurs dérivées premières et secondes. La même procédure de découpage est utilisée pour ces trois groupes d'équations.

Cette procédure de découpage est basée à la fois sur la structure topologique du système et sur la structure algébrique de la matrice Jacobienne des contraintes. En ce qui concerne la topologie, il faut rappeler ici que la structure arborescente résulte de coupures faites par l'utilisateur, dans le cas de systèmes contenant des boucles cinématiques. L'emplacement et le type des coupures utilisées vont conditionner la morphologie de la structure arborescente et donc le parallélisme topologique.

Structure topologique Sur la figure 4.5 nous illustrons la relation existant entre la structure d'une boucle cinématique ouverte par coupure et la structure des équations de contrainte générées pour une telle coupure. Dans le cas d'une coupure bien placée, la structure arborescente va présenter deux branches parallèles, de longueur semblable. Les équations cinématiques générées pour chacune de ces branches sont indépendantes les unes des autres et peuvent donc être évaluées en parallèle.

Comme la complexité du formalisme récursif progressif utilisé pour la génération de la matrice Jacobienne est quadratique en fonction de la longueur des branches, il est intéressant de placer la coupure de telle sorte que l'on obtienne deux branches de taille semblable afin de réduire le nombre total d'opérations dans les équations exprimant la coupure.

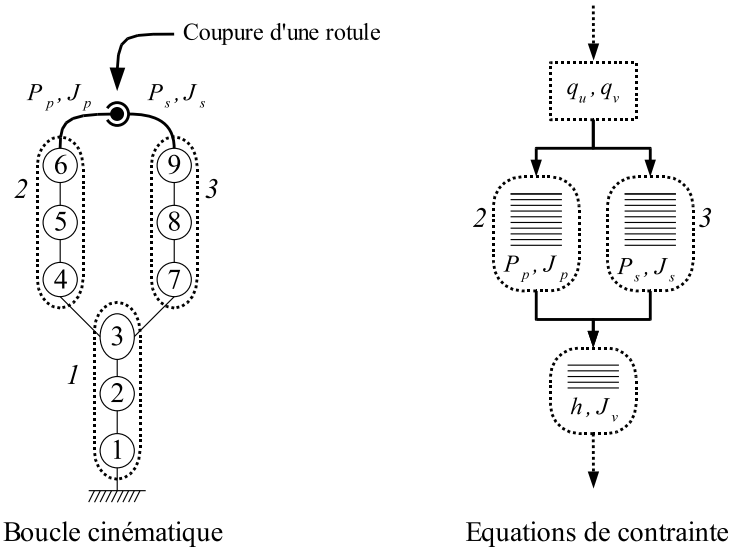


FIG. 4.5 – Séparation des équations de contrainte de coupure d'une boucle

En pratique, on observe rarement des boucles cinématiques très longues et la quantité d'opérations arithmétiques qu'il est possible d'effectuer en parallèle, même lorsque les boucles sont coupées de manière à équilibrer les deux branches, se réduit souvent à quelques centaines voire quelques milliers d'opérations, ce qui est difficilement exploitables sur un ordinateur parallèle classique.

Lors de la génération symbolique, toutes *les équations sont marquées en fonction des branches auxquelles elles se rapportent*, et il est donc possible de les identifier pour les placer dans des blocs de traitement indépendants ou non. Ce choix peut donc toujours être proposé à l'utilisateur ou être automatisé. Dans ce travail, nous n'avons toutefois pas exploité le parallélisme au niveau des branches pour le traitement des équations de contraintes.

Structure de la Jacobienne Nous proposons plutôt de considérer la structure de la matrice Jacobienne. Comme nous l'avons présenté dans la section 2.4.1 relative à la résolution des contraintes, la forme bloc triangulaire de la matrice J_v, lorsqu'elle existe, permet le découplage et la résolution en cascade des équations des contraintes.

$$J_v = \begin{pmatrix} J_v^I & {}_1 0 & 0 & 0 \\ J_v^{II,I} & J_v^{II} & {}_2 0 & 0 \\ J_v^{III,I} & J_v^{III,II} & J_v^{III} & {}_3 0 \\ \otimes & \otimes & \otimes & \end{pmatrix}$$

Les séquences d'opérations à effectuer pour la résolution des contraintes en position :

$$\begin{aligned} & J_v^I(q_v^I) \cdot \Delta q_v^I = h^I(q_u, q_v^I) \\ \hookrightarrow \quad & J_v^{II}(q_u, q_v^I, q_v^{II}) \cdot \Delta q_v^{II} = h^{II}(q_u, q_v^I, q_v^{II}) \\ \hookrightarrow \quad & J_v^{III}(q_u, q_v^I, q_v^{II}, q_v^{III}) \cdot \Delta q_v^{III} = h^{III}(q_u, q_v^I, q_v^{II}, q_v^{III}) \\ & \hookrightarrow \vdots \end{aligned}$$

en vitesse :

$$\begin{aligned} & J_v^I \cdot B_{vu}^I = -J_u^I \\ \hookrightarrow \quad & J_v^{II} \cdot B_{vu}^{II} = -J_u^{II} - J_v^{II,I} \cdot B_{vuI} \\ \hookrightarrow \quad & J_v^{III} \cdot B_{vu}^{III} = -J_u^{III} - J_v^{III,I} \cdot B_{vu}^I - J_v^{III,II} \cdot B_{vu}^{II} \\ & \hookrightarrow \vdots \end{aligned}$$

et en accélération :

$$\begin{aligned} & J_v^I \cdot b'^I = -\dot{J}\dot{q}^I \\ \hookrightarrow \quad & J_v^{II} \cdot b'^{II} = -\dot{J}\dot{q}^{II} - J_v^{II,I} \cdot b'^I \\ \hookrightarrow \quad & J_v^{III} \cdot b'^{III} = -\dot{J}\dot{q}^{III} - J_v^{III,I} \cdot b'^I - J_v^{III,II} \cdot b'^{II} \\ & \hookrightarrow \vdots \end{aligned}$$

peuvent être réorganisées selon le diagramme de la figure 4.6.

Ce diagramme correspond au cas où la Jacobienne possède une structure bloc triangulaire. Seules les opérations correspondant au traitement des trois premiers blocs sont reprises. Il illustre les différentes phases de traitement des contraintes,

- résolution des équations de contraintes et calcul itératif des coordonnées dépendantes q_v,
- résolution des dérivées premières des contraintes et calcul direct des vitesse dépendantes $\dot{q}_v$,
- résolution des dérivées secondes des contraintes et calcul direct des termes de b'.

et le couplage des différents blocs au travers des transferts de données.

Note : dans le cas d'un modèle dynamique direct, on ne calcule pas explicitement les accélérations dépendantes à ce stade, car ce calcul nécessite les accélérations indépendantes qui ne seront connue qu'après résolution de la forme réduite des équations du mouvement. Néanmoins, comme nous l'avons vu précédemment, le vecteur $b' = -J_v^{-1}\dot{J}\dot{q}$ nécessaire au calcul des accélérations dépendantes, est utilisé dans la phase de réduction.

Sur base de la structure bloc de la Jacobienne, nous proposons de regrouper en différentes tâches, les opérations qui utilisent les éléments de Jacobienne appartenant à une même ligne de bloc. Ce regroupement, illustré par les traits verticaux discontinus sur le diagramme de la figure 4.6, respecte la dépendance relative des différents blocs d'opérations.

On remarque que la quantité de calcul des différentes tâches est malencontreusement déséquilibrée. En effet, la troisième tâche ne peut démarrer avant que les coordonnées dépendantes q_v^I et q_v^{II} des blocs I et II ne soient connues, et c'est elle qui semble comporter le plus de calculs en raison de l'évaluation des blocs sous diagonaux de la Jacobienne. Il n'est toutefois pas possible d'évaluer ces blocs sous diagonaux colonne par colonne en raison de leur dépendance potentielle vis à vis des coordonnées dépendantes associées à leur indice de ligne de bloc. Ainsi le bloc $J_v^{III,I}$ peut dépendre des coordonnées q_v^{III} calculées dans la tâche III et ne peut par conséquent pas être évalué avant cela dans la tâche I ou II.

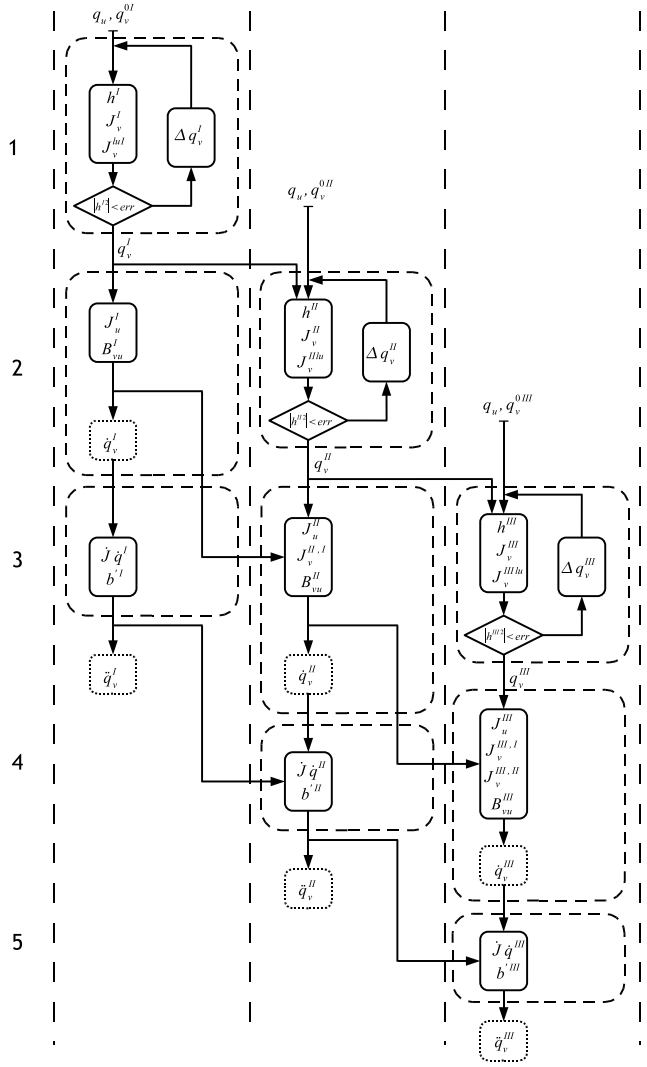


FIG. 4.6 – Diagramme de traitement des équations de contraintes

Si on observe le diagramme avec attention, on constate que lorsque le calcul des coordonnées articulaires q_v^I associées au premier bloc J_v^I de la Jacobienne est terminé dans la première tâche, celles-ci sont transférées à une seconde tâche qui peut alors commencer le calcul des coordonnées q_v^{II} en parallèle avec la première tâche qui poursuit avec le calcul des vitesses généralisées dépendantes $\dot{q}_v^I$.

On notera que le premier bloc B_{vu}^I de la matrice de couplage des vitesses est transmis aux autres tâches. Il s'agit en effet de termes nécessaires pour le calcul des autres blocs de la matrice de couplage des vitesses, B_{vu}^{II}, B_{vu}^{III} ,etc.

Après le calcul des vitesses, chaque tâche procède alors au calcul des termes provenant des dérivées secondes des équations de contraintes. On notera que les sous blocs du vecteur $b^{'}$ sont également transmis de tâche en tâche.

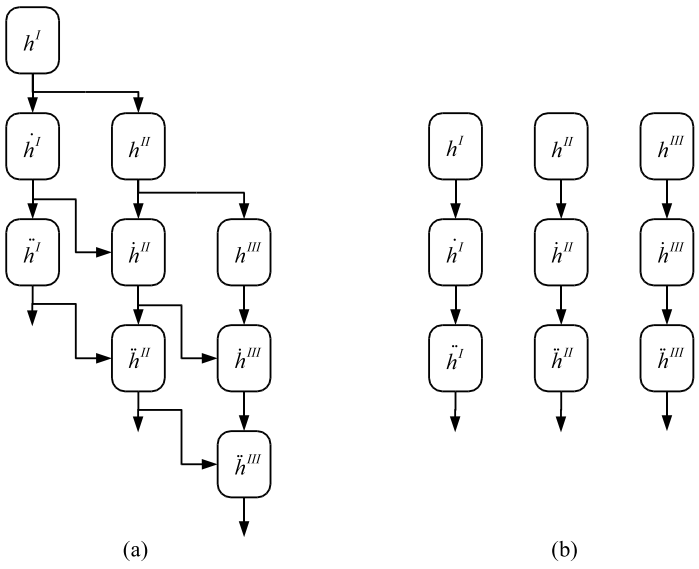


FIG. 4.7 – Schémas de traitement des contraintes, (a) en cascade, (b) en parallèle

La figure 4.7 illustre de façon plus synthétique la structure générale du diagramme de traitement des équations de contraintes. Ce traitement est effectué en cascade lorsque la matrice Jacobienne J_v possède une structure bloc triangulaire. Lorsque J_v possède une structure bloc diagonale, le traitement de chaque bloc de contrainte se fait alors en parallèle de façon totalement indépendante des autres blocs. Ce cas idéal peut se présenter pour différentes suspensions

d'un véhicule ou pour les différentes jambes d'un robot à structure parallèle.

Identification et marquage des équations En pratique, toutes les équations servant à calculer les contraintes et tous les termes de la Jacobienne sont d'abord générées avant de pouvoir procéder à l'analyse et à la factorisation éventuelle de la partie carrée J_v. Il n'est donc pas possible d'implémenter a priori la génération parallèle des blocs de la Jacobienne, ce qui poserait un problème dans le cadre d'une approche numérique.

Lorsque la structure est déterminée après factorisation bloc triangulaire, nous procédons alors de façon récursive, au marquage des équations symboliques, en partant de celles qui définissent les termes de la Jacobienne complète J et des contraintes h, et en remontant le chaînage récursif des variables auxiliaires. Cette procédure de marquage va permettre d'identifier les équations de calcul des termes des différents blocs afin de les séparer a posteriori.

Les termes sont marqués selon l'ordre des blocs. La procédure récursive de marquage d'appartenance à un bloc s'arrête lorsqu'elle arrive sur une variable auxiliaire déjà marquée comme appartenant à un bloc d'indice inférieur. Ainsi les expressions communes sont toujours classées dans les blocs qui seront exécutés les premiers, ce qui compense quelque peu le déséquilibre induit par le regroupement en tâches.

Cette même procédure récursive est utilisée pour le marquage de toutes les autres équations relatives au calcul des blocs de matrices B_{vu}, des vitesses articulaires dépendantes $\dot{q}_v$ et des termes du vecteur $b^{'}$.

Note : Les équations de calcul des termes des blocs sous diagonaux de la matrice Jacobienne $J_v^{M,N}$: $M > N$ ne peuvent pas être évaluées avant que la procédure itérative de résolution du bloc M ait convergé et que les valeurs correctes des coordonnées dépendantes q_v^M soient connues.

4.2.2 Séparation des équations cinématiques et dynamiques

Nous nous intéressons maintenant aux deux blocs correspondant aux équations cinématiques et dynamiques résultant de l'application du formalisme Newton-Euler récursif semi-explicite au graphe topologique arborescent du système. Ces deux blocs issus du diagramme général du modèle dynamique direct illustré sur la figure 4.2 sont repris sur la figure 4.8 pour rappel. Un troisième bloc, illustré en pointillé, correspond au calcul des forces et couples extérieurs.

On se rappelle que dans notre formulation, les forces et couples extérieures f_{ext} et c_{ext} interviennent dans le calcul des équations dynamiques, plus particulièrement dans l'expression du vecteur $c(q, \dot{q}, f_{ext}, c_{ext}, g)$ de l'équation 2.4. Nous commentons la séparation des calculs de ces forces extérieures dans le paragraphe suivant.

La séparation des équations des deux blocs, cinématique progressive et dynamique régressive, repose essentiellement sur la topologie de l'arborescence.

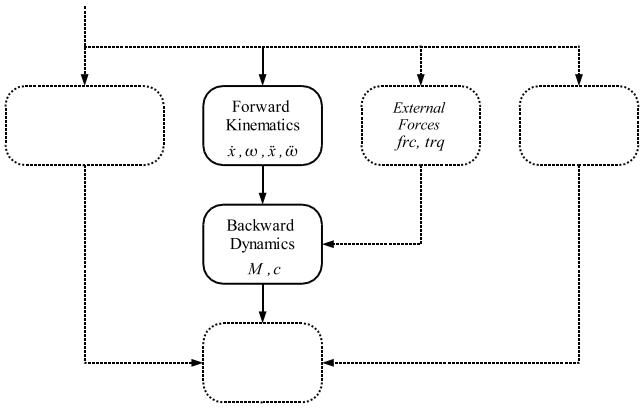


FIG. 4.8 – Équations cinématiques et dynamiques du système arborescent

Nous exploitons le parallélisme présent dans la structure, lequel se retrouve au niveau des équations récursives, comme cela a déjà été montré plus haut à la section 4.1.2. Nous illustrons le cas particulier qui nous occupe sur la figure 4.9.

Les équations relatives aux branches communes de l'arborescence ne sont pas parallélisable, sauf au niveau des opérations.

En ce qui concerne la matrice de masse générée pour une telle topologie, on peut observer la structure suivante :

$$M = \begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & 0 \\ M_{31} & 0 & M_{33} \end{pmatrix}$$

On se rappelle que la matrice de masse est symétrique et que par conséquent $M_{21} = M_{12}^T$ et $M_{31} = M_{13}^T$. Les termes $M_{32} = M_{23}^T$ sont nuls car les branches 2 et 3 n'ont pas d'influence dynamique mutuelle directe. Ces branches sont indépendantes au niveau de la structure arborescente, aucune des deux n'est parent de l'autre. Les termes du bloc M_{22} sont calculés par les équations relatives à la branche 2. Ceux de M_{33} le seront par les équations relatives à la branche 3.

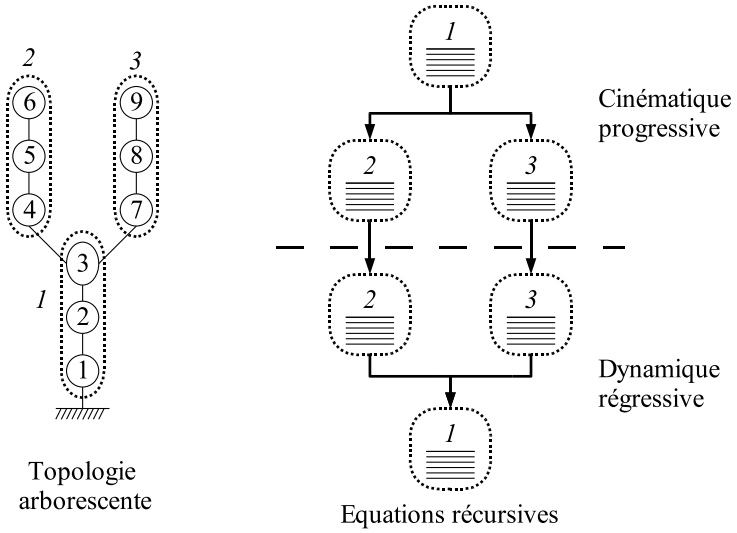


FIG. 4.9 – Parallélisme topologique de la structure et des équations cinématiques et dynamiques

Les équations de la branche 1 fourniront d'abord les termes de M_{31} et de M_{21}, puis ceux de M_{11}.

4.2.3 Calcul des forces extérieures

Lorsque le système est soumis à des forces extérieures f_{ext} provenant de liaisons point-à-point ou d'interactions avec son environnement, il est possible de calculer celles-ci en parallèle avec les équations cinématiques. Dans le cas des forces de liaisons point-à-point, on peut éventuellement profiter d'un certain parallélisme, selon la configuration du système. On notera que les calculs relatifs à plusieurs liaisons ou plusieurs interactions sont également parallélisables. Nous illustrons cela avec le cas des forces de contact.

Forces de liaisons Le cas des liaisons point-à-point est assez similaire au cas des contraintes. La figure 4.10 illustre la relation entre la topologie et la distribution des équations, dans le cas général du calcul d'une liaison entre deux corps.

Dans une première étape, parallèle, on évalue la cinématique des points d'applications des forces sur chaque branche, ensuite on évalue le modèle constitutif de l'élément de liaison. Rappelons que ce modèle peut être plus ou moins complexe selon la nature de cet élément. Finalement, la force de liaison est projetée

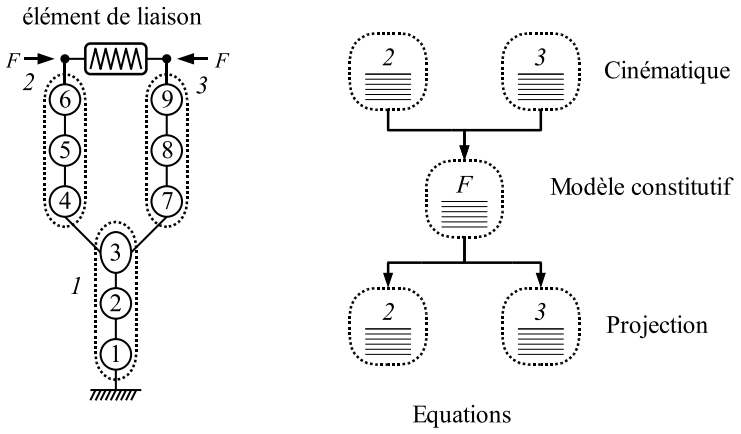


FIG. 4.10 – Equations de calcul de forces de liaison point à point

dans les repères de chacun des corps reliés. On constate qu'il n'y a pas d'équations relatives à la branche 1 parente si on exprime les calculs dans le repère du corps commun aux deux branches parallèles, ce qui est le cas par défaut.

Si la force de liaison ne s'applique pas entre deux branches parallèles ou indépendantes, alors, il n'y a pas, a priori, de parallélisme exploitable au niveau même du calcul des composantes de forces et de couples extérieurs.

Forces de contact Comme nous l'avons exposé dans les sections 1.5.1 et 1.5.3, la procédure consiste à :

- calculer la cinématique du point d'application de la force d'interaction avec l'environnement,
- évaluer un modèle d'interaction qui produit la valeur de la force qui s'applique sur le corps au point de contact,
- projeter la force dans le repère solidaire du corps et éventuellement, calculer les couples de transport.

Pour chaque interaction, la procédure consiste donc essentiellement en un parcours récursif d'une seule chaîne pour chaque point d'application. Il n'y a donc pas de parallélisme de type topologique pour le calcul d'une seule force d'interaction.

Toutefois, si le système a plusieurs interactions avec l'environnement, on pourra généralement les calculer en parallèle. Par exemple pour un véhicule, on peut calculer quasi indépendamment les forces de contacts de chacune des roues. Nous illustrons le diagramme général sur la figure 4.11.

On constate sur ce diagramme que les calculs de la cinématique des différents points de contact peuvent posséder une partie en commun, si il y a une branche

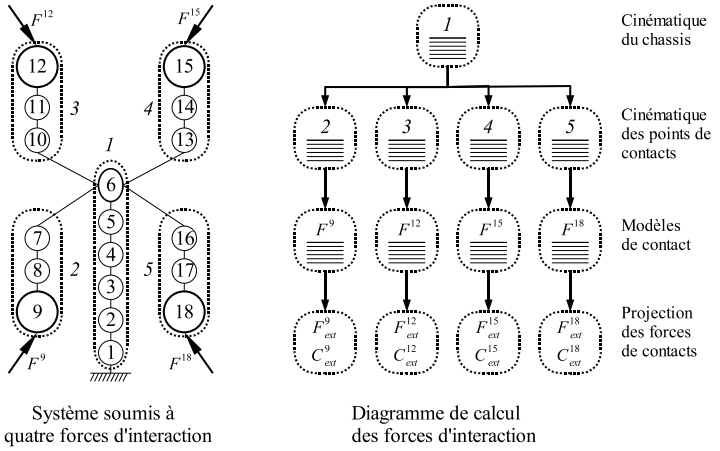


FIG. 4.11 – Diagramme de calcul des forces d'interaction d'un système avec son environnement

commune au niveau de la topologie du système. Ceci est du à l'utilisation du formalisme récursif d'une part et à la nécessité d'exprimer la cinématique des points de contact dans le repère inertiel, d'autre part.

4.2.4 Séparation des opérations de réduction

Les opérations de la procédure de réduction proviennent uniquement de calculs matriciels qui consistent à obtenir les formes réduites de la matrice de masse $\mathcal{M}$ et du vecteur $\mathcal{F}$. Nous rappelons ici les équations 2.45 et 2.46 qui définissent ces éléments :

$$\begin{aligned}
\mathcal{M} &= M_{uu} + M_{uv}B_{vu} + B_{vu}^T M_{vu} + B_{vu}^T M_{vv} B_{vu} \\
F &= c - Q \\
\mathcal{F} &= F_u + M_{uv}\, b' + B_{vu}^T F_v + B_{vu}^T M_{vv}\, b'
\end{aligned}$$

On trouve dans la littérature [62] des procédures de calcul où les différents termes de ces sommes sont évalués en parallèle avant d'être additionnés. Cela crée des calculs de tailles relativement inégales vu la complexité très variable entre ces différents termes, mais chaque terme peut être calculé indépendamment des autres et ceci quelle que soit la structure du système.

Nous proposons d'utiliser à nouveau la structure bloc de la Jacobienne comme guide pour la segmentation des calculs de la matrice de masse $\mathcal{M}$ et du vecteur $\mathcal{F}$ réduits.

Nous avons déjà vu que la matrice B_{vu} de couplage des vitesses, très présente dans les calculs de réduction, peut être obtenue bloc par bloc comme ceci :

$$B_{vu} = \begin{pmatrix} B_{vu}^{I} \\ B_{vu}^{II} \\ B_{vu}^{III} \end{pmatrix} \qquad B_{vu}^{T} = \begin{pmatrix} (B_{vu}^{I})^T & (B_{vu}^{II})^T & (B_{vu}^{III})^T \end{pmatrix}$$

Dans les formules de réduction, on constate que la matrice B_{vu} est multipliée à la matrice de masse M et au vecteur F. Il serait alors intéressant de structurer la matrice M et du vecteur F de la même manière. Ceci ne pose pas de problème : nous avons déjà associé des groupes de coordonnées dépendantes $q_v^I, q_v^{II}, etc.$ à chaque bloc de J_v et donc de B_{vu}. Il nous suffit alors de segmenter les sous matrices de masse M_{vu} et M_{vv} selon le même découpage que la matrice B_{vu}.

La matrice de masse M et le vecteur F peuvent donc être présentés comme ceci :

$$M = \begin{pmatrix} M_{uu} & M_{uv}^{I} & M_{uv}^{II} & M_{uv}^{III} \\ M_{vu}^{I} & M_{vv}^{I} & \otimes & \otimes \\ M_{vu}^{II} & \otimes & M_{vv}^{II} & \otimes \\ M_{vu}^{III} & \otimes & \otimes & M_{vv}^{III} \end{pmatrix} \qquad F = \begin{pmatrix} F_u \\ F_v^I \\ F_v^{II} \\ F_v^{III} \end{pmatrix}$$

On notera que contrairement à ce que l'on observe lorsqu'on restructure la matrice de masse en permutant les coordonnées selon leur appartenance à une même branche, elle ne présente pas nécessairement une structure bloc diagonale pour des blocs de contraintes indépendantes, et donc il n'y a aucune garantie que le sous bloc M_{vv} ne possède que des blocs diagonaux. Les blocs marqués par des $\otimes$ ne sont pas nuls a priori. En pratique, ce sera parfois le cas pour certains

systèmes, car les raisons de l'indépendance des blocs de contraintes sont de nature topologique, notamment lorsque les contraintes proviennent de coupures relatives à des branches indépendantes, par exemple les coupures localisées dans les différentes suspensions indépendantes[4] d'un véhicule.

On peut alors organiser et structurer les calculs permettant d'obtenir les matrices réduites de la façon suivante :

$$\begin{aligned} BtM_u &= B_{vu}^T \cdot M_{vu} \\ &= B_{vu}^{It} M_{vu}^I + B_{vu}^{IIt} M_{vu}^{II} + B_{vu}^{IIIt} M_{vu}^{III} + \cdots \\ &= BtM_u^I + BtM_u^{II} + BtM_u^{III} + \cdots \end{aligned}$$

$$M_{uv} B_{vu} = {BtM_u}^T$$

$$\begin{aligned} BtM_v &= B_{vu}^T \cdot M_{vv} \\ &= \quad B_{vu}^{It} M_{vv}^I + B_{vu}^{It} M_{vv}^{I,II} + B_{vu}^{It} M_{vv}^{I,III} + \cdots \\ &\quad + B_{vu}^{IIt} M_{vv}^{II,I} + B_{vu}^{IIt} M_{vv}^{II} + B_{vu}^{IIt} M_{vv}^{II,III} + \cdots \\ &\quad + B_{vu}^{IIIt} M_{vv}^{III,I} + B_{vu}^{IIIt} M_{vv}^{III,II} + B_{vu}^{IIIt} M_{vv}^{III} + \cdots \\ &\quad + \cdots \\ &= BtM_v^I + BtM_v^{II} + BtM_v^{III} + \cdots \end{aligned}$$

On a tenu compte ici, pour le calcul des blocs de BtM_v de la possibilité de l'existence de termes non nuls hors diagonaux dans M_{vv} ce qui implique la présence des termes $B_{vu}^{it} M_{vv}^{i,j}$ où $j \neq i$.

$$\begin{aligned} BtMB &= (B_{vu}^T M_{vv}) \cdot B_{vu} \\ &= BtM_v^I B_{vu} + BtM_v^{II} B_{vu} + BtM_v^{III} B_{vu} + \cdots \\ &= BtMB^I + BtMB^{II} + BtMB^{III} + \cdots \end{aligned}$$

Il faut être attentif ici au fait que les blocs du terme $BtMB$ sont obtenus en multipliant les blocs de BtM_v par la matrice B_{vu} complète à chaque fois ! Celle-ci doit donc être disponible dans son intégralité pour le calcul de chaque bloc, ce qui nécessite qu'elle soit transmise à chacune des tâches qui auraient à faire ces calculs. Nous reviendrons sur ce point un peu plus loin.

[4] indépendantes du point de vue cinématique, pas dynamique : la présence ou non d'une barre anti-roulis serait sans effet ici

$$\begin{aligned} BtF &= B_{vu}^T \cdot F_v \\ &= B_{vu}^{It} F_v^I + B_{vu}^{IIt} F_v^{II} + B_{vu}^{IIIt} F_v^{III} + \cdots \\ &= BtF^I + BtF^{II} + BtF^{III} + \cdots \end{aligned}$$

$$\begin{aligned} MBMb &= (M_{uv} + B_{vu}^T M_{vv}) b' \\ &= (M_{uv}^I + BtM_v^I) b' + (M_{uv}^{II} + BtM_v^{II}) b' + (M_{uv}^{III} + BtM_v^{III}) b' + \cdots \\ &= MBMb^I + MBMb^{II} + MBMb^{III} + \cdots \end{aligned}$$

On constate que chaque élément de cette séquence de calcul peut être parallélisé. Ainsi on peut obtenir des matrices de masse réduites partielles :

$$\begin{aligned} \mathcal{M}^I &= BtM_u^{I^T} + BtM_u^I + BtMB^I \\ \mathcal{M}^{II} &= BtM_u^{II^T} + BtM_u^{II} + BtMB^{II} \\ \mathcal{M}^{III} &= BtM_u^{III^T} + BtM_u^{III} + BtMB^{III} \\ &\vdots \end{aligned}$$

de façon tout à fait indépendante. En effet aucun de ces blocs n'est nécessaire au calcul d'un autre.

Il en de même pour les forces réduites :

$$\begin{aligned} \mathcal{F}^I &= MBMb^I + BtF^I \\ \mathcal{F}^{II} &= MBMb^{II} + BtF^{II} \\ \mathcal{F}^{III} &= MBMb^{III} + BtF^{III} \\ &\vdots \end{aligned}$$

Finalement, on obtient les matrices globales $\mathcal{M}$ et $\mathcal{F}$ en procédant à l'assemblage des matrices partielles comme ceci :

$$\begin{aligned} \mathcal{M} &= M_{uu} + \mathcal{M}^I + \mathcal{M}^{II} + \mathcal{M}^{III} + \cdots \\ \mathcal{F} &= F_u + \mathcal{F}^I + \mathcal{F}^{II} + \mathcal{F}^{III} + \cdots \end{aligned}$$

On peut finalement illustrer le processus de réduction du système d'équations par le diagramme de la figure 4.12.

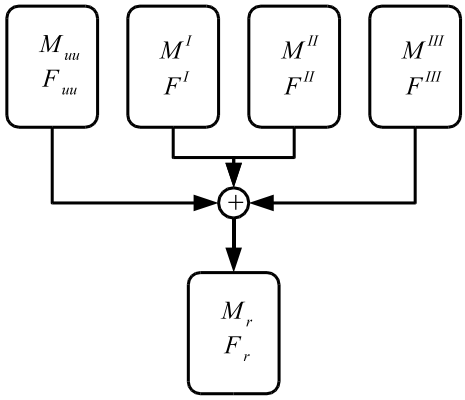


FIG. 4.12 – Distribution des opérations lors de la réduction

On notera que les termes M_{uu} de la matrice de masse et F_u correspondant aux coordonnées indépendantes q_u, peuvent également être distribués dans les groupes d'équations de calcul des différents blocs de $\mathcal{M}$ et $\mathcal{F}$ tout comme c'est le cas pour les éléments de J_u dans la section relative au calcul des blocs de B_{vu}.

4.2.5 Synthèse

Dans les paragraphes qui précèdent nous avons utilisé essentiellement deux clés de segmentation des calculs :

- la *structure algébrique* mise en évidence au niveau de la matrice Jacobienne et
- la *structure topologique* de la représentation arborescente du système, obtenue après coupure des boucles.

Les éléments de base de ces deux modes de segmentation sont respectivement les *blocs* et les *branches*. Ces deux éléments correspondent à des sous groupes de l'ensemble des coordonnées articulaires. Néanmoins, la répartition des coordonnées entre les blocs et les branches sont très différentes.

On constate en pratique que la segmentation selon les branches est plus fine que la segmentation selon les blocs. En effet, chaque bloc rassemble en général plus de coordonnées que chaque branche : en général, deux branches sont impliquées dans une coupure.

Le choix d'une segmentation globale selon les branches conduirait à une parallélisation à grains moyens. Ce mode de segmentation constitue un complément intéressant à la méthode de vectorisation, qui est une méthode de

parallélisation à grains fins. Pour des systèmes à topologie arborescente[5], cette approche constituerait un moyen intéressant de distribuer les opérations entre différents groupes d'unités de calculs constituant une architecture vectorielle du type de celle que nous avons présenté dans la section précédente.

Néanmoins, dans le contexte plus classique du calcul parallèle où on vise des architectures standard telles que celles des ordinateurs multiprocesseurs ou ordinateurs parallèles organisés en grappe, il nous semble plus opportun de considérer une méthode de segmentation à gros grains produisant éventuellement moins de tâches, mais des tâches de taille plus importante et avec un minimum d'interaction entre elles, comme dans le cas de la Jeep Iltis traité dans [64].

Dans cette optique, nous proposons de rechercher un dénominateur commun entre les deux modes de segmentation envisagés. En considérant les coordonnées impliquées dans les différents blocs et les différentes branches, on peut mettre en évidence des relations entre les blocs et les branches. Nous allons utiliser ces relations pour associer des blocs et des branches de manière à obtenir un mode de découpage unique qui pourra être appliqué de façon homogène à l'ensemble des opérations d'un modèle de système articulé et au modèle dynamique présenté ici en particulier.

4.3 La création de tâches parallèles

Après avoir observé dans la section précédente, les différentes façons de découper les différentes étapes d'un modèle dynamique, nous sommes arrivés à la conclusion qu'il serait utile de déterminer une méthode de découpage unique permettant de segmenter l'ensemble du modèle de manière à pouvoir générer des tâches suffisamment importantes et indépendantes pour leur permettre d'être exécutées efficacement sur un ordinateur parallèle classique, tel qu'une machine multiprocesseurs par exemple.

La figure 4.13 illustre les observations faites pour les différentes parties du diagramme d'un modèle dynamique direct.

Notre objectif est donc de trouver une méthode de découpage unique utilisable pour générer des tâches parallèles. Pour cela, nous analysons dans un premier temps la relation qui existe entre les blocs et les branches.

4.3.1 Regroupement de blocs et de branches

On peut associer à chaque branche et à chaque bloc un ensemble de coordonnées articulaires. Nous proposons de regrouper les blocs et les branches qui possèdent des coordonnées en commun. Les blocs font souvent intervenir plus

[5] par comparaison à une topologie purement linéaire

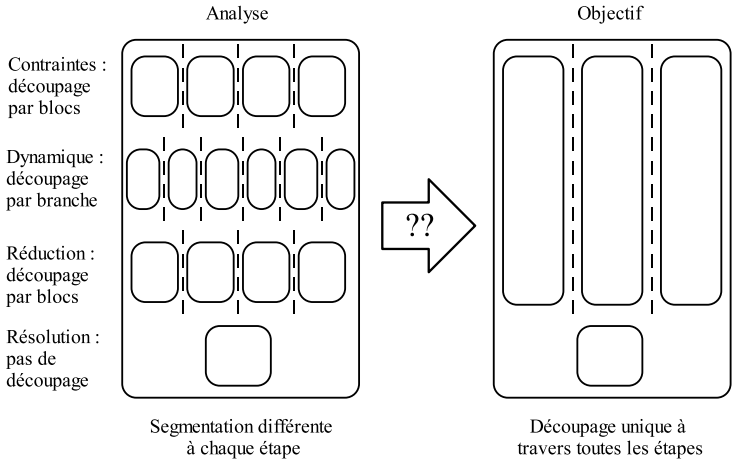


FIG. 4.13 – Segmentation d'un modèle dynamique direct

de coordonnées que les branches, et il est donc courant que plusieurs branches soient associées à un même bloc.

Mais il est également possible que des coordonnées relatives à une même branche interviennent dans deux blocs différents, par exemple dans le cas d'un découplage entre les différentes équations de contrainte scalaires d'une même coupure.

Considérons par exemple le système BA2000 dont la topologie est illustrée sur la figure 4.14. Il s'agit d'un bogie de tram à plancher bas qui circule dans la ville de Bruxelles. Nous n'effectuerons pas de simulation de ce système. Nous utilisons sa topologie particulière dans le seul but d'illustrer notre méthode.

Ce système comporte 23 articulations réparties entre 9 branches comme indiqué dans la table 4.1. Les chiffres entre parenthèses correspondent au numéro de la branche parente pour chacune des neufs branches. Le sol reçoit l'indice 0 par défaut.

Ce système est soumis à des équations de contraintes et après partitionnement des coordonnées et factorisation bloc triangulaire de la matrice Jacobienne associée aux coordonnées dépendantes, on peut observer la structure suivante

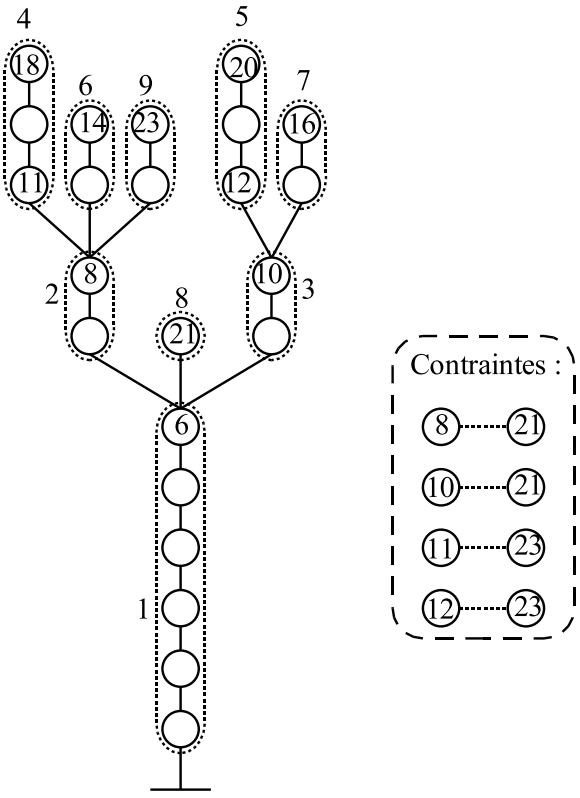


FIG. 4.14 – Topologie du système BA2000

branche	parent	coordonnées
1	(0)	1 2 3 4 5 6
2	(1)	7 8
3	(1)	9 10
4	(2)	11 17 18
5	(3)	12 19 20
6	(2)	13 14
7	(5)	15 16
8	(1)	21
9	(2)	22 23

TAB. 4.1 – Répartition des articulations entre les branches

de la matrice J_v :

$$q_v = \{\, 23 \quad 21 \quad 22 \quad 7 \quad 8 \quad 11 \quad 12 \,\}$$

$$J_v = \begin{pmatrix} \bullet & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \bullet & \cdot & \cdot & \cdot & \cdot & \cdot \\ \bullet & \cdot & \bullet & \bullet & \bullet & \cdot & \cdot \\ \bullet & \cdot & \bullet & \bullet & \bullet & \cdot & \cdot \\ \cdot & \bullet & \cdot & \bullet & \bullet & \cdot & \cdot \\ \cdot & \cdot & \bullet & \cdot & \cdot & \bullet & \cdot \\ \bullet & \cdot & \bullet & \bullet & \bullet & \cdot & \bullet \end{pmatrix}$$

La répartition des coordonnées articulaires dépendantes entre les blocs est données par la table 4.2. Les chiffres entre parenthèses correspondent aux numéro des autres blocs qui doivent nécessairement être résolu avant.

bloc	dép.	coordonnées
1	·	23
2	·	21
3	(1, 2)	22 7 8
4	(3)	11
5	(1,3)	12

TAB. 4.2 – Répartition des coordonnées dépendantes entre les blocs

On peut alors relier les branches et les blocs qui possèdent des coordonnées en commun. On peut illustrer cela sous forme de graphe bipartite, comme sur la

figure 4.15 et ensuite utiliser la matrice d'adjacence de ce graphe pour effectuer le regroupement des blocs et des branches.

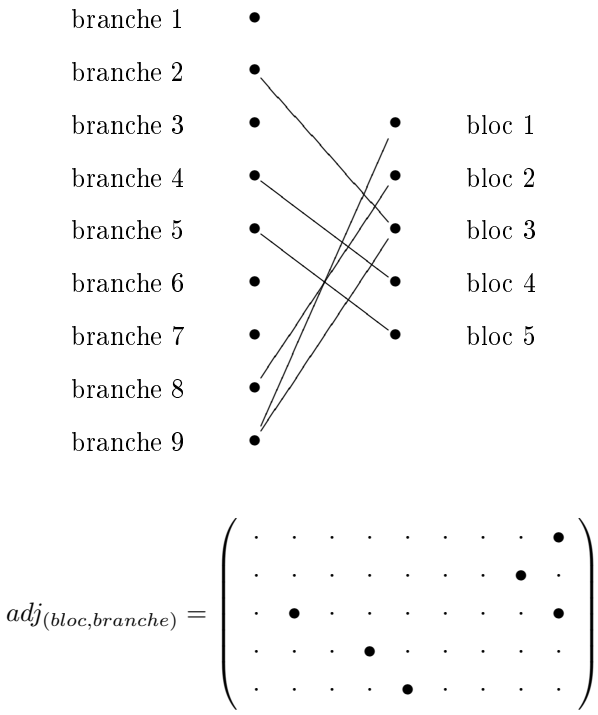


FIG. 4.15 – Connexité entre branches et blocs

On obtient donc les groupes de branches et de blocs suivants :

1. blocs 1 et 3 avec branches 2 et 9
2. bloc 2 avec branche 8
3. bloc 4 avec branche 4
4. bloc 5 avec branche 5

Après cette phase de regroupement, il reste souvent des branches libres. C'est évidemment toujours le cas pour un système arborescent qui n'est pas soumis à des contraintes, car alors, il n'y a pas non plus de matrice Jacobienne, ni de blocs auxquels associer des branches.

Les quatre branches 1, 3, 6 et 7 non pas été associées à des blocs. On les appelle des branches libres à ce stade. Elles sont donc disponibles pour être associées avec d'autres branches libres ou non et former de nouveaux groupes ou rejoindre des groupes existant.

4.3.2 Regroupement des branches libres

Pour regrouper les branches libres, nous allons utiliser la relation de parenté qui existe entre elles. La numérotation des branches respecte toujours leur dépendance : les indices h et i de deux branches dont la branche h est le parent de la branche i, sont tels que $h < i$. Ainsi, en parcourant la liste des branches libres dans l'ordre des indices des branches, on rencontrera toujours la branche parent h avant la branche i.

Nous avons constaté en observant la relation entre la topologie des branches et la topologie des équations, que la même dépendance se retrouvait des deux côtés lorsque les équations provenaient de l'utilisation d'un formalisme récursif progressif, ce qui est le cas de la grande partie des équations présentes dans nos modèles.

Il est alors naturel de regrouper des branches libres consécutives, qui forment une chaîne, car nous savons que les calculs relatifs à une branche i ne peuvent pas être exécutés avant que ceux relatifs à la branche parent h n'aient été effectués. En revanche, deux branches i et j ayant la même branche parent h peuvent se retrouver dans deux groupes d'exécution différents, car nous savons que les calculs relatifs à ces deux branches i et j peuvent être effectués en parallèle. Nous ne devons donc regrouper qu'une seule des deux branches i ou j avec la branche parent h. On mémorise pour chaque branche si une de ses branches enfants libres lui a déjà été associée ou non. Chaque branche qui est associée à un parent, libre ou non, perd sa caractéristique de branche libre.

Après avoir parcouru l'ensemble des branches libres, et les avoir éventuellement associées à leur parents, on connaît alors le nombre de tâches que l'on peut générer pour le système. Ce nombre n'est toutefois pas le nombre maximal, même au niveau du parallélisme des branches, car nous ne prenons pas en compte la possibilité de calculer les forces extérieures en parallèle avec la cinématique progressive, comme on l'a vu précédemment sur la figure 4.8. Le nombre de tâches obtenu ici vaudra donc au maximum le nombre d'extrémités du graphe topologique arborescent, soit le nombre de corps terminaux du système, après coupures. Le nombre de tâches obtenu peut évidemment être inférieur au maximum si plusieurs branches parallèles ont été associées de par leur implication dans un même bloc de contraintes.

A l'issue de cette seconde phase de regroupement, nous obtenons les cinq associations de branches suivantes :

0. branches 1, 3, 7
1. branches 2, 6, 9 (blocs 1 et 3)
2. branche 8 (bloc 2)
3. branche 4 (bloc 4)
4. branche 5 (bloc 5)

La figure 4.16 permet de visualiser ces regroupements au niveau de la topologie du système BA2000.

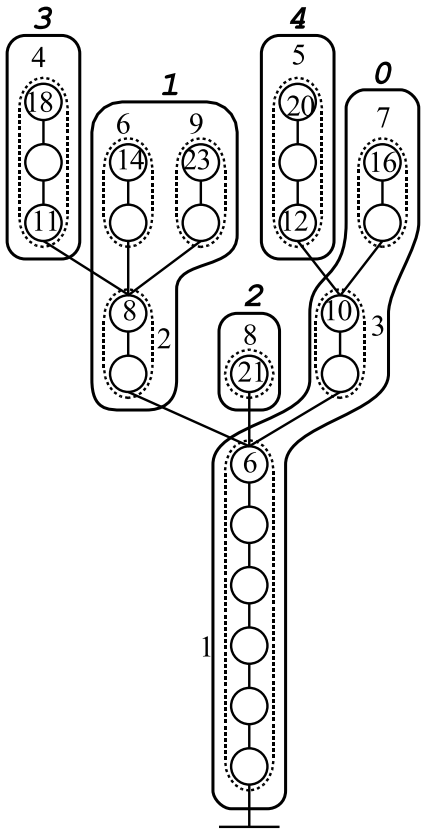


FIG. 4.16 – Association *a priori* des branches du BA2000

La première branche reliée au sol se trouve généralement dans le premier groupe dont l'indice est 0. C'est dans ce groupe que se retrouvent les opérations non parallélisées comme la résolution finale. Les groupes d'indices 1 à 4 correspondent aux 4 groupes formés par les branches impliquées dans les blocs de contraintes. Les groupes restant sont formés par les associations d'autres branches libres entre elles. Il n'y en a pas ici car toutes les branches libres faisaient partie d'une même chaîne qui a formé le groupe d'indice 0 avec la première branche.

Il est important de réaliser que les associations de branches présentées ici sont basées uniquement sur l'analyse de la matrice Jacobienne et la topologie.

Elles pourraient donc être déterminées avant même que les équations soient générées, hormis celles qui permettent d'obtenir les expressions des éléments de la matrice Jacobienne, bien entendu. On peut donc qualifier ces associations de branches de regroupements *a priori*. Ces regroupements constituent la première étape de la création des tâches.

D'autres regroupements plus complexes, mais intuitifs, pourraient éventuellement être effectués :

- On pourrait regrouper les branches impliquées dans des blocs de contraintes avec leur branches parentes, même si celles-ci font déjà partie d'un autre bloc indépendant. En effet, si les équations de branches appartenant à des blocs de contraintes différents peuvent être traitées en parallèle au niveau du traitement des contraintes, ce ne sera pas le cas au niveau du traitement des équations cinématiques si il y a un lien de parenté entre les branches des différents blocs, comme nous l'avons vu précédemment. Donc, on pourrait trouver inutile de créer deux tâches différentes si on croit ne pouvoir exploiter du parallélisme qu'à un seul niveau.
- On pourrait aussi vouloir regrouper systématiquement toutes les branches avec leur branches parentes. Ceci produirait donc un ensemble unique pour toutes les branches d'une même arborescence. On ne pourrait alors générer des tâches différentes que pour des systèmes constitués d'arborescences parallèles, et à condition que les blocs de contraintes ne font pas intervenir des branches appartenant à deux arborescences différentes. Seuls les modèles des systèmes composés à partir de sous systèmes ou ayant une structure fortement parallèle tels les robots du même nom, pourraient alors être segmentés. Cette option de regroupement est très efficace car elle peut produire directement une tâche par sous système.

Nous avons choisi de limiter les regroupements a priori à ceux que nous avons présentés plus haut. Les équations sont alors marquées de manière à pouvoir être regroupées en différents prototypes de tâches.

Marquage des équations Chaque équation symbolique reçoit un code d'identification qui va permettre de la stocker dans une structure hiérarchique arborescente à six niveaux. Ce code d'identification à six champs est comparable à l'adresse de résidence d'une personne qui comporte le pays, la province, la ville, la rue, le numéro, la boite.

Les six champs de notre code d'identification correspondent aux caractéristiques suivantes des équations :

1. numéro de branche
2. numéro de sous séquence
3. numéro de séquence
4. numéro de bloc de la Jacobienne J_v

5. numéro de section
6. numéro de tâche

Chaque champ correspond a une classification soit séquentielle soit parallèle : les numéros de branches, de bloc et de tâche correspondent à différents niveaux de parallélisme. Le numéro de section indique l'appartenance à l'ensemble des équations de traitements de contraintes ou de calcul des forces extérieures ou de réduction, etc. Les numéros de séquence et de sous séquence correspondent aux différentes étapes de calcul de chaque section.

Tous les champs ne sont pas utilisés pour toutes les équations. Par exemple, le numéro d'appartenance à un bloc n'a pas de sens pour une équation de la section relative au calcul des forces externes.

La structure hiérarchique de stockage des équations permet d'accéder de façon rapide et facile aux groupes d'équations ayant une caractéristique commune. Elle est utilisée entre autres pour le traitement des contraintes, l'élaboration des tâches parallèles et la gestion de l'impression finale des équations.

Ce marquage est effectué *avant* l'application des nouveaux traitements symboliques qui sont effectués après la génération et que nous avons présentés dans la section 1.6.2. Ceux-ci peuvent en effet contribuer à améliorer la répartition des opérations entre les tâches de manière à réduire le nombre de données transférées entre les tâches, comme nous allons le voir ci-après.

Contribution des traitements symboliques Des traitements symboliques appliqués aux équations après la génération peuvent avoir un effet sur la distribution des équations entre les différentes tâches. Il s'agit de

- la décomposition des équations,
- de la recherche des doublons,
- de l'élimination des variables auxiliaires inutiles.

Décomposition des équations La décomposition des équations a pour effet d'obtenir un ensemble d'équations qui ne comportent chacune plus qu'une seule opération arithmétique. Il y a alors une certaine identité entre les désignations équation, variable auxiliaire et opération.

La figure 4.17 illustre une situation dans laquelle une équation C provient de la décomposition de D et donc D est normalement le seul enfant de C. A priori C devrait appartenir par défaut à la même tâche que D. Mais si l'analyse montre que les deux parents A et B de l'équations C appartiennent tous deux à une même tâche qui est différente de celle de C, on va faire migrer l'équation C vers la même tâche que ses parents. Cette opération peut réduire d'une unité le nombre de données à transférer entre ces deux tâches.

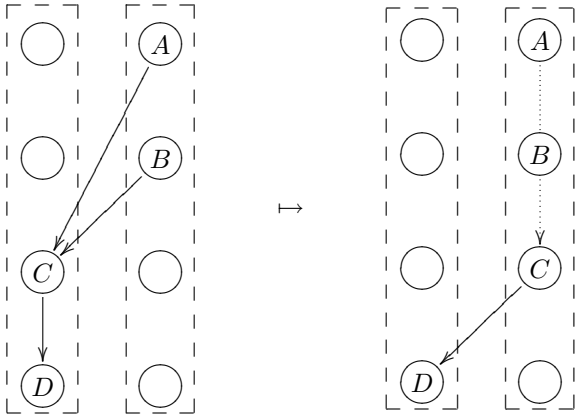


FIG. 4.17 – Changement de tâche après décomposition

Recherche des doublons Après décomposition des équations, il est habituel d'appliquer une recherche des équations qui sont identiques et d'éliminer les doublons pour éviter d'effectuer des calculs redondants.

Dans le contexte du calcul parallèle, il est généralement préférable de recalculer des valeurs pour éviter de devoir les transférer d'une tâche à une autre, car les transferts sont parfois plus lent que les calculs, surtout lorsqu'il s'agit de petites quantités de calculs. C'est précisément ce cas qui est illustré sur la figure 4.18. On ne va pas éliminer ($\not\rightarrow$) le calcul C' identique à C si ils sont localisés dans des tâches différentes, afin d'éviter de créer un nouveau transfert entre les deux tâches, ou du moins d'y rajouter une donnée si un tel transfert existait déjà entre les deux tâches.

Élimination des variables auxiliaires inutiles La dernière de cette série de traitements symbolique consiste à supprimer des variables auxiliaires inutiles en réintégrant les expressions qui les définissent dans celles de leur enfant unique.

La figure 4.19 illustre une situation où une variable C ne possède qu'un seul enfant D qui est situé dans une tâche différente. Cette situation peut être la conséquence naturelle de la migration de C suite à la décomposition de D. Il s'agit alors de maintenir C dans sa tâche afin de ne pas augmenter le nombre de données transférées entre les deux tâches.

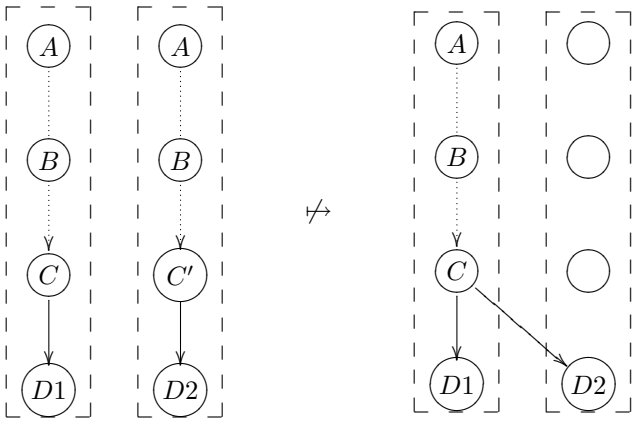


FIG. 4.18 – Recherche et élimination des doublons

4.3.3 Constitution et analyse des prototypes de tâches

La construction de la structure hiérarchique de stockage des équations sur base de leur code d'identification va en fait fournir implicitement la composition des prototypes de tâches. En effet, chaque noeud de niveau six de la structure correspond en fait à une tâche.

Chaque tâche contient toutefois un ensemble d'équations qui appartiennent à des sections différentes et à des séquences différentes au sein de ces sections. Ces équations peuvent en outre dépendre d'autres équations appartenant aux autres tâches. Nous allons donc devoir analyser l'interdépendance de ces équations comme nous l'avons fait lors de la vectorisation des équations. Les entités d'exécution élémentaires correspondaient alors aux opérations arithmétiques contenue dans les équations décomposées. Maintenant, les entités d'exécution seront des groupes d'équations appartenant à une même séquence, c'est à dire les équations appartenant à un même noeud de niveaux trois de notre structure hiérarchique. Une séquence est donc un groupe d'équations dont les champs 3, 4, 5 et 6 de leurs codes d'identification ont respectivement la même valeur pour chaque équation.

Le niveau trois correspond au plus haut niveau où les groupes d'équations n'ont pas de dépendance mutuelle, ce qui signifie que le graphe de dépendance construit en prenant ces groupes de niveau trois comme noeuds ne contient pas de cycles, ni d'arcs bidirectionnels. Choisir un niveau inférieur est inutile car les groupes d'équations seraient alors plus petits et le nombre d'interdépendances plus grand, ce qui conduirait à un nombre de points de connexions plus important entre les tâches.

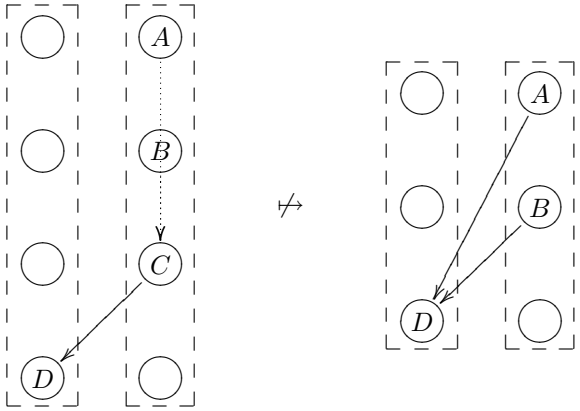


FIG. 4.19 – Conservation d'une variable auxiliaire C *inutile*

Pour chaque séquence d'équation, on va établir une liste de dépendance par rapport aux autres séquences sur base de l'interdépendance entre les équations qu'elles contiennent. Il s'agit donc de créer au niveau des séquences l'équivalent des listes de parents et d'enfants au niveau des équations. Toutefois, au niveau des séquences nous créons deux listes de transferts, une liste d'arrivées et une liste de départs. Chaque transfert contient l'ensemble des données à transférer ainsi que la séquence d'origine et la séquence de destination. Un transfert correspond toujours à un échange unidirectionnel de données entre deux groupes d'équations différents de niveau trois, indépendamment de leur appartenance à une même tâche ou non.

L'analyse d'interdépendance des séquences permet également de déterminer des valeurs d'indice `asap` et `alap` pour chaque séquence, exactement comme nous l'avions fait pour chaque opération dans la méthode de vectorisation. Ces indices vont être utilisés pour déterminer l'ordre d'exécution des séquences pour chaque tâche.

4.3.4 Ordonnancement des séquences d'équations

L'ordre d'exécutions des séquences d'équations doit respecter leur interdépendance, c'est évident. On ne peut pas évaluer une équation avant que toutes les variables auxiliaires nécessaires n'aient été calculées. Toutefois, il y a plusieurs possibilités qui respectent cette dépendance, nous l'avons déjà montré lors de la vectorisation. C'est à nouveau le cas ici.

A priori, toutes les équations sont générées dans un ordre qui respecte cette dépendance. Lors de la création de la structure hiérarchique, cet ordre est res-

pecté à chaque niveau. Toutefois, cet ordre peut ne pas être optimal lorsque les équations sont réparties entre différentes tâches parallèles.

Considérons la situation illustrée par la figure 4.20 où on voit à gauche un ensemble de 6 groupes d'équations A, B, C, F, E, F distribuées entre deux tâches. On constate que la dépendance des groupes est respectées, mais le résultat lors de l'exécution des tâches en parallèle est inefficace si l'ordre est maintenu tel quel au sein des tâches.

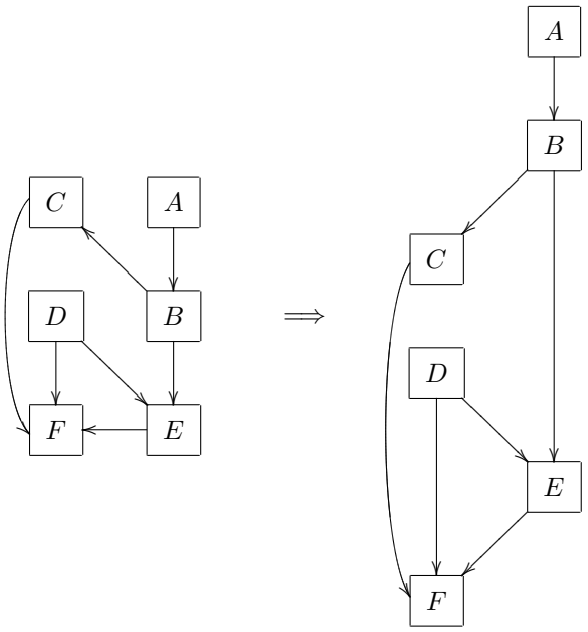


FIG. 4.20 – Exécution en parallèle sans ordonnancement

L'application d'une procédure d'ordonnancement, telle que celle utilisée dans la méthode de vectorisation permet d'optimiser l'ordre d'exécution au sein des tâches en fonction de l'interdépendance des groupes d'équations. La figure 4.21 montre le résultat de l'ordonnancement. L'ordre de C et D a été inversé au sein de la première tâche, ce qui résulte en un schéma d'exécution plus compact où le parallélisme est mis à profit et le temps de calcul global est diminué.

La procédure d'ordonnancement détermine donc un nouvel ordre d'exécution des séquence d'équations au sein de chaque tâche, en fonction des indices `asap` et `alap` ainsi que de la dépendance de cette séquence vis-à-vis des autres séquences. Un numéro d'étape est affecté à chaque séquence en fonction du

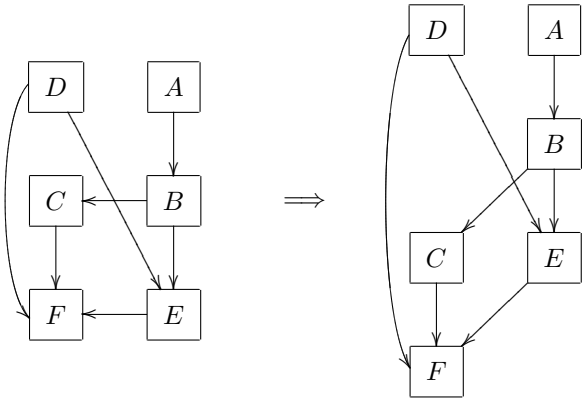


FIG. 4.21 – Exécution en parallèle avec ordonnancement

nombre de séquences qui sont exécutées avant elle au sein de la même tâche.

Longueur critique A la fin de la procédure, on peut également déterminer la longueur critique du schéma d'exécution parallèle. Cette longueur est exprimée en nombre d'opérations et correspond a la somme des opérations des séquences du chemin le plus long [6] qui traverse le graphe de dépendance.

Cette longueur critique vaut au maximum le nombre total d'opérations du modèle et au minimum la longueur de la tâche contenant le plus d'opérations. On parlera aussi de nombre critique d'opérations pour désigner la longueur critique.

En notant N_A le nombre d'opérations de la séquence d'équations A, on peut exprimer la longueur critique du schéma d'exécution parallèle de la figure 4.21 comme $max(N_A + N_B, N_D) + max(N_C, N_E) + N_F$.

Si on pouvait faire abstraction du temps de communication entre les différentes tâches, le rapport entre la longueur critique du schéma parallèle et le nombre total d'opérations du modèles correspondrait au facteur de réduction du temps de calcul obtenu grâce à la parallélisation du modèle.

On appellera *taux de parallélisation* du modèle, le rapport entre le nombre total d'opérations et le produit de la longueur critique du modèle parallèle par le nombre de tâches. Un taux de parallélisation de 100% correspondrait alors au cas idéal où toutes les tâches ont la même taille égale à la longueur critique.

Nombre critique de transferts Une autre caractéristique du schéma d'exécution est le nombre critique de transferts. Il s'agit du nombre maximum de

[6] en terme de nombre d'opérations à effectuer

transferts entre tâches que l'on peut trouver sur un chemin qui traverse le graphe de dépendance.

Résultats A ce stade nous avons obtenu, pour notre système BA2000, le schéma d'exécution parallèle en cinq tâches qui est illustré sur la figure 4.22.

Les rectangles correspondent aux séquences d'équations. Chaque colonne contient les séquences d'équations d'une même tâche. Dans chaque rectangle on trouve les quatre champ du code d'identification de chaque séquence ainsi que le nombre d'opérations entre parenthèses. Les flèches représentent les transferts de données entre les tâches.

Le nombre total d'opérations de ce modèle est de 7557 et la longueur critique de ce schéma d'exécution parallèle est de 3684, ce qui correspondrait à un facteur de diminution[7] du temps de calcul de 2.05 en distribuant les calculs entre cinq tâches parallèles, ce qui correspond à un taux de parallélisation de 41%.

Le nombre critique de transfert vaut 8.

Les caractéristiques des tâches sont données par la table 4.3.

numéro de tâche	0	1	2	3	4
nombre d'opérations	2517	2599	118	982	1373
transferts sortants	28	34	9	10	7
transferts entrants	25	22	4	18	19

TAB. 4.3 – Caractéristiques des tâches BA2000/5P

La table 4.4 reprend quant à elle les nombres de transferts entre les tâches. Les lignes correspondent aux origines et les colonnes aux destinations. Les valeurs entre parenthèses correspondent aux transferts de données internes entre les différentes séquences d'équations d'une même tâche.

	0	1	2	3	4
0	(25)	11	2	5	10
1	10	(59)	2	13	9
2	3	6	(7)	0	0
3	5	5	0	(19)	0
4	7	0	0	0	(19)

TAB. 4.4 – Nombres de transferts entre tâches BA2000/5P

L'analyse de ces résultats montre que :

[7] dans des conditions idéales où on néglige le temps de communication entre tâches

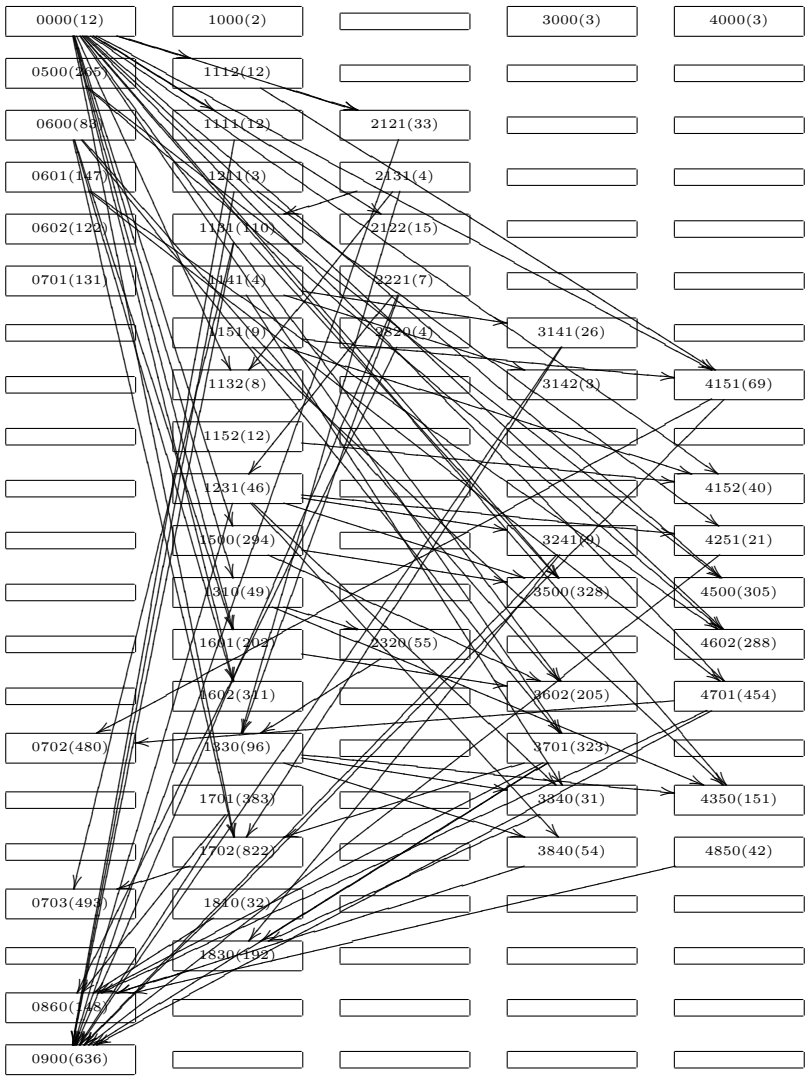


FIG. 4.22 – Modèle dynamique direct du bogie BA2000 : Schéma préliminaire d'exécution parallèle

- les tailles des tâches sont relativement inégales,
- le nombre de transfert entre les tâches est très élevé.

Ces deux observations indiquent que la distribution *a priori* des séquences d'équations ne permet pas d'obtenir directement des schémas parallèles très intéressant.

Il reste encore à réduire le nombre de transferts de données et à égaliser les tâches.

4.3.5 Réduction du nombre de transferts

Regroupement des tranferts Lorsque les séquences d'équations sont ordonnancées, il est possible de réduire le nombre de transferts entre les tâches en les regroupant. Le nombre de données transférées n'est pas réduit, mais des données qui était échangées entre deux tâches en plusieurs transferts sont rassemblées et transférées en une seule fois.

Origine Commune Le premier type de regroupements consiste à trouver une série de transferts partant d'une même séquence et à destination de séquences appartenant à une même tâche. On peut alors rassembler toutes les données et les envoyer toutes ensemble par le premier transfert de la série. Cette situation est illustrée sur la figure 4.23.

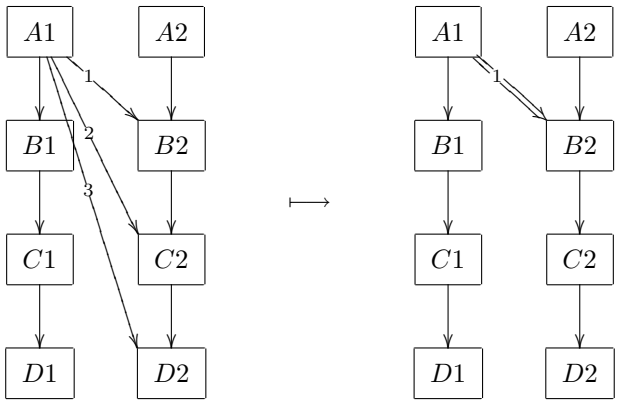


FIG. 4.23 – Regroupement des transferts de même origine

Destination Commune Le second type de regroupement consiste à trouver une série de transferts provenant de différentes séquences d'une même tâche

et arrivant tous à la même séquence. On peut alors rassembler toutes les données et les envoyer toutes ensemble par le dernier transfert de la série. Cette situation est illustrée sur la figure 4.24.

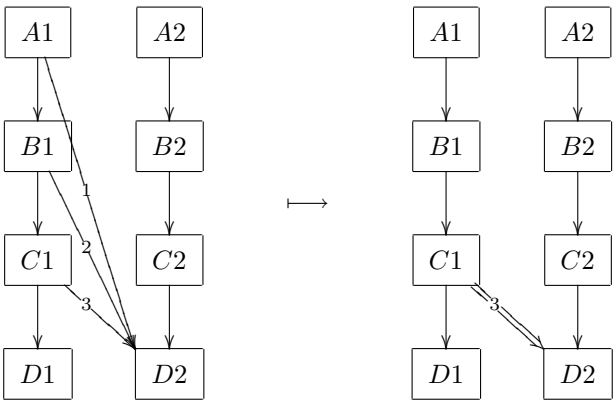


FIG. 4.24 – Regroupement des transferts de même destination

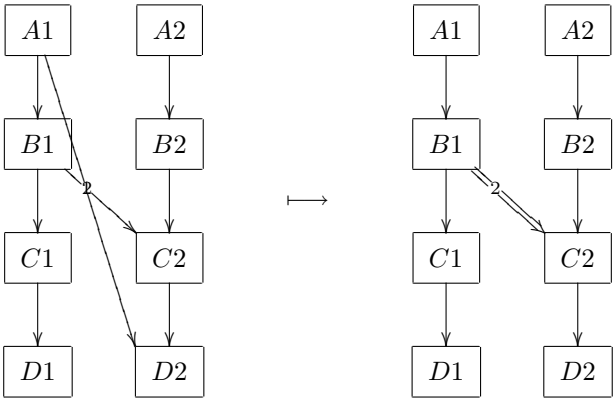


FIG. 4.25 – Regroupement des transferts temporellement incohérents

Incohérence temporelle Un dernier cas intéressant est celui des transferts pouvant conduire à une incohérence temporelle. Il s'agit de transferts provenant de séquences différentes appartenant à une même tâche et destinés à deux séquences différentes d'une même tâche, qui est différente de la tâche

d'origine. On les appelle transferts temporellement incohérents lorsque l'ordre des destinations est l'inverse de celui des origines. En se référant à la figure 4.25 qui illustre cette situation, on peut remarquer que les deux transferts se croisent dans le temps et que le premier transfert, $A1 \rightarrow D2$ est initié avant le second $B1 \rightarrow C2$, alors que le second doit être terminé avant le premier. Cette situation peut causer un blocage lors de l'exécution si on utilise des fonctions de communication qui attendent une confirmation de réception avant de se terminer. La solution est simple et consiste à regrouper les deux transferts pour éviter ce genre de problème à l'exécution. Il est toutefois utile de savoir que cette situation n'est pas une erreur et que l'utilisation de fonctions de communication non blocantes ne nécessite pas de regrouper ces transferts.

Remarques L'intérêt de ces regroupements est discutable. D'une part la réduction du nombre de transferts va entraîner une réduction du nombre d'appels à des instructions de communications entre les tâches et donc un gain en temps d'exécution. D'autre part les regroupements éliminent systématiquement des transferts non critiques pour charger le seul transfert éventuellement critique de la série, ce qui peut augmenter le temps d'une communication critique et ainsi le temps d'exécution.

En pratique, ce sont les caractéristiques de l'architecture parallèle qui dicteront si ces procédures de réduction du nombre de transferts sont intéressantes ou non. L'utilisateur aura donc le choix de l'appliquer ou non.

De plus ces regroupements ne diminuent pas le nombre critique de transferts.

Résultats sur BA2000 Après l'application de ces procédures de regroupement des transferts, le schéma d'exécution parallèle de notre système BA2000, devient tel qu'illustré sur la figure 4.26

Les nombres de transferts entrants et sortant restant dans chaque tâche sont donnés dans la table 4.5. On peut également constater les taux de diminution des nombres de transferts pour chaque tâche après application de la méthode.

numéro de tâche	0	1	2	3	4
transferts sortants	8	14	5	4	3
transferts entrants	10	7	3	7	7
taux de diminution	34%	37.5%	61.5 %	39 %	38%

TAB. 4.5 – Caractéristiques des tâches BA2000/5P

Les nombres de transferts entre tâches sont repris dans la table 4.6. On constate que les nombres des transferts internes aux tâches ne sont pas modifiés par la procédure.

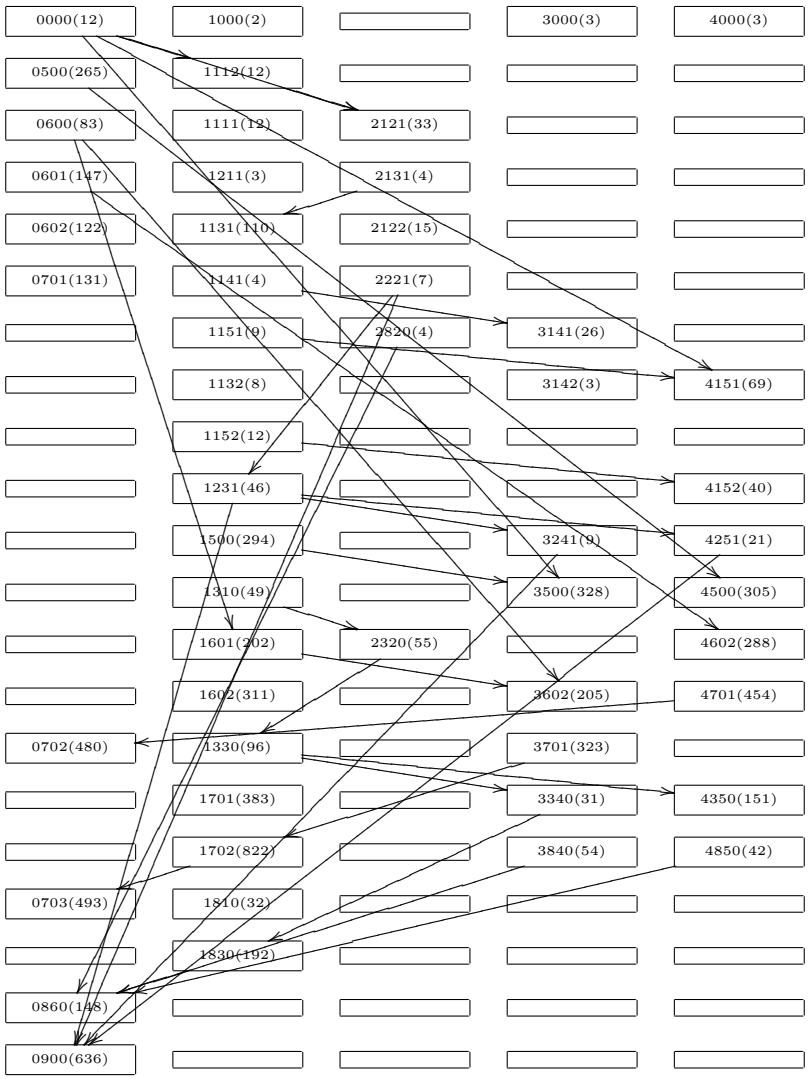


FIG. 4.26 – Modèle dynamique direct du bogie BA2000 : Schéma d'exécution parallèle après réduction du nombre de transferts

	0	1	2	3	4
0	(25)	2	1	2	3
1	3	(59)	2	5	4
2	2	3	(7)	0	0
3	2	2	0	(19)	0
4	3	0	0	0	(19)

TAB. 4.6 – Nombres de transferts entre tâches BA2000/5P après regroupements

Réplication d'équations Une autre procédure de réduction du nombre de transfert consiste à dupliquer certaines séquences d'équations de telle sorte qu'une tâche qui a besoin de valeurs calculées par une autre tâche les recalcule elle même plutôt que d'attendre de les recevoir. Cette procédure peut s'avérer avantageuse lorsque les calculs prennent moins de temps que le transfert de leur résultats. Toutefois, cette procédure n'a pas encore été implémentée au moment de la rédaction de ce texte.

4.3.6 Regroupement des prototypes de tâches

Les règles élaborées sur base des considérations topologiques et de la structure de la Jacobienne pour le regroupement *a priori* des équations, ne produisent pas nécessairement des tâches prototypes de poids équivalents. Nous proposons d'examiner la taille de la tâche la plus petite et de l'associer avec une autre tâche si sa taille est inférieure à une fraction k de la taille moyenne des tâches.

La valeur de la fraction k de la moyenne qui sert de seuil de décision à l'élimination d'une tâche doit être choisie avec précaution. Une valeur trop proche de l'unité provoquerait l'élimination quasi systématique de la plus petite tâche et ceci jusqu'à ce que toutes les tâches aient une taille égale à la moyenne ou bien jusqu'à ce qu'il ne reste plus qu'une seule tâche. Nous utilisons la valeur de 0.5 choisie de façon empirique.

Lorsqu'une tâche a une taille trop petite, plusieurs possibilités de regroupement peuvent être envisagées. Nous en considérons deux :

1. regroupement avec la tâche avec laquelle elle communique le plus,
2. regroupement avec la seconde plus petite tâche.

Le choix d'intégrer les équations de la plus petite tâche dans la tâche avec laquelle elle échange le plus de donnée est motivée par un objectif de réduction du nombre de transferts. Ce critère de choix un peu particulier trouve une justification dans le fait que un nombre important de transfert entre deux tâches exprime une forte interdépendance et donc probablement aussi un faible paral-

lélisme entre elles. Toutefois ce type de regroupement ne conduit généralement pas à l'égalisation des tailles des tâches, mais souvent au contraire.

Le choix de regrouper la plus petite tâche avec la seconde plus petite est naturellement plus pertinent en terme d'équilibrage des tâches. Elle ne peut conduire également qu'à une diminution du nombre de transferts mais généralement pas aussi importante.

Résultats Dans le cas du système BA2000, on remarque que la tâche 2 est la plus petite et a une taille de 118 ce qui est 12.8 fois plus petit que la taille moyenne qui vaut 1517. Cette tâche 2 correspond à la branche 2 de la figure 4.16. Cette branche 2 est une branche fille de la branche 1 qui appartient précisément à la tâche 1, avec laquelle la tâche 2 est la plus fortement connectée du point de vue des transferts de données comme on peut le voir en examinant la table 4.6 qui contient les nombres de transferts entre tâches.

Après ce regroupement, la taille moyenne des tâches devient 1897, soit moins de 2× la taille de chacune d'entre elles et donc aucun autre regroupement n'aura lieu sur base du critère de taille.

Regroupement 1 Nous observons ici le résultat du regroupement de la tâche 2 avec la tâche 1.

Les caractéristiques des quatre tâches restantes sont données par la table 4.7.

numéro de tâche	0	1	2	3
nombre d'opérations	2517	2717	982	1373
transferts sortants	7	12	4	3
transferts entrants	8	4	7	7

TAB. 4.7 – Caractéristiques des tâches BA2000/4P avec regroupement des transferts

La table 4.8 reprend quant à elle les nombres de transferts entre les quatre tâches.

La longueur critique du schéma d'exécution parallèle sur quatre tâches est maintenant de 3776, soit une augmentation quasiment égale au nombre d'opérations de la tâche éliminée.

Le nombre critique de transferts est quant à lui descendu à 4.

Le schéma d'exécution en quatre tâches est illustré sur la figure 4.27

Regroupement 2 Nous observons ici le résultat du regroupement de la tâche 2 avec la tâche 3 qui est la seconde plus petite avec 918 opérations.

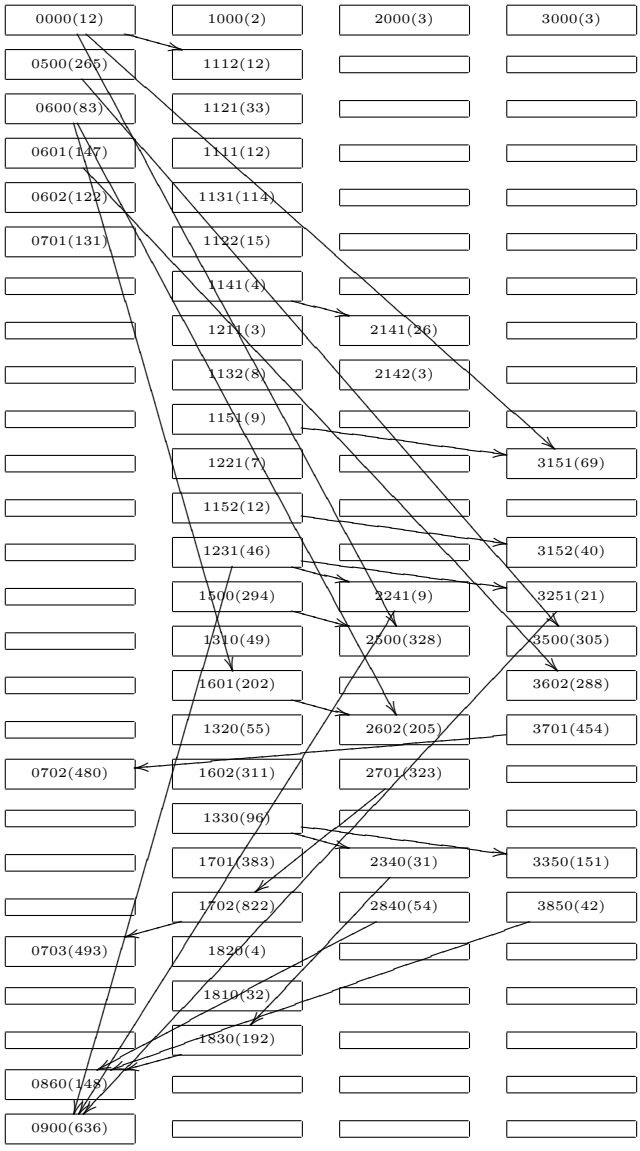


FIG. 4.27 – Modèle dynamique direct du bogie BA2000 : Schéma d'exécution parallèle en quatre tâches, après regroupement de type 1

	0	1	2	3
0	(25)	2	2	3
1	3	(71)	5	4
2	2	2	(19)	0
3	3	0	0	(19)

TAB. 4.8 – Nombres de transferts entre tâches BA2000/4P avec regroupement des transferts

Les caractéristiques des quatre tâches restantes sont données par la table 4.9. On constate que les tailles des tâches sont plus semblables que lors du regroupement de type 1. Les nombres de transferts ne sont ici que légèrement plus nombreux : deux transferts pour chacune des tâches 1 et 2.

numéro de tâche	0	1	2	3
nombre d'opérations	2517	2599	1100	1373
transferts sortants	7	14	7	3
transferts entrants	8	7	9	7

TAB. 4.9 – Caractéristiques des tâches BA2000/4P avec regroupement des transferts

La table 4.10 reprend quant à elle les nombres de transferts entre les quatre tâches.

	0	1	2	3
0	(25)	2	2	3
1	3	(59)	7	4
2	2	5	(26)	0
3	3	0	0	(19)

TAB. 4.10 – Nombres de transferts entre tâches BA2000/4P après regroupement des transferts

La longueur critique du schéma d'exécution parallèle sur quatre tâches a gardé la même valeur 3684, qui est inférieure à celle obtenue dans ce cas-ci par le regroupement de type 1.

Le nombre critique de transferts a lui aussi gardé la même valeur de 8 : on sait que le regroupement de type 2 ne vise pas explicitement à réduire le nombre des transferts.

Le schéma d'exécution en quatre tâches est illustré sur la figure 4.28. On constate que le nombre de lignes de cette représentation de la grille d'exécution est inférieur au nombre de lignes de la grille représentée sur la figure 4.27. Toutefois, ceci n'est qu'une conséquence de la présence de séquences ayant les mêmes codes d'identification partiels dans les deux prototypes de tâches regroupés. Seuls les nombres d'opérations de chaque séquence, la longueur critique ainsi que le nombre de transferts sont pertinents pour l'appréciation de la longueur d'un schéma d'exécution, et pas le nombre d'étapes ou de lignes de la grille.

Remarques En pratique il arrivera souvent que le nombre de tâches soit dicté par l'utilisateur en fonction du nombre de processeur disponibles sur son ordinateur. Dans ce cas, si le nombre de tâches est supérieur au nombre de processeurs disponibles, on peut poursuivre les regroupements selon l'un des deux critères jusqu'à ce que le nombre de tâches soit égal au nombre de processeurs.

Toutefois, on peut penser que la méthode de regroupement a priori n'est peut être pas la plus adaptée à la situation où le nombre de tâches est imposé. Nous avons envisagé une seconde méthode de création de tâche qui ne se base plus sur un principe de regroupement *a priori* des équations. Cette méthode est présentée ci-après.

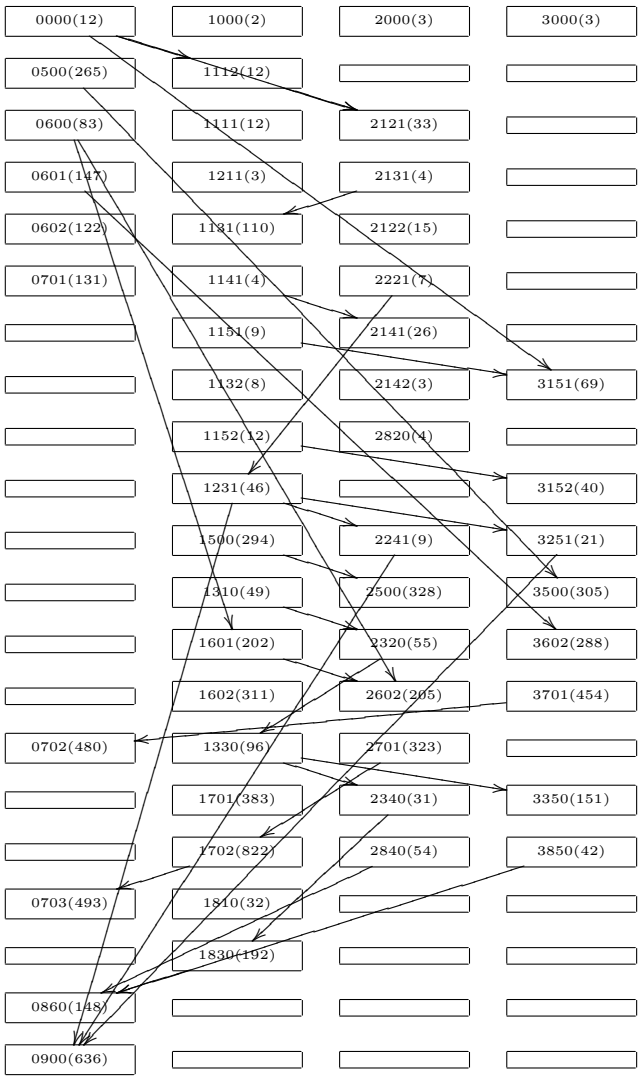


FIG. 4.28 – Modèle dynamique direct du bogie BA2000 : Schéma d'exécution parallèle en quatre tâches, après regroupement de type 2

4.3.7 Création de tâches par ordonnancement direct

Une méthode alternative de création de tâches par ordonnancement direct a été testée afin de comparer les résultats obtenus lorsque le nombre de tâches est imposé. En effet, on ne désire pas toujours exécuter le plus grand nombre de tâches possible par exemple lorsque le nombre de processeurs est inférieur au nombre de tâches parallèles détectées a priori.

Par ordonnancement direct nous exprimons le fait que nous n'effectuons pas les différents regroupements de branches et de blocs proposés ci-dessus. Les équations sont marquées et regroupées dans une structure hiérarchique selon leur code d'identification, comme précédemment, mais la valeur du champ indiquant la tâche a priori est ignorée par la méthode d'ordonnancement qui les distribue directement entre n processeurs.

Les séquences d'équations, les noeuds de niveau trois dans la structure hiérarchique, sont ordonnancées comme s'il s'agissait de macro opérations. La procédure est similaire à celle employée pour la vectorisation. La liste des séquences est triée par ordre croissant d'indice `alap` et `asap`, et celles-ci sont distribuées selon cet ordre dans une grille d'exécution. Le tri de la liste assure que les parents d'une séquence sont toujours ordonnancés avant elle. Chaque ligne de la grille correspond à une étape d'exécution et chaque colonne correspond à une tâche ou un processeur.

Pour chaque séquence, on détermine que l'étape à laquelle elle peut être exécutée au plus tôt est l'étape suivant directement l'étape la plus tardive à laquelle un de ses parents a été programmé. On détermine également une tâche préférentielle dans laquelle se situent le plus grand nombre de ses parents afin d'essayer d'y placer cette séquence et ainsi limiter le nombre de transfert entre tâches. Si la case de la grille d'exécution correspondant à cette étape et à cette tâche est vide, on y place la séquence à exécuter. Si cette case n'est pas libre on en cherche une autre sur la ligne correspondant à l'étape déterminée. Si toutes les cases de la grille sont réservées, on répète la recherche à la ligne suivante, et ainsi de suite jusqu'à trouver une case disponible.

Lorsque toutes les séquences sont distribuées dans la grille, on procède à l'évaluation de la longueur critique du schéma obtenu ainsi qu'au nombre critique de transferts, et on établit une matrice de communication entre tâche.

Résultats Les résultats obtenus avec cette méthode d'ordonnancement libre montrent qu'elle peut fournir des schémas d'exécution ayant une longueur critique plus faible, pour un même nombre de tâches, que celle des schémas obtenus par la méthode d'affectation a priori et les regroupements de tâches. Toutefois, le nombre de transferts entre tâches est systématiquement inférieur en utilisant la méthode a priori et des regroupements de type 1 ou 2.

Nous proposons d'observer les résultats obtenus pour le système BA2000.

Méthode a priori La méthode de regroupement a priori permet d'obtenir cinq prototypes de tâches dont les caractéristiques sont reprises dans la table 4.5.

Les résultats des regroupements de type 1 ou 2 ont été présentés plus haut dans les tables 4.7 et 4.9.

Les regroupements successifs permettent de réduire le nombre de tâches à trois. Leurs caractéristiques sont reprises sur la table 4.11. Le nombre critique d'operations est 4601 et le nombre critique de transferts est de 6.

numéro de tâche	0	1	2
prototypes de tâches	0	1	2,3,4
nombre d'opérations	2517	2599	2473
transferts sortants	5	13	8
transferts entrants	6	7	13

TAB. 4.11 – Caractéristiques des 3 tâches BA2000 obtenues par regroupement

Il est également possibles d'effectuer d'autres regroupements si on désire ne générer que deux tâches. Leurs caractéristiques sont reprises sur la table 4.12. Dans ce cas, les regroupements des prototypes de tâches correspondent a un choix de l'utilisateur. Le nombre critique d'opérations est 4704 et le nombre critique de transferts est de 2.

numéro de tâche	0	1
prototypes de tâches	0,4	1,2,3
nombre d'opérations	3890	3699
transferts sortants	2	7
transferts entrants	7	2

TAB. 4.12 – Caractéristiques des 2 tâches BA2000 obtenues par regroupement

Méthode directe Nous proposons ici les résultats obtenus en utilisant la méthode d'ordonnancement direct sur 4, 3, et 2 processeurs. Les nombres critiques d'opérations et de transferts sont systématiquement supérieurs à ceux obtenus en utilisant la méthode a priori. C'est également le cas pour le nombre total de transferts. Les nombres d'opérations de chaque tâches n'ont pas été calculés dans le cas de l'ordonnancement direct.

Pour quatre processeurs, le nombre critique d'opérations est 4548 et le nombre critique de transferts est de 11.

Pour trois processeurs, le nombre critique d'opérations est 4666 et le nombre critique de transferts est de 11.

numéro de tâche	0	1	2	3
nombre d'opérations	-	-	-	-
transferts sortants	14	14	12	13
transferts entrants	10	12	15	16

TAB. 4.13 – Caractéristiques des 4 tâches BA2000 obtenues par ordonnancement direct

numéro de tâche	0	1	2
nombre d'opérations	-	-	-
transferts sortants	17	16	15
transferts entrants	15	18	15

TAB. 4.14 – Caractéristiques des 3 tâches BA2000 obtenues par ordonnancement direct

Pour deux processeurs, le nombre critique d'opérations est 5487 et le nombre critique de transferts est de 7.

numéro de tâche	0	1
nombre d'opérations	-	-
transferts sortants	16	12
transferts entrants	12	16

TAB. 4.15 – Caractéristiques des 2 tâches BA2000 obtenues par ordonnancement direct

4.4 Implémentation du code parallèle

4.4.1 Introduction

L'exploitation des capacités de calcul parallèle d'un ordinateur requiert l'utilisation d'un paradigme de programmation adéquat. Un programme destiné à une exécution en parallèle contient des directives de compilation, des instructions ou des appels à des fonctions dont le rôle est d'une part la distribution des tâches aux différents processeurs et d'autre part la communication ou la synchronisation entre les différentes tâches.

Différents paradigmes existent et sont utilisables dans différents langages tels que FORTRAN, C ou C++. Les deux principaux standards sont OpenMP

[119, 120] et MPI [121, 122].

Le standard OpenMP est destiné à la programmation des ordinateurs parallèles à mémoire *partagée*[8]. Il définit un ensemble de directives de compilation et une librairie de fonctions de synchronisation. Un avantage de ce standard est qu'il permet d'écrire des programmes parallèles dont la structure est semblable à celle d'un programme destiné à une exécution séquentielle sur un ordinateur mono processeur. Un programme parallèle qui utilise OpenMP peut être écrit de manière à pouvoir être compilé en ignorant les directives OpenMP et être ainsi portable sur tout système de calcul, parallèle ou non.

En pratique, le problème se pose plus généralement dans l'autre sens : il s'agit de transformer un programme séquentiel pour exploiter les capacités de calcul parallèle d'un ordinateur multi processeurs. OpenMP est alors un paradigme qui permet d'effectuer cette transformation, progressivement, sans nécessiter de récrire tout le code, si on dispose d'un système à mémoire partagée. Toutefois, la compilation d'un programme qui utilise des directives OpenMP nécessite l'utilisation d'un compilateur spécifique qui reconnaît ces directives et les traduit de façon à produire l'effet désiré.

Le standard MPI[9] est quant à lui utilisable sur tout type d'ordinateur parallèle, à mémoire *partagée* ou *distribuée*. Il est donc plus général que OpenMP. Il définit un ensemble de fonctions de communication et de synchronisation entre des tâches qui s'exécutent sur des processeurs ou des ordinateurs différents reliés en réseau. Le prix de la portabilité sur tout type d'ordinateur parallèle est l'utilisation d'un environnement d'exécution spécifique qui se charge de la distribution du programme sur les différents processeurs et permet les interactions entre les différentes tâches. L'utilisation de ce standard conduit parfois à une structure de programme qui n'est plus compatible avec une compilation classique en vue d'une exécution séquentielle. Ceci n'empêche toutefois pas l'exécution sur une machine mono processeur, mais nécessite toujours l'utilisation de l'environnement d'exécution MPI. Toutefois, un programme écrit en utilisant le paradigme MPI peut être compilé avec n'importe quel compilateur, ce qui est un argument décisif en ce qui concerne notre choix.

Les ordinateurs parallèles à mémoire partagée contiennent souvent un plus petit nombre de processeurs que les ordinateurs à mémoires distribuées tels que les ordinateurs parallèles en grappe[10]. On trouve très facilement des ordinateurs à mémoire partagée contenant deux ou quatre processeurs et pour un prix seulement légèrement supérieur à leur version mono-processeur. Toutefois, leur prix augmente plus que linéairement en fonction du nombre de processeurs ce qui les rend alors bien plus cher que les ordinateurs parallèles en grappe, composés de plusieurs noeuds mono, bi ou même quadri-processeurs. Les ordinateurs parallèles en grappe sont donc bien plus répandus lorsqu'on considère

[8] SMP : Shared Memory Programming
[9] Message Passing Interface
[10] Cluster

un nombre plus important de processeurs, ce qui semble être une évidence en matière de calcul parallèle.

En ce qui concerne la parallélisation des modèles de systèmes multicorps classiques, on remarque en pratique que le nombre de tâches générées et donc de processeurs réellement exploités est bien souvent inférieur à dix et généralement plus proche de quatre pour des systèmes de taille moyenne ou petite. Les systèmes ferroviaires tels qu'un train composé d'une dizaine de wagons et d'autant de bogies ou les systèmes moléculaires sont des cas particuliers parmi les systèmes multicorps, en ce qui concerne le nombre de tâches obtenues. Dans la majorité des cas, les modèles parallèles produits ne nécessitent pas un grand nombre de processeurs et peuvent donc être exécutés sur de petits ordinateurs à mémoire partagée tels que de petites stations de travail multiprocesseurs.

Dans cette optique, le choix du standard OpenMP semble tout aussi intéressant. Nous envisageons d'ailleurs de proposer des modèles parallèles qui utilisent ce standard, mais au moment de la rédaction de ce texte, seuls des modèles parallèles qui utilisent le paradigme MPI ont été générés et validés.

4.4.2 Implémentation avec MPI

Pour les modèles parallèles basés sur le paradigme MPI, nous avons choisi une structure d'implémentation assez simple qui n'est toutefois plus compatible avec une exécution séquentielle. Suite à la distribution des séquences d'équations dans une grille d'exécution, dont chaque colonne correspond à une tâche, nous produisons une fonction pour chaque tâche. Ces fonctions sont appelées par la fonction principale du modèle selon le processeur sur lequel se déroule l'exécution.

Une caractéristique importante de l'utilisation du paradigme MPI est que le même programme est exécuté sur chaque processeur. Afin de différencier le travail effectué par chaque processeur, il est donc nécessaire que le programme tienne compte du numéro du processeur sur lequel il est exécuté afin d'effectuer le travail destiné à ce processeur. Le numéro de processeur `process_Id` peut être obtenu en faisant appel une seule fois dans le programme principal à la fonction `MPI_Comm_rank`.

Les quelques lignes de code de la figure 4.29 correspondent à la fonction principale `accelredP` qui implémente un modèle dynamique direct. En plus des paramètres habituels, le numéro du processeur `process_Id` est passé en argument de manière à exécuter la tâche prévue sur chaque processeur. On peut noter la présence de l'appel à la fonction de communication `MPI_Bcast` qui permet de distribuer les valeurs des coordonnées articulaires q et de leurs dérivées premières $\dot{q}$ à toutes les tâches au début de chaque exécution du modèle.

En ce qui concerne les fonctions `Task_0`, `Task_1`, etc. qui contiennent le travail à effectuer par chacune des tâches, leur structure est similaire à la structure

```
int  accelredP({paramètres},int process_Id)
{
      int iter;
      ...
      MPI_Bcast(q, nbody +1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
      MPI_Bcast(qd, nbody +1, MPI_DOUBLE, 0, MPI_COMM_WORLD);

      switch(process_Id)
      {
        case 0:
            iter=Task_0({paramètres}, process_Id);
            ...
            break;
        case 1:
            iter=Task_1({paramètres}, process_Id);
            ...
            break;
        ...
        default:
            break;
      }
      return iter;
}
```

FIG. 4.29 – Fonction principale d'un modèle dynamique direct parallèle

des modèle séquentiels, à ceci près que des appels aux fonctions de communication point à point `MPI_Send` et `MPI_Recv` sont placés automatiquement par ROBOTRAN entre les séquences d'équations pour effectuer les transferts de données entre tâches.

Cette structure n'est pas imposée par MPI ; le regroupement des séquences à exécuter par chaque processeur dans des fonctions est un choix qui permet d'implémenter facilement et simplement le travail à effectuer par chaque processeur. Il serait également possible d'insérer des tests en plus des instructions de communication avant chaque séquence d'équations dans la structure séquentielle. Cette façon de procéder pourrait permettre l'élaboration de modèles parallèles pouvant être exécutés sur un nombre de processeurs qui ne devrait pas être fixé a priori lors de la génération. Toutefois, nous n'avons pas évalué les avantages ou inconvénients de cette seconde approche plus complexe.

Remarques Le code des modèles parallèles est généré de façon totalement automatique. La fonction `accelredP` qui implémente un modèle parallèle peut être directement insérée dans un programme de simulation écrit pour MPI, lequel peut être compilé et exécuté sans nécessiter d'autre intervention de l'utilisateur.

Toutefois, l'objectif poursuivi jusqu'à présent au travers de l'implémentation des modèles en utilisant MPI est essentiellement la validation de la méthode de découpage et de la génération du code parallèle. Il n'a pas encore vraiment été question de chercher à générer un code parallèle optimal. Ceci pourra faire

l'objet de développements ultérieurs visant à maximiser les performances d'exécution des modèles parallèles en utilisant éventuellement d'autres fonctions de communications ou même d'autres paradigmes de programmation parallèle.

4.5 Résultats de simulation sur ordinateur parallèle

Afin d'analyser l'intérêt pratique de nos développements en terme de gain en temps calcul pour la simulation de systèmes multicorps, nous avons procédé à quelques mesures de temps d'exécution comparées entre des modèles séquentiels et des modèles parallélisés, tous générés par ROBOTRAN.

4.5.1 Système et simulation

Nous avons choisi d'évaluer l'intérêt de la parallélisation du modèle dynamique direct d'une Audi A6 qui est un véhicule dont la structure des suspensions est assez complexe.

Ce système a déjà été présenté dans la section 2.8 et a été choisi pour sa structure arborescente présentant un parallélisme topologique important afin de se placer dans des conditions favorables en ce qui concerne la quantité de calculs à effectuer par chacune des tâches obtenues.

La parallélisation du modèle dynamique direct a été réalisée automatiquement par ROBOTRAN en utilisant la méthode de découpage *a priori*. Les tâches ont été constituée en effectuant des regroupements de type 2, c'est-à-dire en regroupant deux à deux les prototypes de tâches qui ont les plus petites tailles.

Le modèle *A6* Le modèle *A6* est destiné à la simulation d'un véhicule de type Audi A6. Ce véhicule possède des suspensions dont la structure, illustrée sur la figure 4.30, rappelle celle des suspensions à double triangle. Toutefois ici les triangles inférieurs et supérieurs sont constitués de deux barres séparées. Ainsi, chaque porteur de roue est relié au châssis par quatre barres de suspension et une barre de direction pour l'avant ou une barre de maintien d'orientation pour l'arrière.

On notera que les axes des articulations des barres de suspensions ne sont pas alignés les uns avec les autres. Le nombre de corps du modèle ROBOTRAN s'élève à 84, dont près de cinquante sont fictifs, c'est-à-dire, sans influence dynamique (masse et dimensions nulles). Ceux-ci servent à modéliser les articulations à plusieurs degrés de liberté ou à introduire des repères intermédiaires pour l'orientation des axes de certaines articulations, selon les conventions de modélisation de ROBOTRAN. Il y a donc également 84 coordonnées articulaires dans le modèle.

En pratique, pour la simulation effectuée,

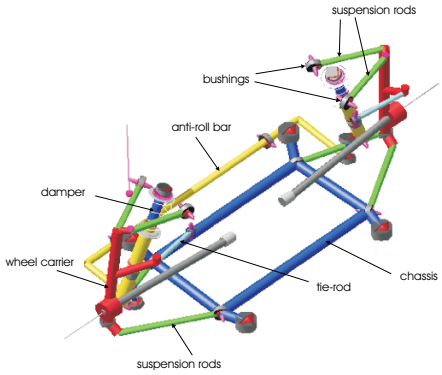


FIG. 4.30 – Structure de la suspension de l'Audi A6

- 29 de ces coordonnées ont des valeurs constantes,
- la cinématique de 6 coordonnées est imposée, à savoir le déplacement longitudinal absolu du châssis, les angles de rotation de quatre roues ainsi que l'angle du volant,
- 40 coordonnées sont dépendantes vu la structure bouclée des suspensions ; leurs valeurs résultent de la résolution de 40 équations de contraintes cinématiques provenant des coupures de ces boucles,
- 9 coordonnées sont libres : il s'agit des coordonnées qui correspondent aux cinq degrés de liberté du châssis et aux débattements des quatre suspensions. Leur dérivées premières et secondes constituent les 18 variables d'état du système.

Les dispositifs ressorts et amortisseurs sont modélisés par des lois de comportement linéaire et le modèle de contact roue/sol est un modèle purement latéral qui fournit les forces appliquées sur chaque roue en évaluant des fonctions résultant de l'approximation de mesures expérimentales.

La complexité de la structure de ce véhicule le situe parmi les gros systèmes modélisés par des corps rigides, juste en dessous des véhicules ferroviaires complexes et des trains.

Le nombre d'opérations arithmétiques en virgule flottante nécessaire à l'évaluation de la formulation réduite du modèle dynamique direct s'élève à près de 35000, plus quatre évaluations du modèle de contact des roues avec le sol.

Néanmoins, nous sommes bien loin des nombres d'opérations rencontrés dans d'autres disciplines de la mécanique telles que l'analyse de structure ou la mécanique des fluides, pour lesquelles le recours au calcul parallèle est habituel.

La manoeuvre simulée consiste en un double changement de bande, similaire à une manoeuvre de dépassement. Nous utilisons une méthode d'intégration numérique de type Runge-Kutta d'ordre 4 avec un pas de temps de 1 milli

seconde. Le temps simulé est de 20 secondes, ce qui implique 80000 évaluations du modèle.

Deux modèles parallèles sont générés, pour 2 et pour 4 processeurs. Sur base des nombres critiques d'opérations déterminés par ROBOTRAN, les facteurs de réduction de temps de calcul attendus sont respectivement de 1.9 et de 3.5, en négligeant les délais de communications.

4.5.2 Ordinateurs parallèles

Les mesures de temps de simulation que nous présentons ont été prises sur différents ordinateurs parallèles disponibles en 2004 au centre de Calcul Intensif de L'UCL, à Louvain-la-Neuve. Tous ces ordinateurs ont une architecture symétrique multiprocesseurs à mémoire partagée ce qui est l'architecture idéale pour limiter le coût de communication entre les tâches.

- *PICCARD* est un serveur HP S-Class muni de 16 processeurs PA-Risc8000 cadencés à 180MHz. La bande passante de communication entre les processeurs et la mémoire centrale est de 1.9 GB/s. C'est une machine qui date d'une dizaine d'années.
- *CHPIT* est un serveur HP Integrity RX4640 muni de 4 processeurs Itanium2 cadencés à 1.5 GHz. La bande passante du bus d'accès à la mémoire centrale est de 12.8 GB/s. Cette machine a moins d'un an.

4.5.3 Mesures de temps de simulation

Pour chaque test effectué, nous consignons dans une table nos observations qui se déclinent en plusieurs considérations :

nombre de tâches : nombre de processus qui sont créés et nombre de processeurs utilisés, pour le calcul de la simulation,

accélération prévue : le rapport entre le nombre total d'opérations du modèle et le nombre critique d'opérations du schéma de calcul parallèle,

temps calcul : mesuré "montre en main", c'est-à-dire, le nombre de secondes nécessaires au calcul de la simulation,

accélération réelle : le rapport entre le *temps calcul* séquentiel sur un seul processeur et le *temps calcul* en parallèle sur n processeurs,

surcoût : différence entre *l'accélération prévue* et *l'accélération réelle* rapportée à *l'accélération réelle*,

efficacité parallèle : le rapport entre *l'accélération réelle* et le nombre de processeurs. Il s'agit d'un critère d'évaluation assez sévère qui correspond au rendement global du calcul d'un modèle parallélisé.

A6 sur PICCARD La table 4.16 reprend les observations relatives aux simulations du modèle A6 en utilisant 1, 2 ou 4 processeurs de l'ordinateur Piccard.

nombre de tâches	1	2	4
accélération prévue	1	1.9	3.5
temps calcul [sec]	98	60	46
réduction relative	0	38%	53%
accélération réelle	1	1.6	2.2
surcoût	0	19%	59%
efficacité parallèle	1	80%	55%

TAB. 4.16 – Simulation du modèle A6 sur PICCARD

Bien que les résultats obtenus sur deux processeurs sont assez bons, on constate que malgré une efficacité parallèle attendue de près de 90%, la réduction de temps calcul obtenue sur 4 processeurs n'est pas aussi importante. Le surcoût des communications par rapport à une situation idéale est assez élevé, 59% de telle sorte que le rendement global du calcul du modèle en parallèle est assez faible : 55%.

nombre de tâches	1	2	4
accélération prévue	1	1.9	3.5
temps calcul [sec]	16.9	9.7	5.9
réduction relative	0	43%	65%
accélération réelle	1	1.7	2.9
surcoût	0	12%	20%
efficacité parallèle	1	85%	72%

TAB. 4.17 – Simulation du modèle A6 sur CHPIT

A6 sur CHPIT On remarque que les résultats sont assez bons. On obtient une meilleure efficacité parallèle sur cette machine récente que sur une machine d'une dizaine d'année. Cela s'explique par une plus grande capacité de transfert de données entre les processeurs et la mémoire centrale, ce qui réduit les temps de communication entre les tâches. La dégradation est également moins marquée entre l'utilisation de 2 et de 4 processeurs. On obtient cette fois une efficacité parallèle de 72% sur quatre processeurs.

Remarque Dans [64], le découpage manuel d'un modèle de Jeep Iltis avait permis d'obtenir des accélérations réelles de 1.9 et 3.4 en utilisant respectivement 2 et 4 processeurs d'un ordinateur équivalent à Piccard. Bien que la voiture Audi A6 soit plus complexe que la Jeep Iltis, nous n'obtenons pas d'aussi bons

résultats. C'est essentiellement du à l'importante réduction de la taille des modèles que nous générons actuellement de manière complètement symbolique. En effet, notre approche réduit fortement le nombre total d'opérations à effectuer par rapport à l'approche mixte symbolique/numérique utilisée dans [64], ce qui joue en défaveur du coût relatif des communications par rapport au poids des tâches, et par conséquent de l'efficacité parallèle.

En revanche, les résultats que nous obtenons ici sont nettement meilleurs que ceux présentés dans [11] où une méthode de modélisation adaptée au calcul parallèle avait été utilisée pour simuler une Jeep Iltis. Nous obtenons des réductions de temps de calcul deux fois plus importantes.

4.6 Conclusions

Après avoir analysé le parallélisme présent dans les modèles dynamiques, nous avons élaboré une méthode de segmentation des modèles. Cette méthode repose sur une analyse de la topologie du système et de la structure algébrique de la matrice Jacobienne des contraintes, et sur le marquage des équations par un code d'identification. Après leur génération, les équations sont alors regroupées en fonction de leur code d'identification pour former des prototypes de tâches. Ces prototypes de tâches sont regroupés à leur tour afin d'obtenir un nombre de tâches équilibrées qui correspond au nombre maximum fixé par l'utilisateur.

Cette méthode de regroupement a priori a été comparée avec une méthode d'ordonnancement direct, similaire à celle utilisée pour la vectorisation. Les modèles parallèles obtenus en utilisant la méthode de regroupement a priori nécessitent moins de communications entre tâches pour un taux de parallélisation équivalent, ce qui démontre l'intérêt d'effectuer la procédure de regroupement a priori pour la constitution des tâches plutôt que d'utiliser une procédure d'ordonnancement direct.

Malgré des taux de parallélisation voisins de 90% et un petit nombre de communications, l'efficacité du calcul de nos modèles dynamiques directs réduits sur des ordinateurs parallèles classiques n'est pas excellente. Ce paradoxe s'explique par leur génération symbolique qui a pour conséquence de réduire très fortement le nombre d'opérations à effectuer lors de leur évaluation. Malgré une forte minimisation du nombre de communications entre tâches, celles-ci ont des tailles relativement petites, ce qui a un effet défavorable sur le rapport du temps calcul au temps de communication, lequel a un effet direct sur le rendement du calcul en parallèle. Il est certainement plus facile d'obtenir de meilleurs résultats en terme d'efficacité parallèle en partant de modèles moins efficaces au départ, comme c'est le cas dans [64].

Il s'avère donc relativement difficile de réduire encore efficacement, par le recours au calcul parallèle classique, le temps calcul des modèles symboliques de

systèmes classiques composés de seulement quelques dizaines de corps rigides.

Cette tentative de diminuer encore les temps de simulations est toutefois loin d'être un échec dans la mesure où :

- les temps de calcul ont tout de même pu être réduit de façon significative : -65% sur 4 processeurs,
- la méthode proposée apporte une solution complémentaire intéressante au problème de la distribution des modèles complexes sur des architectures parallèles à grains fins,
- la procédure de découpage par l'approche symbolique permet d'obtenir des modèles parallèles de façon automatique pour une large gamme de systèmes multicorps.

Finalement, des modèles de systèmes plus complexes, composés de corps flexibles ou encore de taille bien plus grande tels que des trains entiers, auraient probablement une taille suffisante pour justifier le recours au calcul sur ordinateurs parallèles afin de pouvoir les simuler en temps réel avec une meilleure efficacité.

Conclusion

Dans ce travail, nous avons présenté diverses techniques que nous avons implémentées dans le logiciel de génération symbolique ROBOTRAN afin d'obtenir des modèles portables et performants pour l'analyse des systèmes multicorps.

1. Synthèse

Systèmes arborescents

Nous avons montré dans le chapitre 1 que, grâce à l'ajout de quelques fonctionnalités aux librairies symboliques de ROBOTRAN, nous avons pu produire *des modèles symboliques complets* pour systèmes arborescents.

D'une part, l'ajout des fonctions à plusieurs arguments dans la liste des expressions symboliques reconnues par ROBOTRAN permet de générer des modèles symboliques dans lesquels le chaînage des équations récursives n'est pas interrompu par l'utilisation d'appel à des fonctions externes au sein du modèle. Celles-ci sont quasiment indispensables lorsque le système est en interaction avec son environnement ou lorsque qu'il est soumis à des efforts provenant de la présence d'éléments actifs ou passifs dans sa structure. Elle permettent d'utiliser des modèles externes qui implémentent le calcul des forces d'interaction.

Ces fonctions externes sont également un moyen simple et efficace pour *connecter correctement* le modèle mécanique du système multicorps avec des modèles électriques, hydrauliques, etc. des éléments actifs ou passifs présent dans sa structure.

D'autre part, la génération symbolique de certains ingrédients cinématiques permet de simplifier l'implémentation de ces fonctions externes. En effet, celles-ci reçoivent des arguments spécifiques de telle sorte qu'elles ne doivent plus contenir que les équations de calcul des efforts d'interaction. Nous avons illustré ces interfaces pour le calcul de forces extérieures en présentant un modèle de moto.

Systèmes bouclés

Dans le chapitre 2, nous avons montré comment nous produisons des *modèles d'état symboliques* pour l'analyse dynamique des systèmes dont la structure contient des boucles cinématiques et qui sont éventuellement soumis à des contraintes externes.

Une *technique de coupure* est utilisée pour générer les équations de contraintes qui expriment des conditions de fermeture des boucles. La technique du *partitionnement des coordonnées* est utilisée pour convertir l'ensemble d'équations différentielles algébriques en un *système réduit d'équations purement différentielles.*

D'une part nous proposons de résoudre les équations de contraintes à tous les niveaux, à savoir en position, en vitesse et en accélération, ce qui garantit une excellente précision des résultats de simulation en ce qui concerne le respect des conditions de fermeture des boucles et une bonne stabilité du processus d'intégration numérique.

La méthode itérative de Newton-Raphson est choisie pour résoudre les équations de contraintes non linéaires. Bien qu'il s'agisse d'une méthode numérique, nous générons *symboliquement* toutes les expressions nécessaires à son exécution. Celles-ci font partie intégrante des équations récursives du modèle, bien qu'elles puissent être évaluées plusieurs fois de façon itérative, lors de l'exécution du modèle.

En procédant à une factorisation bloc triangulaire de la matrice Jacobienne des contraintes, nous pouvons résoudre les contraintes par petits groupes et ainsi réaliser une économie de calculs par rapport à une application numérique classique de la méthode de Newton-Raphson. Nous nous approchons donc en cela des caractéristiques d'une méthode de résolution analytique des contraintes telle que celle présentée dans [45], sans toutefois être limité au cas des contraintes provenant uniquement des boucles cinématiques.

La résolution symbolique des dérivées premières et secondes des contraintes, respectivement linéaires en les vitesses et les accélérations généralisées, correspond à une résolution analytique et se base sur une factorisation LU de la Jacobienne et une procédure d'élimination de Gauss. Ces opérations de résolution et de factorisation sont effectuées de manière symbolique. ROBOTRAN a la capacité de traiter symboliquement plus d'une centaine d'équations de contraintes.

D'autre part nous proposons de réduire l'ensemble des équations différentielles algébriques de manière à obtenir un système d'équations purement différentielles, du système. L'approche symbolique constitue à nouveau ici un outil précieux pour tirer profit de la structure creuse des matrices impliquées dans la procédure de réduction, ce qui permet de diminuer fortement le nombre d'opérations par rapport à une implémentation numérique de cette procédure.

Finalement, les équations réduites sont résolues par factorisation LDL^T de

la matrice de masse réduite et application d'une procédure d'élimination de Gauss, ce qui permet d'obtenir les accélérations généralisées, et ainsi le modèle d'état du système. ROBOTRAN a la capacité de traiter symboliquement des modèles de systèmes qui ont plus d'une centaine de degrés de liberté.

Vectorisation

La génération des équations du mouvement sous forme symbolique permet de mettre en évidence un taux de parallélisme très important au niveau des opérations arithmétiques, même pour des systèmes multicorps à topologie linéaire. Nous présentons dans le chapitre 3 une méthode qui permet d'extraire ce parallélisme. Les équations récursives sont décomposées et après analyse des inter dépendances des opérations, celles-ci sont triées au sein d'une liste. Elles sont ensuite ordonnancées pour être exécutées par un processeur possédant une architecture appropriée.

L'analyse des équations récursives générées par l'approche symbolique contribue à déterminer les caractéristiques requises au niveau de l'architecture du processeur pour tirer un profit maximum du parallélisme à grain fin.

Des tests ont été effectués sur un prototype d'architecture synchronisée par les données développée dans [59] et implémentée sur une carte multi FPGA [57]. Un modèle dynamique inverse de robot PUMA a pu être distribué sur quatre cellules de calcul contenant chacune une unité de multiplication et une unité d'addition-soustraction en virgule flottante. Une réduction du temps de calcul d'un facteur 3, par rapport à l'exécution sur une seule cellule, a été obtenu.

En utilisant des circuits de prototypage d'une capacité suffisante en terme de portes logiques, et en appliquant cette méthode à des modèles de systèmes de grande taille, il devrait être possible d'obtenir une réduction du temps de calcul de plus d'un ordre de grandeur pour l'exécution de modèles générés symboliquement sur base de formalismes récursifs qui ne sont pourtant pas intrinsèquement parallèles et ceci quelle que soit la topologie du système.

Parallélisation

Dans le chapitre 4, nous proposons une méthode de parallélisation qui permet de découper automatiquement les modèles générés symboliquement en vue de leur exécution sur un ordinateur parallèle classique. Le découpage des modèles repose sur une analyse de la structure de la matrice Jacobienne des contraintes et sur l'analyse de la topologie du système. Ici, c'est essentiellement le parallélisme topologique du système qui est exploité.

La séparation et la répartition des équations entre différentes tâches relativement indépendantes se base sur une technique de marquage des équations. Il n'est dès lors pas nécessaire de recourir à une implémentation parallèle spécifique du formalisme utilisé pour générer les équations.

Une procédure de réduction du nombre de communications entre les différentes tâches est appliquée afin d'augmenter l'efficacité de l'exécution des modèles en parallèle.

Des tests ont été effectués sur des ordinateurs multi processeurs à mémoire partagée en utilisant le paradigme de programmation MPI. Le modèle dynamique direct réduit d'une voiture Audi A6 a été découpé de façon automatique et distribué sur quatre processeurs. Nous avons obtenu une réduction du temps de calcul d'un facteur proche de 3, par rapport à l'exécution du modèle séquentiel sur un seul processeur.

Notre méthode se révèle donc relativement efficace dans la mesure où elle fournit automatiquement des versions parallèles de modèles générés symboliquement en utilisant des formalismes qui ne sont pas intrinsèquement parallèles.

2. Observations

Toutes les techniques élaborées dans ce travail reposent sur un pilier commun qui est l'approche symbolique. Nous avons également choisi d'utiliser le formalisme récursif Newton-Euler et la technique du partitionnement de coordonnées pour générer les modèles dynamiques des systèmes multicorps.

Ces choix nous ont permis d'atteindre les objectifs que nous nous étions fixés en terme de portabilité, de performance et de fiabilité des modèles. Toutefois nous ne pouvons conclure ce travail sans porter un regard critique sur ces choix, a posteriori.

Avantages

L'approche symbolique est manifestement un excellent choix pour produire des modèles portables et efficaces.

Les routines de manipulation et de simplification symbolique implémentées dans le logiciel ROBOTRAN permettent de réduire de façon considérable le nombre d'opérations à effectuer lors de l'évaluation des modèles. Pour le traitement des forces extérieures appliquées au système, pour le traitement des équations de contraintes et surtout pour la réduction des équations du mouvement, l'approche symbolique telle que nous l'implémentons ici offre des possibilités incomparables à celles de l'approche purement numérique.

La combinaison de l'approche symbolique et de la technique du partitionnement des coordonnées a permis de compenser le coût calcul important de cette dernière. Nous obtenons dès lors des modèles réalistes et fiables qui présentent des performances en terme de vitesse d'exécution largement compatibles avec des contraintes temps réel, alors que généralement, les ingénieurs doivent utiliser des modèles simplifiés, moins fidèles, pour satisfaire ces contraintes d'exécution en temps réel[109].

En nous intéressant à la parallélisation des équations du mouvement, nous avons également réalisé que l'approche symbolique offrait des possibilités uniques.

Comment réaliser la *vectorisation* des équations si on n'en dispose pas sous forme symbolique ? La connaissance du graphe de dépendances des opérations est la condition nécessaire à l'application de la méthode. Et seule cette méthode permet d'extraire un taux de parallélisme aussi important dans le calcul des modèles dynamiques de systèmes multicorps.

De même pour le découpage des modèles, la procédure d'analyse de la structure de la Jacobienne, la procédure de marquage et la répartition des équations en tâches indépendantes reposent sur l'existence des équations sous forme symbolique. Il est évidemment impossible de distribuer un modèle généré par l'approche numérique sans implémenter l'algorithme de génération de manière spécifique pour le calculer en parallèle. Or nous pouvons produire des versions parallèles de modèles dynamiques de manière automatique grâce à la méthode de génération symbolique.

Limitations

Il y a tout de même quelques limitations aux choix que nous avons faits, quelques conditions nécessaires pour que la combinaison se révèle gagnante.

Tout d'abord, comme nous l'avons déjà montré dans le chapitre 2, la quantité d'opérations présentes dans les modèles dynamiques peut être très variable en fonction de la structure de la matrice Jacobienne. Or cette structure dépend à la fois du partitionnement des coordonnées et de la manière dont les boucles cinématiques sont ouvertes par la technique des coupures.

Un mauvais partitionnement peut provoquer une augmentation de plus de 200% de la quantité de calculs dans certains cas. Naturellement, en renversant la situation, nous pourrions affirmer fièrement que notre approche permet en fait de réduire la taille des modèles à moins de 50%.

Mais pour cela, il faudrait disposer d'une méthode permettant d'effectuer automatiquement *le bon partitionnement* des coordonnées. Or cela nous semble difficile à réaliser de manière purement symbolique. Impossible ? Probablement, car on ne peut pas négliger les critères de conditionnement numérique de la matrice Jacobienne sous-jacents au partitionnement correct de celle-ci. Il faudrait donc combiner les deux approches symbolique et numérique pour faire cela.

De façon similaire, le choix et le placement des coupures conditionnent évidemment les possibilités de découplage au niveau de la résolution des contraintes. Or ces possibilités de découplage conditionnent non seulement le nombre d'opérations du modèle mais aussi la parallélisation des modèles.

Ici non plus nous n'avons pas proposé de méthode automatique pour la génération des équations de contraintes de fermeture des boucles. Celles-ci dépendent des choix faits par lors de la description de la topologie du système.

Cette limitation repose toutefois en partie sur les conventions de description des systèmes qui sont d'application pour l'usage de ROBOTRAN.

En ce qui concerne le choix du formalisme récursif de Newton-Euler, il est bien connu que sa compétitivité n'est assurée pour la modélisation des systèmes complexes contenant un grand nombre de corps ou d'articulations que sous la condition que les chaînes cinématiques soient courtes. C'était toujours le cas pour les systèmes que nous avons traités, mais lorsque la structure du système présente une majorité de chaînes de plus d'une dizaine voire une quinzaine d'articulations successives, les formalismes $O(N)$ deviennent plus avantageux.

Nous n'avons néanmoins pas implémenté un tel formalisme récursif. Toutefois, l'utilisation d'un formalisme $O(N)$ tel que celui décrit dans [28] serait tout à fait compatible avec les autres choix effectués dans ce travail, et offrirait les mêmes possibilités en termes de simplifications symboliques et même de parallélisation des modèles.

Enfin, la technique de vectorisation des équations, comme toutes les méthodes de parallélisation à grain fin, nécessite impérativement une architecture dédicacée qui présente une grande capacité de communication entre les nombreuses unités de calcul. La poursuite du développement et l'exploitation de cette technique pour des applications réelles, un peu plus complexes que celle envisagée dans ce travail, nécessiterait un système de prototypage un peu plus convivial et surtout de grande capacité. Ce type de matériel commence néanmoins à apparaître sur le marché.

Finalement, la parallélisation des modèles est évidemment limitée par le parallélisme topologique de la structure des systèmes. En effet, l'utilisation de formalismes récursifs intrinsèquement non parallèles, ne permet pas de distribuer les modèles sur un nombre de processeurs croissant en fonction de la complexité des systèmes.

3. Perspectives

Dans ce travail de recherche, nous avons effectué des choix afin d'atteindre une série d'objectifs que nous nous étions fixés au départ. Nous sommes satisfaits d'avoir atteint ces objectifs, mais notre travail d'investigation nous en a révélés d'autres.

Parmi ces nouvelles perspectives de recherche, certaines sont liées aux limitations rencontrées.

- D'une part au niveau du traitement des boucles cinématiques, l'automatisation de la procédure de coupure pourrait apporter une simplification de l'utilisation du logiciel. Cela impliquerait par exemple d'implémenter les techniques qui sont présentées dans [46] et [45] pour le traitement des boucles cinématiques et la résolution analytique des équations de contraintes de fermeture de ces boucles. L'ensemble des coupures disponibles

devrait également être étendu, au minimum, au cas des articulations prismatiques et rotoïdes.

- D'autre part, l'élaboration d'une procédure de partitionnement des coordonnées automatique et fiable alliant les critères numériques et symboliques constituerait un apport important étant donné l'influence du partitionnement sur la taille des modèles et les possibilités de parallélisation.

Le transfert de l'expérience acquise au niveau des choix de représentation des systèmes complexes et du partitionnement des coordonnées dans le logiciel ROBOTRAN permettrait de fournir une garantie d'obtenir systématiquement le modèle le plus performant.

D'autres perspectives, correspondent à l'élargissement de la gamme d'applications visées.

- Il nous semblerait intéressant d'implémenter d'autres formalismes tels que les formalismes $O(N)$ récursifs présenté dans [28] ou ainsi que ceux qui sont particulièrement adaptés au calcul parallèle présentés dans [34] et [31], afin de vérifier l'intérêt de nos choix et de l'approche symbolique en particulier, dans d'autres contextes.
- La plupart des systèmes réels contiennent des corps flexibles. Nos développements ne concernaient toutefois que les systèmes constitués de corps rigides. Or des formalismes récursifs sont implantés dans ROBOTRAN pour la génération symbolique de modèles de systèmes dont la structure contient des poutres flexibles [102]. Ils serait donc intéressant de combiner ces formalismes avec notre approche pour la modélisation des systèmes contenant des boucles cinématiques et des éléments flexibles. Le nombre d'équations nécessaires à la modélisation des corps flexibles est nettement plus élevé que pour celle des corps rigides. Cette augmentation de la taille des modèles devrait avoir un effet favorable sur l'efficacité de leur calcul en parallèle.
- Le logiciel ROBOTRAN ne supporte actuellement que des articulations élémentaires de type rotoïde ou prismatique, qui peuvent être combinées pour modéliser des articulations à plusieurs degrés de liberté. Or il est bien connu que la description de l'orientation de corps possédant trois degrés de liberté de rotation par composition de rotations élémentaires successives peut conduire à des problèmes de singularité mathématique. La solution consiste à utiliser une articulation sphérique basée sur les quaternions, un ensemble de quatre paramètres soumis à une contrainte, pour représenter de façon univoque l'orientation d'un corps libre de l'espace. Notre approche a permis de montrer qu'il est possible de gérer de manière symbolique les modèles d'un système dont les coordonnées sont soumises à des contraintes.

Finalement, au delà des ces perspectives scientifiques, une des plus moti-

vantes est probablement celle d'établir un réseau de contact avec des partenaires industriels en vue de valoriser l'expérience acquise dans le domaine de la modélisation et de l'analyse des systèmes multicorps au cours de ce travail de recherche et de développement du logiciel ROBOTRAN.

Bibliographie

[1] W. Schiehlen. Multibody System Dynamics : Roots and Perspectives. *Multibody System Dynamics*, 1(2) :149–188, 1997.

[2] J. Wittenburg. *Dynamics of Systems of Rigid Bodies*. Teubner, Stuttgart, 1977.

[3] M. Fayet and F. Pfister. Analysis of multibody systems with indirect coordinates and global inertia tensors. *European Journal of Mechanics And Solids*, 13(3) :431–457, 1994.

[4] Scott M. Redmond. Reduction and simplification of kinematic and dynamic equations of motion for multibody systems via indirect coordinates. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada, 2003.

[5] J. J. McPhee. A unified formulation of multibody kinematic equations in terms of absolute, joint, and indirect coordinates. *Proceedings of DETC'01 ASME Design Engineering Technical Conferences*, page 8, 2001.

[6] P. Shi and J. McPhee. Dynamics of flexible multibody systems using virtual work and linear graph theory. *Multibody System Dynamics*, 4 :355–381, 2000.

[7] E.-J. Haug. *Computer Aided Kinematics and Dynamics of Mechanical Systems*. Allyn and Bacon, Boston, 1989.

[8] P.-E. Nikravesh. *Computer Aided Analysis of Mechanical Systems*. Prentice-Hall, London, 1988.

[9] Javier Garciá de Jalón and Eduardo Bayo. *Kinematic and Dynamic Simulation of Multibody Systems. The real-time Challenge*. Mechanical Engineering Series. Springer-Verlag, New-York, 1994.

[10] J. Cuadrado, J. Cardenal, and E. Bayo. Modeling and solution methods for efficient real-time simulation of multibody dynamics. *Multibody System Dynamics*, 1(3) :259–280, 1997.

[11] J. Cuadrado, J. Cardenal, P. Morer, and E. Bayo. Intelligent Simulation of Multibody Dynamics : Space-State and Descriptor Methods in Sequen-

tial and Parallel Computing Environments. *Multibody System Dynamics*, 4 :55–73, 2000.

[12] P. Fisette. *Génération Symbolique des Equations du Mouvement de Systèmes Multicorps et Application dans le Domaine Ferroviaire*. PhD thesis, Université catholique de Louvain, Louvain-la-Neuve, Belgium, 1994.

[13] Jean-Claude Samin and Paul Fisette. *Symbolic Modeling of Multibody Systems*. Kluwer Academic Publishers, 2003. 1st edition.

[14] R. Featherstone and D. Orin. Robot dynamics : Equations and algorithms. In *Proc. IEEE Int. Conf. Robotics & Automation*, pages 826–834, San Francisco, CA, 2000.

[15] J.-Y.-S. Luh, N.-W. Walker, and R.-P.-C. Paul. On-line computational scheme for mechanical manipulators. *Journal of Dynamics Systems, Measurements and Control*, 102 :69–76, 1980.

[16] M.W. Walker and D.E. Orin. Efficient dynamic computer simulation of robotic mechanisms. *Trans. ASME, Journal of Dynamic Systems, Measurement and Control*, 104 :205–211, 1982.

[17] P. Fisette and J.-C. Samin. Symbolic Generation of Large Multibody System Dynamic Equations using a New Semi-Explicit Newton-Euler Recursive Scheme. *Archive of Applied Mechanics*, 66 :187–199, 1996.

[18] G. Golub and C. V. Loan. *Matrix Computations*. Johns Hopkins University Press, 1996. 3rd ed.

[19] M. Renaud. Quasi-minimal computation of the dynamic model of a robot manipulator utilizing the newton-euler formalism and the notion of augmented body. In *IEEE International Conference on Robotics and Automation*, pages 1677–1682, Raleigh, North Carolina, 1987.

[20] P. Maes, J.-C. Samin, and P.-Y. Willems. Linearity of multibody systems with respect to barycentric parameters : Dynamics and identification models obtained by symbolic generation. *Mechanics of Structures and Machines*, 17(2) :219–237, 1989.

[21] X. Chenut, P. Fisette, and J.-C. Samin. Recursive formalism with a minimal dynamic parameterization for the identification and simulation of multibody systems. application to the human body. *Multibody System Dynamics*, 8(2) :117–140, 2002.

[22] A.F. Vereshchagin. Computer simulation of the dynamics of complicated mechanisms of robot manipulators. *Engineering Cybernetics*, 6 :65–70, 1974.

[23] R. Featherstone. The calculation of robot dynamics using articulated body inertias. *The International Journal of Robotic Research*, 2 :13–30, 1983.

[24] R. Featherstone. *Robot Dynamics Algorithms.* Kluwer Academic Publishers, Dordrecht, 1987.

[25] D.-S. Bae and E.J. Haug. A Recursive Formulation for Constrained Mechanical System Dynamics, Part I : Open-Loop Systems. *Mechanics of Structures and Machines*, 15 :359–382, 1987.

[26] R. Schwertassek and W. Rulka. Aspects of efficient and reliable multibody systems simulation. In E.-J. Haug and R.-C Deyo, editors, *Real-Time Integration Methods for Mechanical System Simulation NATO ASI Series, Series F, Vol. 69*, pages 55–96, Berlin, 1989. Springer-Verlag.

[27] D.-S. Bae and E.J. Haug. A Recursive Formulation for Constrained Mechanical System Dynamics, Part II : Closed-Loop Systems. *Mechanics of Structures and Machines*, 15 :481–506, 1988.

[28] P. Fisette and J.-C. Samin. Symbolic generation of a multibody formalism of order n - extension to closed-loop systems using the coordinate partitioning method. *Int. J. of Numerical Methods in Engineering*, 39 :4091–4112, 1996.

[29] A. Fijany, T. Cagin, A. Jaramillo-Botero, and W. Goddard. A fast algorithm for massively parallel long-term simulation of complex molecular dynamics systems. *Parallel Computing*, pages 505–515, 1998.

[30] S.S. Kim and M.J. Vanderploeg. A general and efficient method for dynamic analysis of mechanical systems using velocity transformations. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, 108 :176–182, 1986.

[31] J.M. Jiménez. *Kinematic and Dynamic Formulations for Real-Time Simulation of Multibody Systems.* PhD thesis, University of Navarra, San Sebastián, Spain, 1993.

[32] G. Rodriguez. Kalman filtering, smoothing, and recursive robot arm forward and inverse dynamics. *IEEE Journal on Robotics and Automation*, RA-3(6) :624–639, 1987.

[33] A. Fijany, G. Kwan, and N. Bagherzadeh. A fast algorithm for parallel computation of multibody dynamics on MIMD parallel architectures. In *Proc. Computing in Aerospace 9 Conf*, pages 1021–1032, San Diego, CA, Oct. 1993.

[34] A. Fijany, I. Sharf, and G.M.T. D'Eleuterio. Parallel o(log n) algorithms for computation of manipulator forward dynamics. *IEEE Trans. Robotics & Automation*, 11(3) :389–400, june 1995.

[35] R. Featherstone and A. Fijany. A technique for analysing constrained rigid-body systems and its application to the constraint force algorithm. *IEEE Trans. Robotics & Automation*, 15(6) :1140–4, 1999.

[36] R. Featherstone. A divide-and-conquer articulated-body algorithm for parallel o(log(n)) calculation of rigid-body dynamics. part I : Basic algorithm. *Int. Journal Robotics Research*, 18(9) :867–875, 1999.

[37] R.-A. Wehage and E.-J. Haug. Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems. *Journal of Mechanical Design*, 134 :247–255, 1982.

[38] J. Baumgarte. Stabilization of constraints and integrals of motion. *Computer Methods in Applied Mechanics and Engineering*, 1 :1–16, 1972.

[39] L. Petzold. Differential/algebraic equations are not ode's. *SIAM J. Sci. Stat.Comput.*, 3(3) :367–384, 1982.

[40] C. Fuhrer and B. Leimkuhler. A new class of generalized inverses for the solution of discretized euler-lagrange equations. In E.-J. Haug and C.-D. Roderic, editors, *NATO ASI Series, Series F : Computer and Systems Sciences, Vol. 69*, pages 143–154, Berlin, 1989. Springer-Verlag.

[41] L. Petzold. Methods and softwares for differential/algebraic systems. In E.-J. Haug and C.-D. Roderic, editors, *NATO ASI Series, Series F : Computer and Systems Sciences, Vol. 69*, pages 127–140, Berlin, 1989. Springer-Verlag.

[42] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Springer Series in Computational Mathematics 14, Springer-Verlag, Berlin, 1991.

[43] A. Cardona and M. Géradin. Numerical integration of second-order differential-algebraic systems in flexible mechanism dynamics. In M.-F.-O. Seabra Pereira and J.-A.-C. Ambrósio, editors, *Computer-Aided Analysis of Rigid and Flexible Mechanical Systems, NATO ASI Series, Series E : Applied Sciences, Vol. 268*, pages 501–530, Dordrecht, the Netherlands, 1993. Kluwer Academic Publishers.

[44] P. Fisette and B. Vaneghem. Numerical integration of multibody system dynamic equations using the coordinate partitioning method in an implicit newmark scheme. *Computer Methods in Applied Mechanics and Engineering*, 135 :85–105, 1996.

[45] M. Hiller, A. Kecskeméthy, and C. Woernle. Computer-aided kinematics and dynamics of multibody systems,. Technical report, Universität -GH-Duisburg, Liège, 1992. COMETT Course Computer Aided Analysis of Mechanical Systems, course manuscript.

[46] Manfred Hiller, Andrés Kecskemethy, and Chrisoph Woernle. A loop-based kinematical analysis of complex mechanisms. *ASME-Paper 86-DET-184, New York*, 1986.

[47] Andrés Kecskemethy, T. Krupp, and Manfred Hiller. Symbolic processing of multiloop mechanism dynamics using closed-form kinematics solutions. *Multibody System Dynamics*, 1 :23–45, 1997.

[48] D.S. Bae and S.M. Yang. A stabilization method for kinematic and kinetic constraint equations. *NATO ASI Series, Series F : Computer and Systems Sciences*, 69(4) :209–232, 1989.

[49] E. Bayo, J. Garcia de Jalon, and M.A. Serna. A modified lagrangian formulation for the dynamic analysis of constrained mechanical systems. *Computer Methods in Applied Mechanics and Engineering*, 71 :183–195, 1988.

[50] J.-E. Haug and J. Yen. Generalized coordinate partitioning methods for numerical integration of differential/algebraic equations of dynamics. In E.-J. Haug and R.-C. Deyo, editors, *NATO ASI Series, Vol. F69*, pages 97–114, Berlin, 1990. Springer-Verlag.

[51] P.-E. Nikravesh. Some method for dynamic analysis of constrained mechanical systems : A survey. In E.-J. Haug, editor, *Computer Aided Analysis and Optimization of Mechanical System Dynamics NATO ASI Series, Series F : Computer and Systems Sciences, Vol. 9*, pages 351–367, Berlin, 1984. Springer-Verlag.

[52] A.-A. Shabana. *Dynamics of Multibody Systems.* Wiley & Sons, New-York, 1989.

[53] Pengfei Shi. *Flexible Multibody Dynamics : A New Approach Using Virtual Work and Graph Theory.* PhD thesis, University of Waterloo, Waterloo, Ontario, Canada, 1998.

[54] N. Kirćanski, T. Petrović, and M. Vucobratović. Parallel computation of symbolic robot models and control laws : Theory and application to transputer networks. *Journal of Robotic Systems*, 10(3) :345–368, 1993.

[55] A. Eichberger. Transputer-based multibody system dynamic simulation : Part 2 parallel implementation - results. *Mechanism Structures & Machines*, 1993.

[56] A. Fijany and A. K. Bejczy. Parallel computation of manipulator inverse dynamics. *Journal of Robotics Systems*, 8(5) :599–635, 1991.

[57] J.P. David and J.D. Legat. A 400Kgates, 8Mbytes SRAM multi-FPGA PCI system. In *Proc. International Workshop Logic and Architecture Synthesis (IWLAS 97)*, pages pp. 113–117, Grenoble, 1997.

[58] J.P. David, J.D. Legat, P. Fisette, and T. Postiau. Implementation of very large dataflow graphs on a reconfigurable architecture for robotic application. In *Proc. of the 15th International Parallel & Distributed Processing Symposium, IPDPS'2001*, San Francisco, 23-27 April 2001.

[59] Jean Pierre DAVID. *Architecture synchronisée par les données pour système reconfigurable.* PhD thesis, Université catholique de Louvain, Louvain-la-Neuve, Belgium, Juin 2002.

[60] D.-S. Bae, J. G. Kuhl, and E. Haug. A recursive formulation for constrained mechanical system dynamics : Part 3 parallel implementation. *Mechanism Structures & Machines*, 16(2) :249–269, 1988.

[61] F.-F. Tsai and Haug. Real-Time Multibody System Dynamic Simulation : Part 2. A Parallel Algorithm and Numerical Results. *Mechanism Structures & Machines*, 19(2) :129–162, 1991.

[62] R.-S. Hwang, D.-S. Bae, J.-G. Kuhl, and E.-J. Haug. Parallel processing for real-time dynamic system simulation. *J. Mech. Des., Trans. ASME*, 112(4) :520–528, 1990.

[63] A. Eichberger, C. Fuhrer, and R. Schwertassek. The benefits of parallel multibody simulation and its application to vehicle dynamics. In W. Schiehlen, editor, *Advanced Multibody System Dynamics : Simulation and Software Tools*, pages 107–126, Dordrecht, the Netherlands, 1993. Kluwer Academic Publishers.

[64] P. Fisette and J.-M. Péterkenne. Contribution to parallel and vector computation in multibody dynamics. *Parallel Computing*, 24 :717–728, 1998.

[65] A. Eichberger. Transputer-Based Multibody System Dynamic Simulation : Part 1. The Residual Algorithm - A Modified Inverse Dynamic Formulation. *Mechanism Structures & Machines*, 1993.

[66] Intec Gmbh. SIMPACK - The High End Simulation Software http ://www.simpack.de/websitep.html.

[67] MSC Software. Adams software home page : http ://www.adams.com/.

[68] LMS International. LMS Virtual.Lab Motion : http ://www.lmsintl.com/.

[69] Mechanical Simulation Corp. Carsim : Vehicle dynamics simulation for automobiles. http ://www.carsim.com/products/carsim/index.html.

[70] The MathWorks. Model and simulate mechanical systems http ://www.mathworks.com/products/simmechanics/index.html.

[71] Giles D. Wood and Dallas C. Kennedy. Simulating Mechanical Systems in Simulink with SimMechanics. Technical report, The MathWorks, http ://www.mathworks.com/products/simmechanics/technical-literature.html, 2003.

[72] Waterloo Maple Software : http ://www.maplesoft.com/.

[73] Chad Schmitke and John McPhee. Modelling mechatronic multibody systems using symbolic subsystem models. In *ECCOMAS, Multibody Dynamics*, IDMEC/IST, Lisbon, Portugal, July 1-4, 2003.

[74] J.-C. Piedboeuf. Symbolic manipulation of flexible manipulators. In *AAS/AIAA Astrodynamics Specialist Conference*, pages AAS 95–357, Halix, Canada, 1995.

[75] Brian Moore. Optimisation de la Génération Symbolique du Code et Estimation des Paramètres Dynamiques de Structures Mécaniques dans SYMOFROS. Master's thesis, Université de Québec, Canada, 2000.

[76] M.-W. Sayers. Autosim. In W. Kortüm and R.-S. Sharp, editors, *Multibody Computer Codes in Vehicle System Dynamics*, pages 53–56, Lisse, 1993. Swets & Zeitlinger.

[77] Th.-R. Kane and D.-A. Levinson. *Dynamics : Theory and Applications*. McGraw-Hill Inc., New-York, 1985.

[78] Symbolic Dynamics Inc. SD/FAST User's Manual. http ://www.sdfast.com/, 1994.

[79] D.E. Rosenthal. An order n formulation for robotic systems. *The Journal of Astronautical Sciences*, 38(4) :511–529, 1990.

[80] University of Stuttgart. Neweul - software package for the dynamic analysis of mechanical systems. http ://www.mechb.uni-stuttgart.de/Arbeit/neweul.html.

[81] E. Kreuzer and W. Schiehlen. Neweul - software for the generation of symbolical equations of motion. In W. Schiehlen, editor, *Multibody System Handbook*, pages 181–202, Berlin, 1990. Springer-Verlag.

[82] D. A. Levinson and T. R. Kane. AUTOLEV - A New Approach to Multibody Dynamics. In W. Schiehlen, editor, *Multibody System Handbook*, pages 81–102, Berlin, 1990. Springer-Verlag.

[83] Pr. Ch. Woernle Pr. A. Kecskeméthy, Pr. M. Hiller. Mobile - object-oriented multibody simulation software. http ://www.mechatronik.uni-duisburg.de/robotics/mobile/index.htm.

[84] Andrés Kecskeméthy. MOBILE 1.3 user's guide : http ://www.mechatronik.uni-duisburg.de/robotics/mobile/manual/manual.htm.

[85] Pr. M. Hiller et Bosch. FASIM C/C++ : http ://www.mechatronik.uni-duisburg.de/vehicle/welcome-e.html#fasim.

[86] Scott McMillan. Dynamechs (dynamics of mechanisms) : A multibody dynamic simulation library. http ://dynamechs.sourceforge.net/, 1991.

[87] Russell Smith. Open Dynamics Engine. http ://www.ode.org/.

[88] P. Maes, J.-C. Samin, and P.-Y. Willems. Robotran. In W. Schiehlen, editor, *Multibody System Handbook*, pages 246–264, Berlin, 1990. Springer-Verlag.

[89] P. Fisette and J.-C. Samin. Robotran symbolic generation of multibody system dynamic equations. In W. Schiehlen, editor, *Advanced Multibody System Dynamics : Simulation and Software Tools*, pages 373–378, Dordrecht, the Netherlands, 1993. Kluwer Academic Publishers.

[90] Centre for Research in Mechatronics UCL. ROBOTRAN : http ://www.prm.ucl.ac.be/recherche/projets/robotran/, 2001.

[91] B. Raucent and J.-C. Samin. Modeling and identification of dynamic parameters of robot manipulators. In R. Bernhart and S. Albright, editors, *Robot Calibration*, pages 197–232, London, 1993. Chapman & Hall.

[92] P. Fisette, K. Lipinski, and J.-C. Samin. Dynamic behaviour comparison between bogies : Rigid or articulated frame, wheelset or independent wheels. *Supplement to Vehicle System Dynamics*, 25 :152–174, 1995.

[93] R.-E. Valembois, P. Fisette, and J.-C. Samin. Comparison of various techniques for modelling flexible beams in multibody dynamics. *Nonlinear Dynamics*, 12 :367–397, 1997.

[94] P. Fisette and J.-C. Samin. The modelling of complex mechanical systems in the MATLAB/SIMULINK environment. In *First Benelux MATLAB Usersconference*, pages 1–11 (chapter 4), Amsterdam, 1997.

[95] B. Gorla and M. Renaud. *Modèles des Robots Manipulateurs, application à leur commande*. Cepadues-Editions, Toulouse, 1984.

[96] T. Postiau, L. Sass, P. Fisette, and J.-C. Samin. High-performance multibody models of road vehicles : Fully symbolic implementation and parallel computation. *Supplement to Vehicle System Dynamics*, 2001.

[97] H.-B. Pacejka and R.-S. Sharp. Shear force development by pneumatic tyres in steady-state conditions : A review of modelling aspects. *Vehicle System Dynamics*, 20 :121–175, 1991.

[98] E. Bakker, L. Nyborg, and H.-B. Pacejka. Tyre modelling for use in vehicle dynamics studies. Technical report, SAE Technical Paper 870421, 1-15, Society of Automotive Engineers, Inc., 1987.

[99] T.-D. Gillespie. *Fundamentals of Vehicle Dynamics*. Society of Automotive Engineers, Inc., Warrendale, 1992.

[100] T. Postiau, P. Fisette, and J.D. Legat. Fine grain parallelization of multibody system equations of motion. In *Proc. of Parallel Computing, pp 193-200*, Delft (The Netherlands), August 1999.

[101] P. Fisette, B. Raucent, and J.-C. Samin. Minimal dynamic characterization of tree-like multibody systems. *Nonlinear Dynamics*, 9 :165–184, 1996.

[102] P. Fisette, D.-A. Johnson, and J.-C. Samin. A fully symbolic generation of the equations of motion of multibody systems containing flexible beams. *Computer Methods in Applied Mechanics and Engineering*, 142 :123–152, 1997.

[103] W. Klier A. Dürrbaum and H. Hahn. Comparison of automatic and symbolic differentiation in mathematical modeling and computer simulation of rigid-body systems. *Multibody System Dynamics*, 7 :331–355, 2002.

[104] Robin S. Sharp. Stability, control and steering responses of motorcycles. *Vehicles Systems Dynamics*, 35(4-5) :291–318, 2001.

[105] J.-C. Samin and P.-Y. Willems. Multibody systems with loops : A pseudo-constraint approach. *Journal of Dynamics Systems, Measurements and Control*, 115 :208–213, 1993.

[106] Alex Pothen and Chin-Ju Fan. Computing the block triangular form of a sparse matrix. *ACM Transactions on Mathematical Software*, 16 :303–324, December 1990.

[107] Wolfgang Rulka and Eli Pankiewicz. Mbs approach to generate equations of motions for hil-simulations in vehicle dynamics. In *ECCOMAS, Multibody Dynamics*, IDMEC/IST, Lisbon, Portugal, July 1-4, 2003.

[108] D.S.Bae, C.H.Lee, and D.J. Yun. Integration methods for realtime simulation of multibody vehicle models. In *ECCOMAS, Multibody Dynamics*, IDMEC/IST, Lisbon, Portugal, July 1-4, 2003.

[109] Oliver Öttgen and Manfred Hiller. Software verification of an active automotive safety system using hardware-in-the-loop. In *ECCOMAS, Multibody Dynamics*, IDMEC/IST, Lisbon, Portugal, July 1-4, 2003.

[110] Laurent Sass. *Symbolic Modelling of Electromechanical Multibody Systems*. PhD thesis, UCL - Université catholique de Louvain, Louvain-la-Neuve, Belgique, 2004.

[111] SIMD from Wikipedia, the free encyclopedia http ://en.wikipedia.org/wiki/SIMD.

[112] Velocity Engine - Altivec. http ://developer.apple.com/hardware/ve/.

[113] Intel SSE2 Preview : http ://www.tommesani.com/sse2intro.html.

[114] Texas Instrument C6000 DSPs : Architecture http ://dspvillage.ti.com/docs/.

[115] Prof. Brian L. Evans. EE 345S Real-Time Digital Signal Processing Laboratory - C6x Architecture. http ://www.ece.utexas.edu/~bevans/courses/realtime/lectures/02_Architecture/ C6x.html.

[116] Atmel's mAgic Complex Domain VLIW DSP Soft Core Delivers 1.0 GFLOPS At 100 MHz. http ://www.atmel.com/products/IPCores/, June 23, 2003.

[117] Tips for designing DSPs/FPGAs http ://www.embedded-computing.com/pdfs/AccelChip.Sum04.pdf.

[118] AccelChip - Automated Synthesis of MATLAB Models http ://www.accelchip.com/.

[119] Openmp, simple, portable, scalable smp programming. www.openmp.org.

[120] Rohit Chandra et alii. *Parallel Programming in OpenMP*. Morgan Kauffman, Elsevier, 2001. ISBN : 1-55860-671-8, 231 pages.

[121] Message passing interface forum home page. www.mpi-forum.org.

[122] Peter Pacheco. *Parallel Programming with MPI*. Morgan Kauffman, Elsevier, 1996. ISBN : 1-55860-339-5, 500 pages.