UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE DE LA MAîTRISE EN
MATHÉMATIQUES ET INFORMATIQUE APPLIQUÉES

PAR
BRIAN MOORE

OPTIMISATION DE LA GÉNÉRATION SYMBOLIQUE DU CODE ET ESTIMATION DES PARAMÈTRES DYNAMIQUES DE STRUCTURES MÉCANIQUES DANS SYMOFROS

AOÛT 2000

Résumé

SYMOFROS est un logiciel permettant de modéliser et de simuler des structures mécaniques quelconques. La modélisation consiste à générer certaines grandeurs physiques associées à la structure. Par la suite, ces grandeurs physiques peuvent être utilisées afin de simuler le comportement de la structure mécanique dans un environnement donné.

Dans ce travail, nous présentons les modifications apportées à la version 3.2 de SYMOFROS dans le but d'améliorer l'efficacité de la génération symbolique des équations. Ces modifications ont en fait pour but de générer les équations plus rapidement en utilisant un minimum de mémoire et de générer des équations sous une forme qui demande le moins d'opérations possibles. Pour ce faire, plusieurs méthodes sont proposées afin d'optimiser et de manipuler les expressions obtenues symboliquement. Les outils présentés dans ce document sont très généraux puisqu'ils sont indépendants de la forme des équations et pourraient donc être appliquées à d'autres domaines de la modélisation.

Une telle simulation nécessite cependant une bonne connaissance de la structure. On doit notamment connaître certains paramètres comme la masse, les premier et deuxième moments d'inertie pour chacun des membres de la structure. Ces paramètres sont appelés paramètres dynamiques et ne sont habituellement pas connus à priori. Il est donc nécessaire de les estimer. Pour ce faire, nous proposons une méthode afin d'obtenir l'ensemble minimal de paramètres dynamiques pour une structure quelconque formée de membres rigides et flexibles. De plus, nous présentons une approche afin d'estimer la valeur des paramètres dynamiques pour une trajectoire donnée.

Abstract

SYMOFROS is a software for modeling and simulation of mechanical structures. The modeling part consists of generating physical quantities related to a given structure. Then, these quantities can be used to simulate the behavior of the structure in a known environment.

We present some modifications done on SYMOFROS 3.2 to improve the efficiency of the symbolic generation of the equations. The goal is to reduce the time and memory required to generate the equations and to write these equations in an efficient form to increase simulation speed. To achieve this goal, we propose several methods to handle and optimize recursive equations. These methods are general and could be used to model other processes.

The value of several characteristics like the mass and the inertia of each link of the structure have to be known. But these characteristics or parameters are often unknown and difficult to measure. To estimate these values, we propose an automatic and general method to find a set of identifiable and independant dynamical parameters for rigid and flexible structures. Then, a method to estimate the value of these parameters for a known trajectories is presented.

Remerciements

J'aimerais tout d'abord remercier mon superviseur, le professeur Pierre Sicard du département de génie électrique et informatique de l'Université du Québec à Trois-Rivières (UQTR) qui malgré ses nombreuses occupations, a toujours été disponible afin de guider ce projet. Il a fait aussi preuve d'un grand professionnalisme et m'a fait profiter d'un grand nombre de conseils afin de mener à bien ce projet.

Je tiens aussi à exprimer ma profonde reconnaissance à Jean-Claude Piedboeuf, gestionnaire du groupe de robotique du département des technologies spatiales de l'Agence Spatiale Canadienne (ASC) pour m'avoir accueilli dans son équipe dans le cadre du projet présenté dans ce mémoire. Je lui suis très reconnaissant de l'aide scientifique qu'il m'a apporté ainsi que d'avoir eu confiance en la collaboration entre mathématiciens et ingénieurs.

J'exprime également mes sincères remerciements à mes collègues du groupe de robotique de l'ASC, en particulier à Régent L'Archevêque pour son aide technique tout au long de ce projet.

J'aimerais remercier Belkacem Abdous et Hélène Desaulniers de l'UQTR pour avoir examiné ce document ainsi que le fond FCAR pour l'apport financier apporté à ce projet.

J'aimerais finalement réserver mes plus chaleureux remerciements à mes parents et amis qui m'ont supporté tout au long de ce projet.

Table des matières

Liste des figures

Liste des tableaux

Notation

Les vecteurs sont représentés par des lettres minuscules en caractères gras alors que les matrices sont représentées par des lettres majuscules en caractères gras ou des lettres de l'alphabet grec.

$\mathbf{B}$	Matrice d'amortissement $(\frac{\partial \mathbf{M}}{\partial \mathbf{u}}\ddot{\mathbf{q}} + \frac{\partial \mathbf{g}}{\partial \mathbf{u}})$
$\mathbf{g}$	Vecteur contenant les forces centrifuges, de Coriolis et de gravité
$\mathbf{K}$	Matrice de rigidité $(\mathbf{K}_1 + \mathbf{K}_2)$
$\mathbf{K}_1$	Matrice de rigidité associé à $\mathbf{M}$ $(\frac{\partial \mathbf{M}}{\partial \mathbf{q}}\ddot{\mathbf{q}})$
$\mathbf{K}_2$	Matrice de rigidité associé à $\mathbf{g}$ $(\frac{\partial \mathbf{g}}{\partial \mathbf{q}})$
$\mathbf{M}$	Matrice d'inertie
$\mathbf{q}$	Coordonnées généralisées
$\dot{\mathbf{q}}$	Vitesses généralisées
$\ddot{\mathbf{q}}$	Accélérations généralisées
$\mathbf{u}$	Vecteur contenant les entrées du système (ex : trajectoires désirées)
$\mathbf{U}$	Matrice d'entrées $(\mathbf{U}_1 + \mathbf{U}_2)$
$\mathbf{U}_1$	Matrice d'entrées associé à $\mathbf{M}$ $(\frac{\partial \mathbf{M}}{\partial \mathbf{u}}\ddot{\mathbf{q}})$
$\mathbf{U}_2$	Matrice d'entrées associé à $\mathbf{g}$ $(\frac{\partial \mathbf{g}}{\partial \mathbf{u}})$
$\mathbf{x}$	Variables d'états et d'accélérations $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$

Chapitre 1

Introduction

1.1 Modélisation et simulation

Le désir de compréhension du monde qui nous entoure est la racine de la science et une grande motivation de l'homme depuis de nombreux siècles. Afin de comprendre les phénomènes qui nous entourent, les scientifiques ont construit des modèles (souvent sous forme de lois) qui nous permettent de représenter certains phénomènes (systèmes) et de les simuler afin de les étudier, les contrôler ou les prédire. La simulation de tels systèmes demande une quantité énorme de calculs qui était inimaginable il y a à peine quelque décennies.

Avec la puissance actuelle des ordinateurs, il est maintenant possible de simuler des systèmes d'une grande complexité. Cependant, avant d'effectuer la simulation d'un tel système, nous devons le modéliser. Le processus de modélisation d'un système consiste en deux étapes principales : le choix du modèle associé à une certaine famille de systèmes ainsi que la dérivation des équations associées à un système particulier décrit par certaines caractéristiques.

La première étape, soit le choix du modèle, consiste à établir des règles et des principes qui permettront de représenter une certaine famille de systèmes. Par exemple, plusieurs formulations et plusieurs méthodes (Newton-Euler, Lagrange-Euler, méthode des puissances virtuelles) peuvent être utilisées afin de modéliser la dynamique d'un système mécanique.

Une fois le choix du modèle effectué et à l'aide de ce dernier, il est nécessaire de dériver les équations associées à un système en particulier. Dans l'exemple précédent concernant la modélisation de structures mécaniques, il pourrait s'agir de modéliser un pendule double avec certaines caractéristiques connues pour les deux membrures du pendule comme la longueur, la masse et l'inertie. La dérivation des équations pour différents systèmes d'une même famille à l'aide d'un même modèle est une tâche fastidieuse si elle est reproduite manuellement. C'est pourquoi, il est très utile de développer des outils permettant de dériver les équations de façon automatique pour une certaine famille de systèmes avec un modèle donné. L'usager n'a alors qu'à spécifier certaines caractéristiques propres au système à modéliser et l'outil de modélisation génère de façon automatique les équations associées à ce dernier.

Une fois les équations obtenues pour un système particulier, on peut le simuler dans le temps. Pour ce faire, le temps est divisé en pas de calcul de durée fixe ou variable appelé temps d'échantillonnage alors que la fréquence d'échantillonnage représente le nombre d'échantillons par unité de temps. Pour chaque pas de calcul, on effectue les calculs des états au temps présent en fonction des états précédents, des entrées et des paramètres. Un état représente une caractéristique du système à un temps donné dans la simulation et dépend de l'historique du système. Les entrées sont des valeurs extérieures qui permettent d'interagir avec le système. Les paramètres sont des grandeurs qui caractérisent un système physique et qui peuvent être modifiées ou non lors de la simulation. Les deux principaux logiciels à usage général utilisés dans l'industrie afin d'effectuer de telles simulations sont Simulink (The MathWorks Inc.) et MatrixX (Integrated Systems Inc.).

La simulation en temps réel consiste à simuler un processus aussi rapidement que dans la réalité. Pour ce faire, le temps nécessaire afin de simuler un pas de calcul doit être équivalent au temps réel de ce pas. En d'autres termes, la simulation d'un processus d'une durée t doit être inférieure ou égale à t, car on peut toujours ralentir une simulation si elle est trop rapide.

La modélisation et la simulation en temps réel sont des outils très utiles, sinon essentiels, dans plusieurs domaines de haute-technologie comme la robotique, l'aérospatial, les télécommunications, la météorologie, la génétique et l'étude des réseaux de distribution de l'électricité. Ces deux outils permettent, entre autre, d'entraîner des humains dans un environnement hostile où leur vie

peut être en danger (entraînement des astronautes, des pilotes d'avions), de tester certains systèmes face à des situations particulières (fuite dans une centrale nucléaire, panne dans un réseau électrique) et de prévoir certaines situations (prévisions météorologiques).

1.2 Modélisation de structures mécaniques

Lors du processus de conception d'une structure mécanique, il est très important d'évaluer les besoins et d'effectuer certains choix concernant la structure (matériaux utilisés, dimension, ...). Afin d'effectuer ces choix adéquatement, il est très utile de pouvoir simuler une structure mécanique avant de la construire afin d'étudier son comportement avec différentes caractéristiques dans un environnement donné. Pour ce faire, on doit tout d'abord générer certaines quantités physiques qui permettront d'effectuer la simulation, comme le Jacobien, la matrice d'inertie de la structure, la matrice de rigidité,... Évidemment la génération de ces grandeurs physiques demande une grande quantité de temps et d'efforts. De plus, lorsque la structure change, il est nécessaire de regénérer ces valeurs. Afin de pallier à ce problème, plusieurs logiciels de modélisation de structures mécaniques ont vu le jour. Le logiciel SYMOFROS[1] [PDLL99, MDP00] de l'Agence Spatiale Canadienne, le logiciel Robotran [MSW89b] de l'Université Catholique de Louvain ainsi que le logiciel commercial Dymola de Dynasim[2] [EBO96] sont des exemples de ce type d'outil de modélisation. Ces logiciels permettent, par exemple, d'obtenir automatiquement certaines quantités physiques reliées à une structure définie par un usager.

Le choix des modèles afin de représenter une structure mécanique consiste principalement à choisir des coordonnées afin de représenter la géométrie de la structure (cinématique) et de choisir une méthode afin de dériver les équations du mouvement de la structure (dynamique). La géométrie de la structure est habituellement représentée à l'aide de référentiels attachés aux membres de la structure. Les trois méthodes les plus utilisées afin de dériver les équations dynamiques sont : Newton-Euler, Lagrange-Euler et la méthode des puissances virtuelles aussi appelé méthode de Kane ou principe de d'Alembert généralisé. Les deux premières méthodes sont

[1] http ://extranet.space.gc.ca/symofros
[2] http ://dynasim.se

présentées dans la plupart des ouvrages de robotique, notamment dans Craig [Cra89]. La dernière méthode a été initialement développée par Jourdain [Jou09] plutôt que par Kane comme l'a montré Piedboeuf [Pie93]. Kane [Kan61, KL83, Kan83] a ensuite développé une méthodologie associée à cette méthode.

Comme Ferretti [Fer96] l'a fait remarquer, un logiciel de modélisation a deux buts qui entrent souvent en conflit : la généralité et l'efficacité. Un logiciel de modélisation est dit général s'il permet de modéliser des structures mécaniques quelconques dans un environnement quelconque. Les structures mécaniques peuvent être composées de membrures rigides, de membrures flexibles ou d'une combinaison des deux types de membrures. L'étude des structures contenant seulement des membrures rigides a fait l'objet d'un grand nombre de travaux qui sont bien résumés dans plusieurs ouvrages, dont ceux de Craig [Cra89] et Angeles [Ang97]. Plusieurs travaux de recherche récents, dont ceux de Piedboeuf [Pie95a, Pie95b, Pie96, Pie98] ont permis de trouver une représentation efficace pour les structures flexibles.

La présence de chaînes cinématiques fermées, qui transforment les équations de la dynamique d'équations différentielles ordinaires à équations différentielles algébriques, demande un traitement particulier. Afin d'inclure ces contraintes algébriques dans les équations de la dynamique, plusieurs méthodes ont été proposées. Une première méthode consiste à utiliser les multiplicateurs de Lagrange [dJB94]. Une autre méthode consiste à effectuer une projection des contraintes tel que présenté par Scott [Sco88] et Blajer [Bla92].

La simulation de structures mécaniques requiert la connaissance *a priori* de la structure et des différentes propriétés de cette dernière. Les différentes grandeurs physiques associées à la structure en tant que tel sont les paramètres. Les paramètres sont habituellement divisés en deux catégories : les paramètres cinématiques et les paramètres dynamiques. Le problème principal consiste donc à identifier la valeur de ces paramètres pour la structure à l'aide de méthodes d'identification. La résolution de ce problème a fait l'objet de plusieurs travaux de recherche, notamment ceux des chercheurs à l'Université catholique de Louvain [Rau90, RSG91, RCB+91, FRS95, OC97]. Nous présenterons plus en détail ces méthodes dans le chapitre 2.

1.3 Objectifs et aperçu du document

SYMOFROS (*Symbolic Modeling of Flexible Robots and Simulation*) est un logiciel de modélisation et de simulation pour des structures mécaniques développé initialement par Jean-Claude Piedboeuf de l'Agence Spatiale Canadienne et qui est l'exemple même d'un logiciel avec un haut niveau de généralité. En effet, SYMOFROS permet de générer un ensemble de grandeurs physiques associées à des structures mécaniques quelconques formées de membrures rigides ou flexibles pouvant contenir ou non des boucles cinématiques fermées. La modélisation dans SYMOFROS est effectuée de façon symbolique à l'aide du logiciel Maple et permet de générer les équations associées à la cinématique et à la dynamique du modèle non-linéaire et du modèle linéaire obtenu à l'aide d'une linéarisation du modèle autour d'un point d'opération. Tous ces aspects seront couverts avec plus de détails dans la prochaine section. De plus, nous présenterons quelques notions et travaux concernant l'estimation des paramètres dynamiques d'une structure mécanique.

Certains aspects de SYMOFROS concernant l'efficacité du logiciel seront améliorés dans la version 4.0 (par rapport à la version 3.2) :

1. Réduction de la quantité de mémoire utilisée lors de la génération symbolique des équations ;
2. Réduction du temps nécessaire afin de générer les équations qui serviront ensuite à la simulation ;
3. Plus grande flexibilité pour l'usager qui peut personnaliser la façon dont le code C est généré afin d'améliorer l'efficacité de la simulation.

Ces modifications feront l'objet de la section 3 et ne se limitent pas aux structures mécaniques. Les outils développés pourront être appliqués à d'autres outils de modélisation dans d'autres domaines d'applications. La principale modification consiste donc à effectuer l'optimisation lors de la création des équations plutôt que d'effectuer l'optimisation après la génération des équations. Nous verrons aussi comment nous réduirons le nombre de variables temporaires afin de réduire la quantité de mémoire utilisée et le temps nécessaire à la génération symbolique des équations. Une nouvelle approche sera proposée et implantée afin d'améliorer la linéarisation du modèle autour d'un point d'opération. Ceci permettra de réduire l'ordre du nombre de dérivées

à effectuer. Nous montrerons de façon théorique et expérimentale l'efficacité de la nouvelle approche par rapport à l'ancienne. Finalement, le groupement dynamique des fonctions par l'usager sera présenté afin de permettre à l'usager d'optimiser le code généré selon ses besoins.

Dans la section 4, nous présenterons les outils développés afin d'estimer les paramètres dynamiques pour une structure mécanique dans SYMOFROS. Nous proposerons une méthode automatique afin d'obtenir un ensemble minimal de paramètres en éliminant les paramètres qui ne sont pas identifiables et en regroupant les paramètres identifiables seulement en combinaisons linéaires. Par la suite, nous traiterons de l'estimation des paramètres dynamiques pour une trajectoire donnée.

Chapitre 2

Théorie, méthodes et outils utilisés

Dans cette section, nous présenterons certains outils et certains aspects théoriques qui nous seront par la suite essentiels. Nous introduirons tout d'abord certaines particularités de la version 5.1 du logiciel Maple utilisé par SYMOFROS afin de générer symboliquement les équations pour une structure mécanique donnée. Nous présenterons ensuite certains aspects théoriques importants en robotique, comme la linéarisation du modèle autour d'un point d'opération et le traitement des boucles cinématiques fermées. Ces aspects nous permettront de présenter et de mieux comprendre les principales étapes nécessaires à la modélisation de structures mécaniques dans la version 3.2 de SYMOFROS. Pour conclure la section, nous présenterons le problème de l'estimation des paramètres ainsi que quelques solutions proposées dans la littérature.

2.1 Calcul symbolique avec Maple 5.1

2.1.1 Manipulation des expressions

Maple est un logiciel de calcul symbolique développé initialement par l'université de Waterloo. Ce logiciel est de plus en plus utilisé dans l'industrie, surtout dans les domaines des mathématiques et de l'ingénierie en général.

Étudions tout d'abord la façon dont Maple manipule les expressions. Dans

les exemples suivants, les expressions alignées à gauche représentent les commandes entrées par l'usager alors que les expressions alignées au centre représentent les résultats dans Maple. Notons de plus que la commande *restart* permet de libérer la mémoire utilisée par Maple. Soit l'exemple suivant :

```
> restart;
> eq1 := sin(x)*cos(x);
                        eq1 := sin(x) cos(x)
> eq2 := cos(x)*sin(x);
                        eq2 := sin(x) cos(x)
> op(1,eq1);
                        sin(x)
> op(1,eq2);
                        sin(x)
```

On remarque que dans *eq2*, Maple a réorganisé les termes afin de mettre sin(x) avant cos(x). On pourrait donc supposer que Maple établit un ordre des termes pour une équation donnée (par exemple que sin vient toujours avant cos). Cependant, l'exemple suivant (qui correspond à l'équation précédente sauf que l'ordre des termes est inversé) montre que le procédé d'ordonnancement est plus subtil :

```
> restart;
> eq3 := cos(x)*sin(x);
                        eq3 := cos(x) sin(x)
> eq4 := sin(x)*cos(x);
                        eq4 := cos(x) sin(x)
> op(1,eq3);
                        cos(x)
> op(1,eq4);
                        cos(x)
```

Dans l'exemple précédent, l'expression cos(x) est en premier et sin(x) en deuxième. Donc Maple reconnait que *eq4* est équivalente à *eq3*, et reformule l'équation dans le même ordre. Ceci est aussi vrai pour des équations plus complexes comme *eq6* :

```
> restart;
> eq5 := cos(x)*y*sin(x)*x*z;
```

```
                          eq5 := cos(x) y sin(x) x z
> eq6 := sin(x)*cos(x)*z*x*y;
                          eq6 := cos(x) y sin(x) x z
> eq7 := y*x*sin(x)*cos(x);
                          eq7 := y x sin(x) cos(x)
```

Notons cependant que Maple ne réordonne pas les termes si l'expression n'est pas équivalente, même si elle est semblable (voir *eq7*).

2.1.2 Optimisation et variables temporaires

Un des grands problèmes du calcul symbolique est que la quantité de mémoire requise afin d'exprimer et de manipuler des expressions le moindrement complexe est gigantesque. C'est pourquoi il est important d'optimiser les équations. Dans Maple, la commande *optimize* permet d'optimiser des équations en les formulant à l'aide de variables temporaires. Voici un exemple qui illustre le principe des variables temporaires :

```
> restart:  readlib(optimize):
> eq := sin(x) + x*y + x*y*z;

    eq := sin(x) + x y + x y z

> optimize(eq);

    t1 = sin(x), t2 = x y, t4 = t1 + t2 + t2 z
```

L'optimisation permet de mettre certaines expressions en évidence permettant ainsi de réduire la quantité de mémoire utilisée et de réduire le nombre d'opérations nécessaires afin d'effectuer le calcul. En effet, dans l'exemple de *eq*, l'expression xy est calculée à une seule reprise plutôt qu'à deux reprises. On remarque que dans l'optimisation de *eq*, les variables temporaires sont t1, t2 et t4. Une question légitime serait : pourquoi ne pas avoir pris les variables t1, t2 et t3 (plutôt que t1, t2 et t4) ?

Afin de répondre à cette question, étudions tout d'abord le fonctionnement de la procédure *optimize*, qui peut être décomposée en trois étapes principales :

1. Prétraitement
2. Optimisation et création de la table *ass* contenant les variables temporaires. Par exemple, la table *ass* associée à *eq* est

```
> ass[0] => 4
> ass[1] => t1 = sin(x)
> ass[2] => t2 = x y
> ass[3] => t3 = t2 z
> ass[4] => t4 = t1 + t2 + t3
```

   Puisque la table *ass* est définie comme locale à la procédure, l'information contenue dans cette table est supprimée lorsque Maple sort de la procédure *optimize*. Il n'est donc pas possible de réutiliser cette table lors d'une prochaine optimisation.

3. Réduction de la table *ass* afin d'enlever les variables temporaires utilisées une seule fois dans la table *ass* et de remplacer ces variables dans les autres variables temporaires. On remarque que dans *ass[4]*, l'expression $t2\ z$ n'apparaît pas explicitement, mais sous la forme de $t3$. La substitution de cette expression est effectuée dans la procédure *optimize/remove* appelé dans la commande *optimize*. Maple compte en fait le nombre de fois qu'une expression est utilisée et enlève les variables temporaires qui sont utilisées une seule fois dans la table et qui respectent certaines conditions (par exemple : l'expression n'est pas de type *fonction*, l'expression n'est pas une addition de plus de 10 termes, ...). C'est donc pour cette raison que les variables temporaires sont $t1$, $t2$ et $t4$ plutôt que $t1$, $t2$ et $t3$.

2.1.3 Utilisation de l'option remember

Une des raisons de l'efficacité de la procédure *optimize* et de plusieurs autres procédures de Maple repose sur l'utilisation d'une table mémoire (à l'aide de *option remember*). Cette option permet de garder en mémoire le(s) résultat(s) de la procédure associé(s) aux entrées. Lors de l'appel de la procédure, il y a vérification afin de savoir si la procédure a déjà été appelée avec les mêmes paramètres. Si c'est le cas, Maple regarde dans la table et trouve la

valeur de la sortie correspondant aux entrées sans refaire les calculs déjà effectués précédemment. Dans la procédure *optimize*, l'*option remember* permet d'éviter de déclarer plusieurs variables temporaires pour une même expression. De plus, cette recherche dans la table est très rapide (elle est en fait linéaire [MGH+96]).

Donc, l'utilisation de l'option *remember* dans *optimize* et le fait que Maple exprime deux équations équivalentes sous la même forme nous permet de conclure qu'une table de variables temporaires ne contient jamais deux expressions identiques.

2.2 Modélisation de structures mécaniques

L'objectif de cette section est de donner un aperçu de certaines notions de mécanique qui nous seront utiles par la suite. Nous présenterons une méthode afin de dériver les équations de la dynamique pour une structure mécanique quelconque, en plus d'introduire les notions de base concernant la linéarisation du modèle autour d'un point d'opération.

2.2.1 Équations de la dynamique d'une structure mécanique en boucle ouverte

Dans l'introduction, nous avons mentionné que plusieurs modèles peuvent être utilisés afin de représenter les équations de la dynamique d'une structure mécanique. Dans le reste du document, nous utiliserons la méthode des puissances virtuelles afin de représenter les équations puisque c'est la méthode utilisée par SYMOFROS. Cette approche comporte plusieurs avantages comme l'a mentionné Piedboeuf [Pie95b].

En utilisant la méthode des puissances virtuelles, il est possible d'exprimer les équations de la dynamique d'une structure mécanique sous la forme d'équations différentielles non-linéaires de la forme :

$$\mathbf{M}(\mathbf{q}, \mathbf{u})\ddot{\mathbf{q}} + \mathbf{g}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}) = \mathbf{O} \tag{2.1}$$

où $\mathbf{q}$ est le vecteur de coordonnées généralisées, $\dot{\mathbf{q}}$ le vecteur des vitesses généralisées, $\ddot{\mathbf{q}}$ le vecteur des accélérations généralisées et $\mathbf{u}$, le vecteur d'en-

trées (trajectoire désirée, moments de forces appliqués aux articulations,...). La matrice $\mathbf{M}$ est la matrice d'inertie alors que le vecteur $\mathbf{g}$ contient les forces de Coriolis, les forces de gravité et les forces centrifuges.

L'utilisation de la méthode des puissances virtuelles permet d'exprimer $\mathbf{M}$ et $\mathbf{g}$ de façon récursive pour des membres rigides ou flexibles comme l'a montré Piedboeuf [Pie95b, Pie98].

2.2.2 Équations de la dynamique d'une structure mécanique en boucle fermée

Une boucle cinématique fermée apparaît lorsqu'une contrainte algébrique limite le mouvement possible d'une structure. Par exemple, les doigts d'une main forment une structure arborescente sans boucle cinématique lorsqu'aucun doigt ne se touche. Cependant, si deux doigts se touchent, le mouvement de l'un dépend de l'autre et le mouvement des deux doigts est alors limité. En d'autres termes, la présence de boucles cinématiques fermées entraîne une perte du nombre de degrés de liberté de la structure. L'équation de contraintes peut être exprimée sous la forme suivante :

$$\boldsymbol{\phi}(\mathbf{q}, t) = 0 \tag{2.2}$$

La méthode des multiplicateurs de Lagrange permet de tenir compte de ces contraintes algébriques dans les équations de la dynamique en ajoutant aux équations les multiplicateurs de Lagrange ($\boldsymbol{\lambda}$). Comme de Jalón et Bayo l'ont montré dans [dJB94], l'équation (2.1) peut alors s'écrire sous la forme :

$$\mathbf{M}(\mathbf{q}, \mathbf{u})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}) + \boldsymbol{\phi}_q^T \boldsymbol{\lambda} = \mathbf{0} \tag{2.3}$$

La grandeur $\boldsymbol{\phi}_q$ représente la matrice Jacobienne de l'équation de contraintes (2.2) par rapport à $\mathbf{q}$. Les multiplicateurs de Lagrange permettent d'ajouter les forces qui obligeront le système à respecter les contraintes de la structure.

2.2.3 Linéarisation autour d'un point d'opération

Afin de simuler le comportement de la structure, l'équation (2.1) doit être résolue. Dans plusieurs cas, la structure mécanique travaille près d'une confi-

guration donnée (configuration d'équilibre) nous permettant ainsi de linéariser les équations différentielles autour de ce point d'équilibre (voir Brauen et Nohel [BN86]). Cette linéarisation est appelée linéarisation du modèle autour d'un point d'opération. Les équations différentielles obtenues sont donc linéaires et permettent de simplifier les algorithmes de contrôle en plus de permettre l'analyse de certains aspects difficiles à analyser à l'aide du modèle non-linéaire (voir de Jalòn et Bayo[dJB94]). Il est aussi possible, dans certain cas, de linéariser une équation différentielle de façon exacte [Khe96]. Cependant, dans ce document, nous ne traiterons pas de la linéarisation exacte car elle n'est pas supportée dans SYMOFROS.

Afin d'obtenir les équations du modèle linéarisé, rappelons tout d'abord en quoi consiste le développement en série de Taylor. Soit une fonction $f(x)$ continue dont les m premières dérivées sont aussi continues sur un intervalle contenant le point x_0, alors on peut écrire $f(x)$ sous la forme :

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + ... + \frac{f(x_0)^{(m-1)}}{(m-1)!}(x - x_0)^{m-1} + R_m(x)$$

Pour une fonction $g(x, y)$ continue dont les m premières dérivées sont aussi continues sur un intervalle contenant le point (x_0, y_0), le développement en série de Taylor de la fonction $g(x, y)$ autour de $(x, y) = (x_0, y_0)$ est donnée par :

$$\begin{aligned} g(x,y) = g(x_0,y_0) + g_x(x_0,y_0)(x-x_0) + g_y(x_0,y_0)(y-y_0) \\ +\frac{1}{2!}\Big\{f_{xx}(x_0,y_0)(x-x_0)^2 + 2f_{xy}(x_0,y_0)(x-x_0)(y-y_0) \\ +f_{yy}(x_0,y_0)(y-y_0)^2\Big\} + ... \end{aligned}$$

La linéarisation de l'équation de la dynamique consiste à conserver le terme constant et les termes linéaires du développement en série de Taylor de l'équation (2.1) autour de $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{u}) = (\mathbf{q_0}, \dot{\mathbf{q}}_0, \ddot{\mathbf{q}}_0, \mathbf{u_0})$. Le modèle linéarisé peut donc s'exprimer sous la forme suivante :

$$\frac{d\,(\mathbf{M}\ddot{\mathbf{q}} + \mathbf{g})}{d\mathbf{q}} \Big|_{\mathbf{q}_0,\dot{\mathbf{q}}_0,\ddot{\mathbf{q}}_0,\mathbf{u}_0} \Delta\mathbf{q} + \frac{d\mathbf{g}}{d\dot{\mathbf{q}}} \Big|_{\mathbf{q}_0,\dot{\mathbf{q}}_0,\mathbf{u}_0} \Delta\dot{\mathbf{q}} +$$

$$\mathbf{M}\,|_{\mathbf{q}_0,\mathbf{u}_0}\ \triangle\ddot{\mathbf{q}} + \frac{d\,(\mathbf{M}\ddot{\mathbf{q}} + \mathbf{g})}{d\mathbf{u}}\,|_{\mathbf{q}_0,\dot{\mathbf{q}}_0,\ddot{\mathbf{q}}_0,\mathbf{u}_0}\ \triangle\mathbf{u} = 0 \tag{2.4}$$

2.3 Modélisation dans SYMOFROS 3.2

2.3.1 Aperçu général

Comme nous l'avons mentionné précédemment, le logiciel SYMOFROS permet de modéliser et de simuler des structures mécaniques définies par l'usager. Le processus consiste en trois étapes principales :

1. Description de la structure par l'usager
2. Génération symbolique des équations et du code C
3. Utilisation du code généré afin d'effectuer des simulations

La topologie de la structure mécanique est définie par l'usager dans l'interface de SYMOFROS ajoutée dans l'environnement de Simulink. Cette interface permet de créer schématiquement la structure composée de membres flexibles et/ou rigides, d'articulations et pouvant contenir des boucles cinématiques fermées. L'interface permet aussi de définir les paramètres reliés aux membres et aux articulations. Ces paramètres peuvent être soit nominaux (ces paramètres sont représentés par des variables dans le code généré et on peut modifier leurs valeurs lors de la simulation), soit numériques. À partir de cette description graphique, SYMOFROS génère plusieurs fichiers sous format texte décrivant la structure dans le langage de Maple.

Ensuite, les fonctions associées au modèle de l'usager sont obtenues de façon symbolique dans Maple. Dans ce document, une fonction correspond tout simplement à une grandeur physique associée à un modèle particulier comme par exemple le Jacobien et la matrice d'inertie. La génération symbolique consiste en plusieurs étapes qui seront présentées de façon plus détaillée dans la section 2.3.2. Par la suite, les équations sont optimisées et le fichier C contenant les fonctions est généré.

Finalement, les fonctions peuvent être utilisées à l'intérieur des *s-function* dans un schéma Simulink [Mat97].

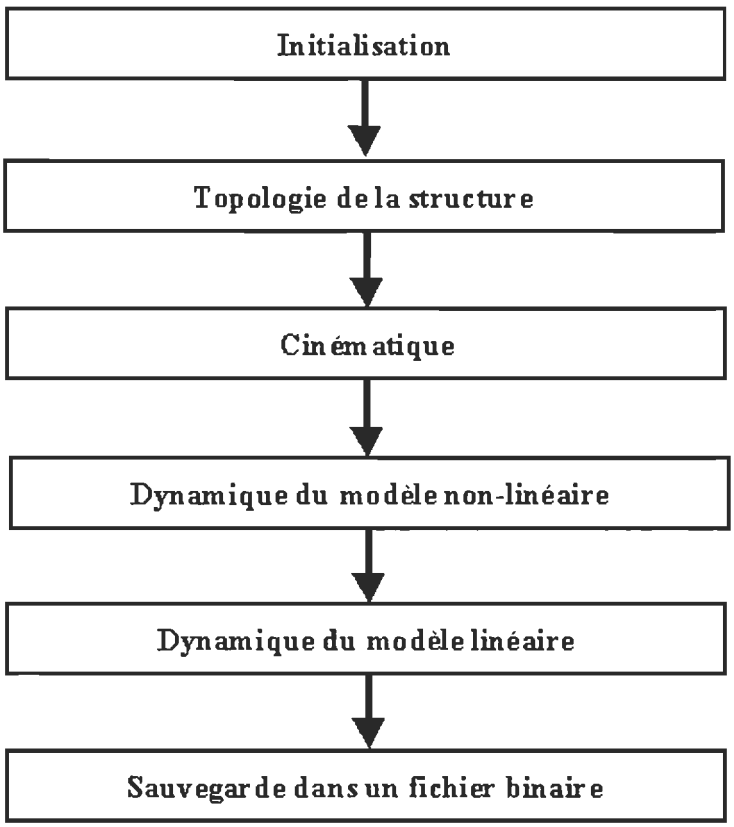

FIG. 2.1 – Principales étapes de la génération symbolique des équations dans SYMOFROS

2.3.2 Génération symbolique des équations

Un aperçu du processus de génération symbolique des équations dans Maple est présenté dans la figure 2.1.

La première étape consiste à lire les données de l'usager inscrites dans les fichiers textes créés à partir de l'interface graphique. Par la suite, SYMOFROS crée une structure arborescente de données contenant l'information sur la topologie de la structure.

Les équations de la cinématique sont ensuite générées pour chacune des

extrémités de la structure. On obtient alors les grandeurs suivantes par rapport au référentiel inertiel :

- Matrice de rotation
- Vecteur position
- Vecteur de vitesse angulaire
- Vecteur de vitesse
- Jacobien de rotation
- Jacobien de translation
- Dérivée du Jacobien de rotation
- Dérivée du Jacobien de translation

L'étape suivante consiste à obtenir les valeurs associées à la dynamique du modèle non-linéaire. Pour ce faire, SYMOFROS effectue les calculs suivants :

- Calcul du vecteur de gravité
- Calcul de l'énergie et de la puissance
- Calcul de la matrice d'inertie globale
- Calcul du vecteur non-linéaire global
- Calcul de la matrice d'amortissement interne
- Addition des effets gyroscopiques si nécessaire
- Calcul des valeurs associées aux contraintes
- Calcul du modèle rigide qui consiste à mettre les coordonnées flexibles et élastiques à zéro

Les fonctions ainsi générées (matrice d'inertie, jacobien en rotation ,...) sont ensuite regroupées en groupes de fonctions dans SYMOFROS. Ce regroupement permet de générer du code plus efficace en évitant de faire des calculs redondants lors de la simulation. Par exemple, soit deux fonctions f_1 et f_2 d'un groupe g_i qui contiennent toutes deux le calcul du sinus d'un angle ϕ, dont la valeur change dans le temps. Le groupement de ces fonctions permettra de calculer cette valeur une seule fois par pas de calcul au lieu de la calculer deux fois par pas de calcul.

Les équations de la dynamique associées au modèle linéarisé sont ensuite obtenues en linéarisant le modèle autour d'un point d'opération (voir section 2.3.5). Les équations associées aux modèles suivants sont ensuite obtenues :

- Modèle sans contrainte

- Contraintes permanentes
- Contraintes intermittentes

Finalement, les équations sont sauvegardées dans un fichier binaire. Ce fichier sera ensuite utilisé afin de générer le code C.

2.3.3 Génération du code C

La première étape afin de générer le code C consiste à lire les équations dans le fichier binaire créé lors de la génération symbolique des équations. Ensuite, pour chaque groupe de fonctions, les équations de la table de variables temporaires sont optimisées à l'aide de la commande *optimize* de Maple et les équations associées à chaque groupe de fonctions sont inscrites dans le fichier C.

2.3.4 Manipulation et optimisation des équations

Dans les premières étapes de la génération symbolique des équations (initialisation, cinématique et dynamique), toutes les expressions sont optimisées en fonction des variables temporaires inscrites dans la même table de variables temporaires. Il y a création d'une variable temporaire pour chaque élément d'un vecteur ou d'une matrice qui respecte une des trois règles suivantes :

- L'expression est de type *function* dans Maple (ex : fonctions trigonométriques).
- L'expression est une somme ou une exponentielle contenant au moins une variable indexée. Ces variables indexées représentent soit un état (**x**[]), une entrée (**u**[]) ou une variable temporaire (z_t[]).
- L'expression contient au moins deux variables indexées.

Ceci permet de réduire la quantité de mémoire en évitant de répéter plusieurs fois la même expression dans plusieurs expressions. En effet, dans SYMOFROS, la plupart des expressions obtenues sont des produits de plusieurs matrices ou vecteurs. Le simple produit de trois matrices pourra facilement nous convaincre de l'importance de ces variables temporaires :

$$\begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix} \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix}$$

$$= \begin{bmatrix} a_1b_1c_1 + a_2b_3c_1 + a_1b_2c_3 + a_2b_4c_3 & a_1b_1c_2 + a_2b_3c_2 + a_1b_2c_4 + a_2b_4c_4 \\ a_3b_1c_1 + a_4b_3c_1 + a_3b_2c_3 + a_4b_4c_3 & a_3b_1c_2 + a_4b_3c_2 + a_3b_2c_4 + a_4b_4c_4 \end{bmatrix}$$

L'utilisation des variables temporaires permet d'écrire le résultat précédent de façon récursive :

$$\begin{bmatrix} z_5 & z_6 \\ z_7 & z_8 \end{bmatrix}$$

où

$$\begin{aligned} z_1 &= a_1b_1 + a_2b_3 \\ z_2 &= a_1b_2 + a_2b_4 \\ z_3 &= a_3b_1 + a_4b_3 \\ z_4 &= a_3b_2 + a_4b_4 \\ z_5 &= z_1c_1 + z_2c_3 \\ z_6 &= z_1c_2 + z_2c_4 \\ z_7 &= z_3c_1 + z_4c_3 \\ z_8 &= z_3c_2 + z_4c_4 \end{aligned}$$

Plus la taille des matrices sera grande, plus il y aura de multiplications et plus l'optimisation permettra de sauver de la mémoire. L'autre avantage est que l'optimisation permet de réduire le nombre d'opérations de façon considérable. Le tableau suivant indique le nombre d'opérations nécessaire afin de calculer le résultat de la multiplication matricielle précédente de façon non-récursive et récursive :

	Additions	Multiplications
Non-récursif	12	32
Récursif	8	16

Cependant, le principal problème de la méthode utilisée afin d'optimiser les équations lors de la génération symbolique des équations dans SYMOFROS

3.2 est que plusieurs variables de la table peuvent contenir la même expression. Donc, il y a une certaine redondance dans la table de variables temporaires. La procédure *RemoveRedundant* de SYMOFROS permet d'enlever ces termes redondants de la table de variables temporaires (juste avant la linéarisation du modèle). Cette étape demande une quantité importante de temps.

Par la suite, il y a création d'une table de variables temporaires pour chaque groupe de fonctions (que nous appellerons table locale) à partir de la table globale. Notons que ces tables locales demandent une quantité importante de mémoire sans apporter d'informations supplémentaires et nécessaires à la génération des équations.

Une fois la linéarisation du modèle autour d'un point d'opération effectuée, il y a élimination des termes redondants de la nouvelle table de variables temporaires créée lors de la linéarisation et création de tables locales pour chaque groupe de fonctions issu de la linéarisation.

Une fois le tout complété, pour chaque table locale, il y a optimisation et écriture des équations dans le fichier C. À ce niveau, l'optimisation est effectuée à l'aide de la commande *optimize* de Maple. Notons que lors de l'optimisation des tables locales, la table optimisée calculée est directement écrite dans le fichier C et peut donc être effacée de la mémoire avant de créer la table locale suivante.

Les étapes présentées précédemment concernant l'optimisation et la manipulation des variables temporaires sont illustrées dans la figure 2.2.

2.3.5 Linéarisation autour d'un point d'opération

Dans la section 2.2, nous avons présenté en quoi consiste la linéarisation du modèle autour d'un point d'opération. Nous avons vu que nous devons effectuer la dérivée de certaines quantités par rapport à $\mathbf{q}$, $\dot{\mathbf{q}}$ et $\mathbf{u}$. La dérivation d'une expression de façon symbolique dans Maple est une tâche simple dans le cas où l'expression est exprimée de façon non-récursive. Cependant, dans le cas de SYMOFROS, les expressions à dériver sont exprimées à l'aide de variables temporaires récursives. Par exemple, une expression peut dépendre d'une variable temporaire z_n, qui peut à son tour dépendre de la variable

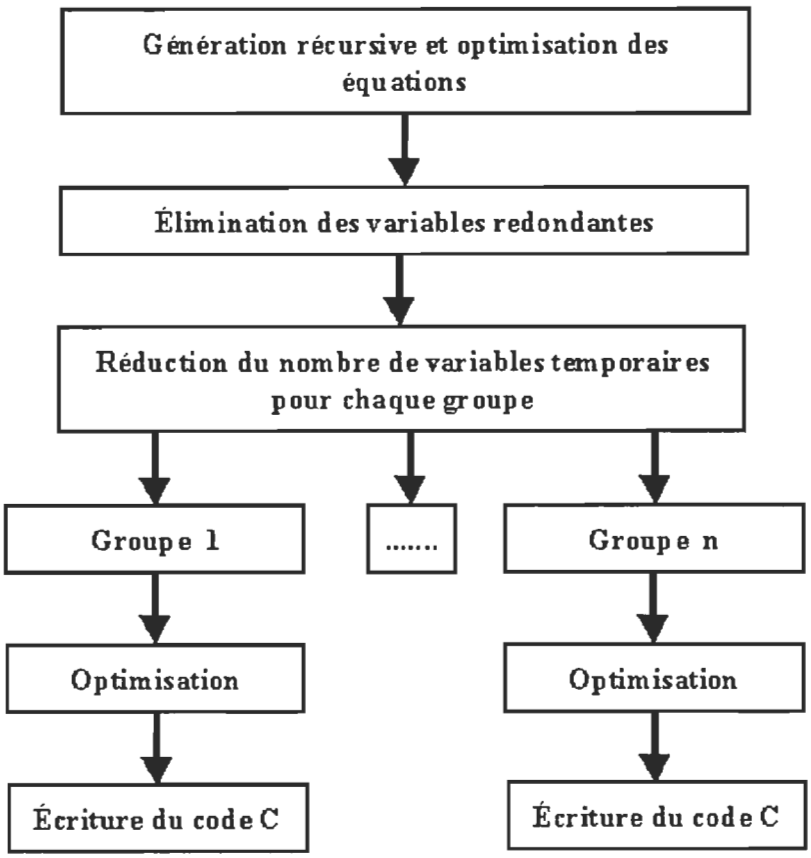

FIG. 2.2 – Optimisation et manipulation des expressions dans SYMOFROS 3.2

z_{n-1}, etc...Nous pouvons exprimer cette dépendance de la façon suivante :

$$z_n = f(z_1, ..., z_{n-1}, \mathbf{x}, \mathbf{u})$$

où $\mathbf{x}$ représente les états et les accélérations du système $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ et $\mathbf{u}$ les entrées.

Définissons tout d'abord l'opérateur D_q comme la dérivée partielle d'une expression non récursive par rapport à la variable q. L'exemple suivant montre la différence entre D_q et $\frac{\partial}{\partial q}$ pour une expression écrite de façon récursive.

Exemple 2.1 *Soit les variables temporaires suivantes :*

$$\begin{aligned} z_1 &= q^2 \\ z_2 &= L \cdot z_1 \end{aligned}$$

En dérivant ces expressions par rapport à q on obtient que :

$$\frac{\partial z_1}{\partial q} = 2q$$

$$\frac{\partial z_2}{\partial q} = 0$$

alors que

$$D_q z_2 = D_q \left(L \cdot q^2 \right) = 2 \cdot L \cdot q$$

Une autre façon de calculer la valeur de $D_q z_2$ consiste à effectuer la dérivée en chaîne :

$$D_q z_2 = \frac{\partial z_2}{\partial z_1} \cdot \frac{\partial z_1}{\partial q} + \frac{\partial z_2}{\partial q} = (L) \cdot (2 \cdot q) + 0 = 2 \cdot L \cdot q$$

Nous pouvons donc généraliser le principe de la dérivée en chaîne pour une expression A exprimée à l'aide de variables temporaires z_i de la façon suivante :

$$D_v A = \left(\sum_{i=1}^{n} \frac{\partial A}{\partial z_i} D_v z_i \right) + \frac{\partial A}{\partial v} \tag{2.5}$$

L'exemple suivant illustre cette façon de procéder :

Exemple 2.2 *Soit les variables temporaires suivantes :*

$$\begin{aligned} z_1 &= q^2 \\ z_2 &= q \cdot z_1 \\ z_3 &= q \cdot z_1 \cdot z_2 \end{aligned}$$

Alors on obtient que :

$$\frac{\partial z_1}{\partial q} = D_q z_1 = 2q$$

$$\frac{\partial z_2}{\partial q} = z_1$$

$$D_q z_2 = \frac{\partial z_2}{\partial z_1} \cdot D_q z_1 + \frac{\partial z_2}{\partial q} = (q)(2q) + z_1 = 3q^2$$

$$\frac{\partial z_3}{\partial q} = z_1 \cdot z_2$$

$$\begin{aligned} D_q z_3 = \frac{\partial z_3}{\partial z_1} \cdot D_q z_1 + \frac{\partial z_3}{\partial z_2} \cdot D_q z_2 + \frac{\partial z_3}{\partial q} &= (z_2 \cdot q)(2q) + (z_1 \cdot q)(3q^2) + z_2 \cdot z_1 \\ &= 2q^5 + 3q^5 + q^5 = 6q^5 \end{aligned}$$

Ce résultat est le bon car $z_3 = q \cdot q^2 \cdot q^3 = q^6$.

De la même façon, on peut effectuer la dérivée en chaîne par rapport à plusieurs variables à la fois en utilisant le Jacobien. Soit $\mathbf{v} = [v_1, ..., v_k]$, un vecteur, alors

$$\mathbf{D}_{\mathbf{v}} A = \left(\sum_{i=1}^{n} \frac{\partial A}{\partial z_i} \mathbf{D}_{\mathbf{v}} z_i \right) + J(A, \mathbf{v})$$

où

$$\mathbf{D}_{\mathbf{v}} z_i = [D_{v_1} z_i, ..., D_{v_k} z_i]$$

$$J(A, \mathbf{v}) = \left[\frac{\partial A}{\partial v_1}, ..., \frac{\partial A}{\partial v_k} \right]$$

La dérivation dans SYMOFROS 3.2 se fait à l'aide de l'équation précédente. Cependant, les calculs des $\mathbf{D}_{\mathbf{z_i}}$ pour toutes les variables temporaires z_i de la table de variables temporaires demandent beaucoup de temps car ils nécessitent l'appel de la procédure de différentiation un très grand nombre de fois. Au chapitre 3, nous présenterons une approche légèrement différente qui permettra de réduire ces coûts de façon considérable.

2.3.6 Problèmes associés à SYMOFROS 3.2

Voici une synthèse des principaux problèmes de la version 3.2 de SYMOFROS qui ont motivé les modifications qui seront présentées dans le chapitre 3 :

1. Les fonctions sont groupées de façon statique, c'est-à-dire que l'usager n'a aucun contrôle sur ce regroupement. De plus, le calcul d'une fonction du groupe nécessite le calcul de tout le groupe, incluant les autres fonctions du groupe. Cette approche est évidemment très contraignante lorsque l'usager désire faire une simulation en temps réel.
2. La quantité de mémoire utilisée lors de la génération symbolique des équations est énorme. Ceci est une conséquence, entre autre, d'une redondance dans l'information contenue dans les structures de données, notamment lors de la création des variables temporaires et de l'assignation d'une table de variables temporaires à chacun des groupes.
3. Les expressions sont optimisées à deux reprises : pendant la génération symbolique et lors de l'écriture des groupes de fonctions dans le fichier C.
4. La linéarisation du modèle autour d'un point d'opération est très coûteuse en temps et en mémoire.

2.4 Estimation des paramètres dynamiques

Les équations de la dynamique générées lors de la modélisation sont exprimées à l'aide de différents paramètres associés à des caractéristiques physiques de la structure comme la longueur et la masse. Lors de la génération symbolique des équations, les valeurs de ces paramètres n'ont pas besoin d'être connues car ils peuvent être simplement remplacés par des variables. Cependant, lorsque l'on veut simuler le comportement d'une structure mécanique

dans le temps, il est essentiel de connaître les valeurs de ces paramètres.

Les paramètres de la structure se divisent en deux grandes catégories : les paramètres cinématiques et les paramètres dynamiques. Les paramètres cinématiques concernent seulement la géométrie de la structure, par exemple la longueur. Les autres paramètres sont les paramètres dynamiques et comprennent notamment les paramètres inertiels qui caractérisent la répartition de la masse dans la structure. Dans ce document, nous nous intéresserons seulement à l'estimation des paramètres dynamiques en posant l'hypothèse que les paramètres cinématiques sont connus.

2.4.1 Formulation du problème et choix des paramètres

Une première approche afin d'estimer les paramètres dynamiques consiste à mesurer les paramètres de chacun des membres de la structure individuellement. Cependant, cette méthode est peu souhaitable pour l'usager et parfois même impossible car l'usager doit alors démonter la structure. Il est donc nécessaire d'estimer ces paramètres expérimentalement, en tenant compte de la structure comme un tout. Soit $\boldsymbol{\phi} \in \Re^{n \times p}$, $\boldsymbol{\delta} \in \Re^{p \times 1}$ et $\mathbf{Z} \in \Re^{n \times 1}$, où n représente le nombre de coordonnées généralisées et p le nombre de paramètres à estimer. L'approche la plus couramment utilisée consiste à isoler les paramètres dynamiques $\boldsymbol{\delta}$ dans les équations de la dynamique afin d'obtenir une équation de la forme $\boldsymbol{\phi\delta} = \mathbf{Z}$, avec $\boldsymbol{\phi}$ et $\mathbf{Z}$ indépendants des paramètres dynamiques $\delta_i \in \boldsymbol{\delta}$. Pour ce faire, les paramètres δ_i doivent apparaître de façon linéaire dans les équations de la dynamique. Il est aussi possible d'utiliser le modèle énergétique plutôt que les équations de la dynamique afin d'estimer les paramètres comme l'ont entre autre proposé Gautier et Khalil [GK88, GK90, GK92].

Les paramètres inertiels habituellement choisis pour une structure mécanique rigide sont la masse, les moments du premier ordre et les moments du second ordre (les moments d'inertie) pour chacun des membres de la structure car ils apparaissent de façon linéaire dans les équations de la dynamique. Un autre choix des paramètres inertiels consiste à regrouper la masse, les moments du premier ordre et les moments du second ordre entre eux afin de former les paramètres barycentriques tel que présenté par Raucent [Rau90]. Maes, Sa-

min et Willems [MSW89a] ont montré que ces paramètres apparaissent aussi de façon linéaire dans les équations de la dynamique. Dans les deux cas, nous obtenons 10 paramètres inertiels à estimer pour chacun des membres rigides de la structure.

Le choix des paramètres pour une structure flexible dépend de nombreux facteurs, dont la méthode de modélisation utilisée ainsi que les hypothèses effectuées afin de modéliser la structure. Il est néanmoins possible de classifier les paramètres dynamiques à estimer pour une membrure flexible en deux grandes catégories : les paramètres inertiels ainsi que les paramètres associés à la flexibilité de la structure. En utilisant le modèle énergétique et à l'aide de plusieurs hypothèses, Martineau [Mar96] a proposé un ensemble total de 8 paramètres composés de 4 paramètres inertiels, soit un paramètre associé à la masse volumique ainsi que trois paramètres pour les moments d'inertie principaux, et 4 paramètres associés à la flexibilité de la structure, soit 3 modules d'élasticité transversale et un module d'élasticité longitudinale. En utilisant la méthode des puissances virtuelles, Oliviers et Campion [OC97] ont proposé un ensemble de paramètres barycentriques pour une membrure flexible qui consiste en un ensemble de 14 paramètres : la masse, le vecteur de position du centre de masse ainsi que 10 paramètres permettant de caractériser l'inertie, soit une matrice symétrique, un vecteur et un scalaire.

2.4.2 Détermination de l'ensemble minimal de paramètres identifiables

Une fois le choix des paramètres effectué, il est possible de classer les paramètres dans trois catégories :

1. Les paramètres qui ne sont pas identifiables
2. Les paramètres identifiables individuellement
3. Les paramètres identifiables seulement en combinaisons linéaires

Les paramètres qui ne sont pas identifiables sont les paramètres qui n'apparaissent pas dans les équations de la dynamique. Les paramètres identifiables seulement en combinaisons linéaires sont des paramètres qui dépendent de façon linéaire des mêmes variables et ne peuvent pas être estimés individuellement. Il est donc possible de réduire le nombre de paramètres en gardant

seulement des paramètres ou des combinaisons de paramètres qui sont identifiables et indépendants entre eux. Nous appellerons cet ensemble de paramètres l'ensemble minimal ou l'ensemble de base. Notons que l'ensemble minimal n'est pas un ensemble unique mais que le nombre d'éléments (que nous appellerons nombre minimal de paramètres) dans l'ensemble est quant à lui unique.

Plusieurs méthodes ont été proposées afin de trouver l'ensemble de base pour une structure rigide sans boucle cinématique fermée. Gautier et Khalil [GK88, GK90] ont présenté des règles afin de réduire le nombre de paramètres dynamiques à partir de la topologie de la structure en utilisant les équations de l'énergie cinétique et de l'énergie potentielle. Cette méthode prend avantage de la structure des équations de Lagrange-Euler afin de réduire le nombre de paramètres. Raucent [Rau90] a présenté une méthode ainsi que des règles de regroupement lorsque les paramètres barycentriques sont utilisés alors que Fisette, Raucent et Samin [FRS96] ont montré comment déterminer l'ensemble de base pour une structure arborescente lorsque les paramètres barycentriques sont utilisés. Toutes les méthodes proposées précédemment considèrent seulement des articulations à un degré de liberté (rotatives ou prismatiques).

Des méthodes afin d'estimer les paramètres pour une structure flexible ont été présentées par Dépincé [DCB93], par Martineau [Mar96], par Chedmail *et al* [CDGP93] ainsi que par Oliviers et Campion [OC97]. L'estimation de paramètres pour une structure avec des boucles fermées a fait l'objet de plusieurs travaux de recherche récents, dont ceux de Fisette [FRS95], de Gautier [GKR95] et de Lipinsky [LFRS91].

Les méthodes proposées dans la littérature sont très spécifiques, c'est-à-dire qu'elles sont seulement applicables sur des structures particulières. De plus, ces méthodes dépendent fortement de la forme des équations utilisées (équation de la dynamique, modèle énergétique) ainsi que de la méthode de modélisation utilisée.

2.4.3 Choix de la trajectoire

Une fois l'ensemble de base obtenu, nous devons évaluer la valeur des paramètres de l'ensemble minimal à l'aide de la structure mécanique. Pour ce faire,

il est nécessaire de choisir une trajectoire qui nous permet d'évaluer avec le plus de précision possible la valeur des paramètres malgré les imprécisions des mesures et la présence de bruits.

Plusieurs approches ont été proposées afin de trouver de telles trajectoires pour une structure rigide sans boucle cinématique fermée. Armstrong [Arm89] a proposé une mesure basée sur la condition d'une matrice de corrélation afin de trouver une trajectoire qui maximise l'excitabilité pour une structure donnée. Cette formulation demande cependant une quantité énorme de calculs. Pour cette raison, Gautier et Khalil [GK92] ont proposé une approche basée sur le modèle énergétique permettant d'obtenir des trajectoires excitantes en effectuant un nombre réduit de calculs. Swevers *et al* [SGT+97b] ont proposé une approche statistique afin de générer les trajectoires. Ils présentent entre autre deux critères basés sur la matrice de covariance pouvant être utilisés afin de générer des trajectoires optimales. Un rapport écrit par les mêmes auteurs [SGT+97a] décrit de façon détaillée l'implémentation de l'algorithme présenté dans l'article.

2.4.4 Estimation des paramètres pour une trajectoire donnée

Une fois la trajectoire trouvée, nous obtenons une série de N valeurs pour $\mathbf{q}$ $(\mathbf{q}^{(1)}, ..., \mathbf{q}^{(N)})$, où chaque $\mathbf{q}^{(i)}$ est un vecteur de n éléments. À l'aide des $\mathbf{q}^{(i)}$, il est aussi possible de trouver les valeurs des $\dot{\mathbf{q}}^{(i)}$ et des $\ddot{\mathbf{q}}^{(i)}$, pour $i = 1, ..., N$. En fournissant ces valeurs au contrôleur de la structure mécanique réelle, il est possible de mesurer les moments de forces nécessaires afin que la structure réelle se déplace selon la trajectoire désirée. Par conséquent, à l'aide de la trajectoire et des mesures des couples appliqués à la structure, il est possible de déterminer numériquement les valeurs de $\boldsymbol{\phi}$ et $\mathbf{Z}$ pour chacun des points de la trajectoire. Cette méthode est appelée méthode interne car elle demande la mesure des forces et des moments de forces tel que mesurés dans les actionneurs de la structure. La méthode externe demande plutôt la mesure des forces et moments de forces à la base de la structure.

Notons par $\boldsymbol{\phi}^{(i)}$ la valeur de $\boldsymbol{\phi}$ au point i et $\mathbf{Z}^{(i)}$ la valeur de $\mathbf{Z}$ au point i. Il est alors possible d'écrire :

$$\Phi\delta = \begin{bmatrix} \phi^{(1)} \\ \vdots \\ \phi^{(i)} \\ \vdots \\ \phi^{(N)} \end{bmatrix} \delta = \begin{bmatrix} \mathbf{Z}^{(1)} \\ \vdots \\ \mathbf{Z}^{(i)} \\ \vdots \\ \mathbf{Z}^{(N)} \end{bmatrix} = \zeta$$

En supposant que le nombre de points N dans la trajectoire est suffisamment grand, c'est-à-dire que $n \cdot N > p$, alors le système d'équations est surdéterminé. L'estimation des paramètres consiste donc à estimer la valeur des éléments de δ.

L'estimation des paramètres est traité en détail par Richalet [Ric98]. Plusieurs méthodes sont proposées, chacune ayant ses avantages et ses inconvénients. Il faut donc choisir une méthode appropriée pour un problème donné.

Une des méthodes les plus utilisées est la méthode des moindres carrés. Supposons que ϕ est de plein rang, alors la méthode des moindres carrés consiste à trouver δ' qui minimise le carré des erreurs :

$$\|\Phi\delta' - \zeta\|^2$$

Plusieurs méthodes afin de résoudre ce problème sont présentées notamment par Golub et Van Loan[GL96]. Le principal défaut de cette méthode est qu'elle est biaisée. Swevers *et al* [SGT+97b, SGT+97a] ont montré qu'il est possible d'enlever le biais si les variances du bruits affectant les mesures sont connues.

2.5 Conclusion

Le calcul symbolique est un outil essentiel afin de modéliser un processus quelconque. Cependant, l'utilisation du calcul symbolique requiert une quantité énorme de mémoire. Pour cette raison, il est essentiel d'optimiser les expressions. Dans le logiciel Maple, cette optimisation est effectuée à l'aide de la procédure *optimize*. Cette procédure permet une optimisation rapide

et efficace grâce notamment à l'*option remember*. L'utilisation de cette option permet de créer une table de variables temporaires qui ne contient aucun terme redondant. Cependant, cette procédure d'optimisation est efficace seulement si toutes les expressions sont connues au moment de l'optimisation. Il n'est donc pas possible d'optimiser les expressions pendant le processus de création des expressions, ce qui est une grande limitation pour les problèmes complexes.

La génération des équations de la dynamique d'une structure mécanique peut se faire à l'aide de la méthode des puissances virtuelles. Cette méthode permet de représenter le comportement dynamique d'une structure arborescente ou d'une structure contenant des boucles cinématiques fermées à l'aide d'équations différentielles non-linéaires. La linéarisation autour d'un point d'opération permet d'obtenir des équations linéaires à partir des équations non-linéaires en conservant les termes linéaires du développement en série de Taylor.

Le processus de génération des fonctions dans la version 3.2 de SYMOFROS consiste en deux étapes principales : la génération symbolique des équations et la génération du code C. La méthode utilisée afin de manipuler les expressions pendant le processus de génération ainsi que certains problèmes associés à cette manipulation de l'information ont été présentés en détail dans cette section.

L'estimation ou l'identification des paramètres consiste à déterminer les valeurs numériques des paramètres afin de pouvoir simuler la structure mécanique. Le processus d'identification peut être divisé en plusieurs étapes : le choix des paramètres apparaissant de façon linéaire dans les équations, le regroupement des paramètres afin d'obtenir l'ensemble de base, la génération de trajectoires excitantes ainsi que l'estimation des valeurs des paramètres en fonction de la trajectoire et des mesures obtenues expérimentalement.

Chapitre 3

Modélisation de structures mécaniques dans SYMOFROS 4.0

Les modifications apportées dans la version 4.0 de SYMOFROS portent surtout sur la manipulation et l'optimisation des équations ainsi que sur les structures de données. La représentation de la structure et la méthode de modélisation utilisée afin d'obtenir les équations de la cinématique et de la dynamique restent quant à elles inchangées, ce qui implique que les grandes étapes du processus de génération présentées pour SYMOFROS 3.2 dans la figure 2.1 sont les mêmes pour la version 4.0.

La figure 3.1 illustre les principales étapes du processus de modélisation dans SYMOFROS 4.0 concernant la manipulation des équations issues de la cinématique et de la dynamique.

La première étape consiste à optimiser les équations pendant la génération symbolique à l'aide de la nouvelle procédure d'optimisation *optimize*. Cette nouvelle procédure d'optimisation est basée sur la procédure *optimize* de Maple tel que présentée dans le chapitre 2 et fera l'objet de la première sous-section. Nous présenterons ensuite une nouvelle méthode beaucoup plus rapide afin de linéariser le modèle autour d'un point d'opération. Une fois les équations générées en fonction des variables temporaires pour la cinématique et la dynamique des modèles non-linéaire et linéaire, nous devons réduire le nombre de variables temporaires utilisées. Par la suite, le code C est inscrit

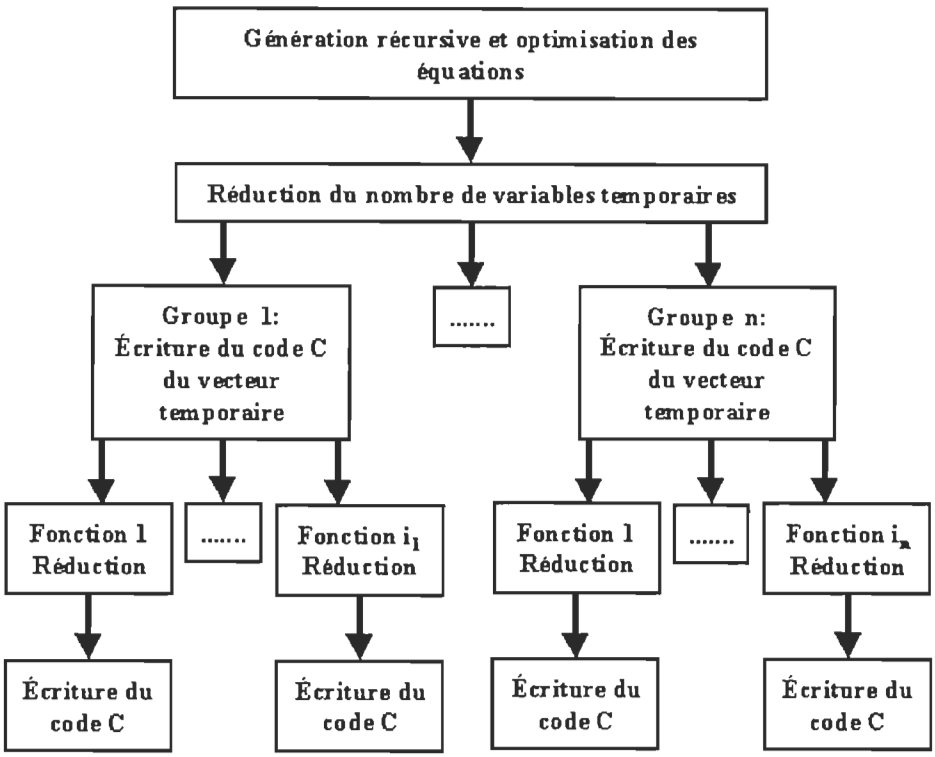

FIG. 3.1 – Optimisation et manipulation des expressions dans SYMOFROS 4.0

dans le fichier pour chaque groupe de fonctions sous forme d'un vecteur temporaire et de fonctions. Chaque vecteur temporaire contient des expressions apparaissant dans plusieurs fonctions du groupe, permettant ainsi de générer du code de simulation qui est plus efficace.

Finalement, nous présenterons les avantages et les inconvénients de la nouvelle version de SYMOFROS par rapport à la version 3.2 à l'aide de deux modèles de structures mécaniques. Nous verrons que la nouvelle structure de SYMOFROS est beaucoup plus flexible pour l'usager et que la linéarisation est beaucoup plus rapide. Cependant, la grande quantité de variables temporaires générées ralentit l'étape de la réduction de la table de variables temporaires.

3.1 Optimisation des équations

Comme nous l'avons mentionné au chapitre 2, l'optimisation des équations dans SYMOFROS 3.2 est effectuée à deux reprises : lors de la génération symbolique des équations et lors de l'écriture des fonctions dans le fichier C. De plus, la méthode utilisée comporte certains problèmes qui ont été mentionnés dans la section 2.3.4. Afin de remédier à ces problèmes, une nouvelle approche est proposée afin d'optimiser les expressions seulement lors de la génération symbolique des équations en exprimant toutes les expressions à l'aide de variables temporaires d'une seule table de variables temporaires que nous appellerons la table globale. Nous verrons aussi que cette méthode comporte plusieurs avantages par rapport à la méthode utilisée dans la version 3.2 de SYMOFROS.

3.1.1 Modification de la procédure *optimize* de Maple

Dans la section 2.1.2, nous avons présenté la procédure *optimize* de Maple et nous avons montré que cette procédure peut être subdivisée en trois étapes distinctes : pré-traitement, optimisation et post-traitement. Les modifications apportées portent sur les deux dernières étapes.

La première modification apportée à la procédure *optimize* de Maple consiste à déclarer la table contenant les variables temporaires comme une variable globale plutôt que comme une variable locale à la procédure. Cette modifi-

cation permettra de réutiliser la même table pour l'optimisation d'une expression subséquente. Cependant, puisque la table est maintenant déclarée comme une table globale, l'usager doit préalablement initialiser certaines variables (voir annexe A).

La dernière étape (post-traitement) de la commande *optimize* consiste à substituer les variables temporaires qui sont utilisées une seule fois par l'équation qui est optimisée. Cependant, dans la nouvelle procédure d'optimisation, cette approche réduira l'efficacité du code généré car pendant la génération des équations, il est impossible de savoir si une expression apparaîtra ultérieurement et donc nous ne pouvons pas réduire la taille de la table de variables temporaires pendant la génération sans remplacer les termes qui seront réutilisés ultérieurement. Pour cette raison, le post-traitement effectué dans la commande *optimize* a été enlevé de la procédure *optimize*. Ce post-traitement sera effectué lorsque toutes les équations auront été générées et fera l'objet de la section 3.2.

Voici un exemple qui permettra de comparer la procédure *optimize* de Maple et la procédure *optimize* modifiée jointe en annexe (annexe A) :

Exemple 3.1 *Soit les commandes suivantes entrées dans Maple*

```
> restart;
> readlib(optimize);
> eq1 := x*sin(x) + y*cos(y);
> eq2 := x*sin(x) + z*w;
> eq3 := y*cos(y) + w*z + x*y;
```

*Le problème consiste à optimiser les expressions eq1, eq2 et eq3 l'une après l'autre. En d'autres termes, nous supposons les expressions eq2 et eq3 inconnues lors de l'optimisation de eq1 et nous supposons eq3 inconnue lors de l'optimisation de eq2. Cette supposition permet de tenir compte du fait que l'on veut optimiser les expressions lors de leur création. Remarquons que plusieurs termes (x*sin(x), w*z, y*cos(y)) sont communs à plusieurs expressions. Voici le résultat obtenu lorsque nous optimisons les expressions précédentes à l'aide de la commande optimize de Maple :*

```
> optimize(eq1);
                        t5 = x sin(x) + y cos(y)
```

```
> optimize(eq2);
                        t4 = x sin(x) + z w
> optimize(eq3);
                        t5 = y cos(y) + z w + x y
```

Les équations sont alors optimisées une par une sans tenir compte des précédentes. Les termes communs ne peuvent donc pas être mis en évidence. En optimisant les expressions avec la nouvelle procédure optimize nous obtenons :

```
> optimize(eq1, evaln(Z_t));
                                        z[5]
> eval(Z_t);
  table([
        0 = 5
        1 = sin(x)
        2 = x z[1]
        3 = cos(y)
        4 = y z[3]
        5 = z[2] + z[4]
        ])

> optimize(eq2, evaln(Z_t));
                                        z[7]

> eval(Z_t);
  table([
        0 = 7
        1 = sin(x)
        2 = x z[1]
        3 = cos(y)
        4 = y z[3]
        5 = z[2] + z[4]
        6 = z w
        7 = z[2] + z[6]
        ])

> optimize(eq3, evaln(Z_t));
```

```
                                    z[9]

> eval(Z_t);
  table([
        0 = 9
        1 = sin(x)
        2 = x z[1]
        3 = cos(y)
        4 = y z[3]
        5 = z[2] + z[4]
        6 = z w
        7 = z[2] + z[6]
        8 = x y
        9 = z[4] + z[6] + z[8]
        ])
```

où Z_t est la table de variables temporaires, z[i] est la variable temporaire associée à l'élément i de la table de variables temporaires (Z_t[i]) et Z_t[0] contient le nombre d'élément dans la table.

3.1.2 Avantages de la nouvelle approche

La nouvelle approche proposée dans la section précédente comporte plusieurs avantages par rapport à l'approche utilisée dans la version 3.2 de SYMOFROS :

1. La table de variables temporaires ne contient aucun terme redondant car la procédure *optimize* utilise l'option *remember* de Maple et Maple exprime deux expressions équivalentes sous la même forme (voir section 2.1.1). Cette absence de redondance dans la table permet d'éviter d'avoir à resubstituer toutes les variables temporaires redondantes permettant ainsi une grande économie en temps et en mémoire lors de la génération symbolique des équations.
2. La nouvelle approche donne la possibilité à l'usager de regrouper les fonctions en groupes de fonctions selon ses besoins (voir section 3.4), ce qui n'était pas possible dans la version 3.2 car chaque table de variables

temporaires était associée à un groupe de fonctions préalablement établi dans SYMOFROS.

3. Puisque toutes les fonctions sont exprimées à l'aide des variables d'une seule table de variables temporaires et que cette table ne contient aucun terme redondant, l'information conservée en mémoire est minimale. Ce n'était pas le cas dans la version 3.2 car une table était associée à chacun des groupes et qu'une même expression pouvait donc se retrouver dans plusieurs tables, d'où une redondance d'information.

3.2 Réduction du nombre de variables temporaires

Les modifications apportées à la procédure *optimize* nécessitent certaines modifications dans SYMOFROS 4.0. En effet, puisque le post-traitement de *optimize*, qui consiste à remplacer les variables temporaires apparaissant à une seule reprise dans les expressions, a été enlevé de la nouvelle procédure d'optimisation, il est nécessaire d'ajouter une nouvelle procédure afin d'effectuer ce post-traitement.

La réduction de la table de variables temporaires consiste donc à remplacer les variables temporaires qui sont utilisées une seule fois par les autres variables ou par les équations des fonctions et d'enlever ces variables de la table. Cette réduction consiste aussi à enlever les variables de la table qui ne sont jamais utilisées et qui peuvent apparaître lorsque l'on multiplie une variable temporaire par 0. Voici un exemple qui illustre le procédé de réduction des variables temporaires :

Exemple 3.2 *Nous supposons que SYMOFROS contient seulement deux fonctions :* $\mathbf{M}$ *et* $\mathbf{g}$ *(voir 2.1) afin de simplifier l'exemple. Les valeurs sont issues d'un double pendule oscillant dans un plan XY. Les membres sont numérotés de 1 à 2 en partant de la base. Les masses sont représentées par* m_1 *et* m_2 *alors que les longueurs sont représentées par* l_1 *et* l_2*. La valeur* c_2 *représente la position du centre de masse par rapport à la base du deuxième membre.* $\mathbf{M}$ *et* $\mathbf{g}$ *sont exprimées à l'aide des variables temporaires et ont les valeurs suivantes :*

$$\mathbf{M} = \begin{bmatrix} I_2 & t[12] \\ t[12] & I_1 + t[17] \end{bmatrix}, \mathbf{g} = \begin{bmatrix} t[22] \\ t[28] \end{bmatrix}$$

où I_1 et I_2 représentent respectivement l'inertie du premier et du deuxième membre en Z. Voici les valeurs des variables temporaires :

$t[1]$	$=$	$\cos(X[1])$	$t[15]$	$=$	$t[3]^2$
$t[2]$	$=$	$\sin(X[1])$	$t[16]$	$=$	$t[15]m_2$
$t[3]$	$=$	$t[1]l_1$	$t[17]$	$=$	$I_2 + 2t[11] + t[14] + t[16]$
$t[4]$	$=$	$t[2]l_1$	$t[18]$	$=$	$t[7]l_1$
$t[5]$	$=$	$X[4] + X[3]$	$t[19]$	$=$	$t[8]l_1$
$t[6]$	$=$	$X[4]^2$	$t[20]$	$=$	$t[9]m_2$
$t[7]$	$=$	$t[6]t[1]$	$t[21]$	$=$	$t[20]c_2$
$t[8]$	$=$	$t[6]t[2]$	$t[22]$	$=$	$t[10]t[19]$
$t[9]$	$=$	$t[5]^2$	$t[23]$	$=$	$t[4]m_2$
$t[10]$	$=$	m_2c_2	$t[24]$	$=$	$t[23]t[18]$
$t[11]$	$=$	$t[3]t[10]$	$t[25]$	$=$	$t[4]t[21]$
$t[12]$	$=$	$I_2 + t[11]$	$t[26]$	$=$	$t[3]m_2$
$t[13]$	$=$	$t[4]^2$	$t[27]$	$=$	$t[26]t[19]$
$t[14]$	$=$	$t[13]m_2$	$t[28]$	$=$	$t[22] - t[24] - t[25] + t[27]$

La réduction consiste à remplacer les variables temporaires qui sont utilisées une seule fois par les fonctions ou par les autres variables temporaires de la table. Les variables $t[1], t[2], t[3], t[4], t[6], t[10], t[11], t[12], t[19], t[22]$ sont les seules variables temporaires qui sont utilisées plus d'une fois. Nous aurons donc un total de 10 variables temporaires qui seront notées par la variable $z[]$ pour éviter toute confusion. Après la réduction, nous obtenons donc les variables temporaires suivantes :

$z[1]$	$=$	$\cos(X[1])$	$z[6]$	$=$	$m_2 c_2$
$z[2]$	$=$	$\sin(X[1])$	$z[7]$	$=$	$z[3]z[6]$
$z[3]$	$=$	$z[1]l_1$	$z[8]$	$=$	$I_2 + z[7]$
$z[4]$	$=$	$z[2]l_1$	$z[9]$	$=$	$z[2]z[5]l_1$
$z[5]$	$=$	$X[4]^2$	$z[10]$	$=$	$z[6]z[9]$

ainsi que les nouvelles valeurs des fonctions :

$$\mathbf{M} = \begin{bmatrix} I_2 & z[8] \\ z[8] & I_1 + I_2 + 2z[7] + m_2(I_2 + z[7]) + m_2 z[3]^2 \end{bmatrix}$$

$$\mathbf{g} = \begin{bmatrix} z[10] \\ z[10] - m_2 l_1 z[1]z[4]z[5] - m_2 c_2 z[4](X[4] + X[3])^2 + m_2 z[3]z[9] \end{bmatrix}$$

La procédure de réduction permet donc de réduire de façon considérable la taille de la table globale de variables temporaires.

3.3 Linéarisation autour d'un point d'opération

Dans la version 3.2 de SYMOFROS, la linéarisation du modèle autour d'un point d'opération est très dispendieuse en mémoire et en temps. Nous présenterons une nouvelle méthode afin de dériver les équations reliées au modèle sans contraintes, aux contraintes intermittentes ainsi qu'aux contraintes permanentes. Nous montrerons aussi théoriquement l'efficacité de la nouvelle méthode par rapport à l'ancienne.

3.3.1 Dérivation de la table de variables temporaires

Dans la section 2.3.5, nous avons présenté le problème de la dérivation des variables comprises dans la table de variables temporaires. Nous avons aussi mentionné que l'approche utilisée dans la version 3.2 de SYMOFROS est lente. Nous présenterons maintenant une méthode qui permet de réduire de

façon considérable le temps nécessaire afin de dériver la table de variables temporaires.

Supposons que l'on désire calculer les dérivées des n variables temporaires $z[1], ..., z[n]$ par rapport à la variable x. Chaque $z[j]$ peut dépendre directement de x mais aussi des $z[i]$, $i = 1, ..., j-1$, qui peuvent à leur tour dépendre directement de x où des variables temporaires précédentes. La dérivation de $z[j]$ doit donc être effectuée en chaîne selon l'équation 2.5 qui peut être reformulée de la façon suivante :

$$D_x z[j] = \sum_{i=1}^{j-1} \frac{\partial z[j]}{\partial z[i]} D_x z[i] + \frac{\partial z[j]}{\partial x} \tag{3.1}$$

L'algorithme précédent était utilisé intégralement pour le calcul de chaque variable temporaire de la table dans la version 3.2 de SYMOFROS. Il était donc nécessaire d'effectuer j dérivées afin de calculer la valeur de la dérivée de $z[j]$. Donc la dérivée d'une table de j éléments demandait de calculer

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$$

dérivées. Le nombre de dérivées à effectuer est donc dans l'ordre de n^2.

La dérivation en chaîne peut se faire de façon plus efficace. En commençant la dérivation en chaîne par les premiers termes de la table, nous remarquons qu'une variable temporaire ne dépend pas nécessairement de toutes les variables temporaires précédentes de la table. La nouvelle procédure de dérivation de la table de variables temporaires tire profit de ce fait afin de réduire le nombre d'appels à la procédure de dérivation. La nouvelle méthode consiste en fait à dériver seulement par rapport aux variables temporaires apparaissant directement dans l'expression à dériver, les autres dérivées partielles étant nulles. Nous pouvons donc réécrire l'équation (3.1) de la façon suivante :

$$D_x z[j] = \sum_{i \in \omega_j} \frac{\partial z[j]}{\partial z[i]} D_x z[i] + \frac{\partial z[j]}{\partial x} \tag{3.2}$$

où

$$\omega_j = \{i \mid z[j] = f(z[i])\}$$

Dans, l'expression précédente, la fonction $f(z[i])$ indique que la variable $z[i]$ apparaît directement dans l'expression.

Si nous bornons par une constante c le nombre de variables temporaires apparaissant dans une variable temporaire quelconque (ie : la cardinalité de ω), le nombre d'appel à la procédure de dérivation à effectuer afin de dériver la table est linéaire plutôt que quadratique. En effet, suite à l'hypothèse précédente, il est possible de borner supérieurement le nombre de dérivation par :

$$\sum_{j=1}^{n} c = cn$$

L'hypothèse consistant à borner par une constante le nombre de variables temporaires apparaissant dans une expression de la table de variables temporaires est valide car lors de l'optimisation des équations, Maple divise en plusieurs variables une expression trop complexe. La valeur de c est de beaucoup inférieure à la valeur de n pour un modèle le moindrement complexe et ne dépend pas de la complexité du modèle.

3.3.2 Processus de linéarisation

La linéarisation du modèle autour d'un point d'opération consiste à calculer à partir de l'équation de la dynamique (équation 2.1) les valeurs suivantes pour chacun des modèles (modèle sans contrainte, modèle avec contraintes permanentes, modèle avec contraintes intermittentes) :

1. $\mathbf{K}_1 = \frac{\partial \mathbf{M}\ddot{\mathbf{q}}}{\partial \mathbf{q}}$: Matrice de rigidité associée à $\mathbf{M}$
2. $\mathbf{K}_2 = \frac{\partial \mathbf{g}}{\partial \mathbf{q}}$: Matrice de rigidité associée à $\mathbf{g}$
3. $\mathbf{K} = \mathbf{K}_1 + \mathbf{K}_2$: Matrice de rigidité
4. $\mathbf{U}_1 = \frac{\partial \mathbf{M}\ddot{\mathbf{q}}}{\partial \mathbf{u}}$: Matrice d'entrées associée à $\mathbf{M}$
5. $\mathbf{U}_2 = \frac{\partial \mathbf{g}}{\partial \mathbf{u}}$: Matrice d'entrées associée à $\mathbf{g}$
6. $\mathbf{U} = \mathbf{U}_1 + \mathbf{U}_2$: Matrice d'entrées

7. $\mathbf{B} = \frac{\partial \mathbf{g}}{\partial \dot{\mathbf{q}}}$: Matrice d'amortissement

Pour identifier à quel modèle correspond chacune des valeurs précédentes, nous ajouterons l'exposant PC dans le cas des contraintes permanentes et l'exposant IC dans le cas des contraintes intermittentes. La linéarisation du modèle autour d'un point d'opération dans SYMOFROS 4.0 consiste en plusieurs étapes :

1. Création d'une table (T1) contenant les variables temporaires apparaissant directement ou indirectement dans $\mathbf{g}$, $\mathbf{g}^{PC}$ ou $\mathbf{g}^{IC}$.
2. Dérivation de T1 par rapport à $\dot{\mathbf{q}}$.
3. Calcul de $\mathbf{B}, \mathbf{B}^{PC}, \mathbf{B}^{IC}$.
4. Création d'une table (T2) contenant les variables temporaires apparaissant directement ou indirectement dans $\mathbf{M}$, $\mathbf{M}^{PC}$ ou $\mathbf{M}^{IC}$ ou T1($\mathbf{g}$, $\mathbf{g}^{PC}$, $\mathbf{g}^{IC}$).
5. Dérivation de T2 par rapport à $\mathbf{q}$.
6. Calcul de $\mathbf{K}_1, \mathbf{K}_2, \mathbf{K}, \mathbf{K}_1^{PC}, \mathbf{K}_2^{PC}, \mathbf{K}^{PC}, \mathbf{K}_1^{IC}, \mathbf{K}_2^{IC}, \mathbf{K}^{IC}$.
7. Dérivation de T2 par rapport à $\mathbf{u}$.
8. Calcul de $\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}, \mathbf{U}_1^{PC}, \mathbf{U}_2^{PC}, \mathbf{U}^{PC}, \mathbf{U}_1^{IC}, \mathbf{U}_2^{IC}, \mathbf{U}^{IC}$.

Le principal avantage de la nouvelle approche par rapport à l'approche utilisée dans la version 3.2 est qu'aucune expression n'est dérivée plusieurs fois par rapport à la même variable. Dans SYMOFROS 3.2, chacun des modèles était dérivé à tour de rôle. Donc, une expression apparaissant dans plusieurs modèles serait dérivée par rapport à $\mathbf{u}$, $\mathbf{q}$ et $\dot{\mathbf{q}}$ pour chacun des modèles.

L'analyse des performances de la nouvelle procédure de linéarisation par rapport à la procédure de linéarisation de la version 3.2 de SYMOFROS est présentée dans la section 3.5.

3.4 Groupement des fonctions

Le groupement des fonctions consiste à rassembler les fonctions qui seront utilisées à un même taux d'échantillonnage dans la simulation. Les expressions sont mises en évidence dans un vecteur temporaire si elles apparaissent

dans au moins deux des fonctions du groupe et si elles sont indépendantes des accélérations et des entrées. Cette contrainte résulte du fait que les accélérations et les entrées peuvent être modifiées à l'intérieur d'un pas de calcul, ce qui n'est pas le cas des états (positions et vitesses) dans SYMOFROS. Lors de la simulation, chaque fonction appelle tout d'abord le vecteur temporaire. Chaque vecteur temporaire est appelé au plus une fois par pas de calcul. Cette approche permet d'éviter de refaire le même calcul dans plusieurs fonctions pendant le même pas de calcul.

Ce regroupement est effectué par l'usager après la génération symbolique des équations ce qui permettra à l'usager de modifier ces groupes de fonctions sans avoir à regénérer les équations symboliques du modèle. Cette approche est possible dans SYMOFROS 4.0 car lors de la génération symbolique des équations, toutes les fonctions sont exprimées à l'aide des variables de la même table de variables temporaires. Ceci n'était pas le cas dans SYMOFROS 3.2, d'où l'impossibilité de regrouper dynamiquement les fonctions après la génération symbolique.

Les interactions entre l'usager et SYMOFROS 4.0 lors du processus de génération sont présentées dans la figure 3.2.

3.5 Analyse des résultats

Dans cette section, nous avons présenté en détail les modifications apportées dans le processus de génération symbolique des équations, de groupement des fonctions ainsi que de l'écriture du code C. Afin d'analyser l'efficacité de la version 4.0 par rapport à la version 3.2, nous présenterons deux modèles de structures mécaniques construits dans SYMOFROS.

3.5.1 Modèles de tests

Le premier modèle (**Robot1**) qui sera utilisé afin de tester les modifications est une bielle-manivelle. Cette structure est formée d'un membre rigide et d'un membre flexible avec une masse à l'extrémité. Le mouvement de l'extrémité de la structure est limité à un plan. Ce modèle est donc une structure flexible avec une boucle cinématique fermée.

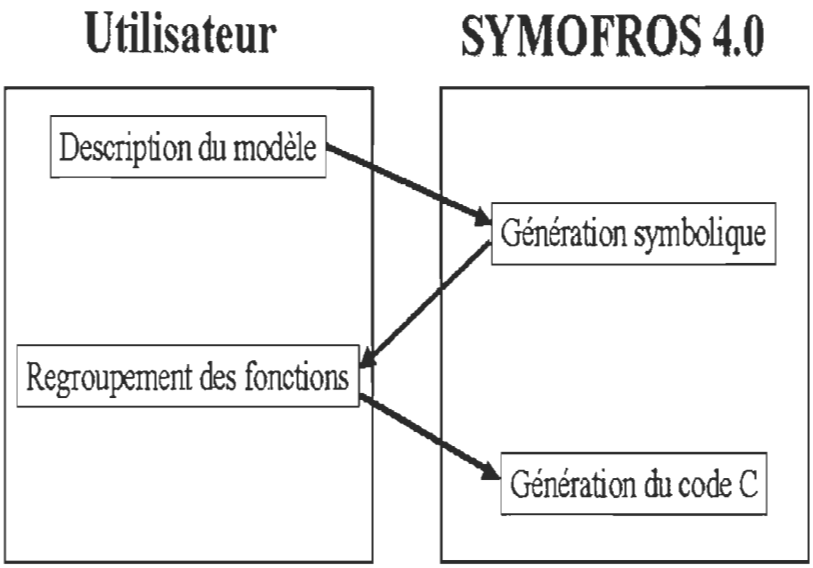

FIG. 3.2 – Interactions entre l'usager et SYMOFROS 4.0

Le deuxième modèle (**Robot2**-i) est une structure rigide de i membres ($2 \leq i \leq 6$) avec i degrés de liberté, soit un degré de liberté par membre. Cette structure permet d'étudier l'efficacité de la génération d'un modèle pour une structure rigide sans contrainte.

Tous les tests présentés dans ce document ont été effectués selon une méthodologie bien précise. Les tests ont été effectués à l'aide d'un processeur PentiumII de 400 MHz avec 128 M de mémoire vive fonctionnant sous Windows NT. Avant chacun des tests, l'ordinateur a été redémarré afin de s'assurer que Maple libère vraiment l'espace mémoire utilisé préalablement.

3.5.2 Réduction du nombre de variables temporaires

Le tableau 3.1 illustre l'importance de la réduction du nombre de variables temporaires dans la version 4.0 de SYMOFROS. Le tableau indique le nombre de variables temporaires avant et après la réduction :

On remarque que la réduction permet d'éliminer 80% des variables tempo-

Modèle	Avant réduction	Après réduction	Après / Avant (%)
Robot1	1 511	530	35.08 %
Robot2-2	2 386	526	22.05 %
Robot2-3	5 839	1 171	20.05 %
Robot2-4	9 815	1 969	20.06 %
Robot2-5	15 192	3 030	19.94 %
Robot2-6	25 889	5 262	20.33 %

TAB. 3.1 – Nombre de variables temporaires avant et après la réduction dans SYMOFROS 4.0

raires.

3.5.3 Temps de génération

Nous présentons maintenant une comparaison entre les temps nécessaires afin de générer le modèle symbolique dans SYMOFROS 3.2 et dans SYMOFROS 4.0. Le tableau 3.2 ainsi que la figure 3.3 permettent de comparer le temps nécessaire (en secondes) afin de générer symboliquement les équations dans SYMOFROS 3.2 et SYMOFROS 4.0. Cette comparaison est incomplète car elle ne compare pas des procédures qui effectuent exactement les mêmes tâches et ne permet donc pas d'isoler l'effet des modifications apportées par l'auteur. Cependant, ces comparaisons permettront de voir les points forts et les points faibles de la nouvelle version de SYMOFROS.

Nous remarquons que le temps requis afin de linéariser le modèle autour d'un point d'opération a été réduit de plus de la moitié comme l'illustre le tableau 3.3 ainsi que la figure 3.4.

Cependant, la réduction du nombre de variables temporaires dans SYMOFROS 4.0 est plus dispendieuse en temps que dans la version 3.2 comme l'illustre le tableau 3.4 et la figure 3.5. Ceci est une conséquence du fait que la nouvelle procédure d'optimisation crée plus de variables temporaires et donc la réduction est plus longue.

Modèle	Version 3.2	Version 4.0	v.4.0 / v3.2 (%)
Robot1	46.9 s	30.9 s	65.88 %
Robot2-2	31.7 s	25.6 s	80.76 %
Robot2-3	62.1 s	51.5 s	82.93 %
Robot2-4	107.6 s	86.2 s	80.11 %
Robot2-5	191.4 s	152.6 s	79.73 %
Robot2-6	390.1 s	363.6 s	93.21 %

TAB. 3.2 – Temps requis pour générer symboliquement les équations

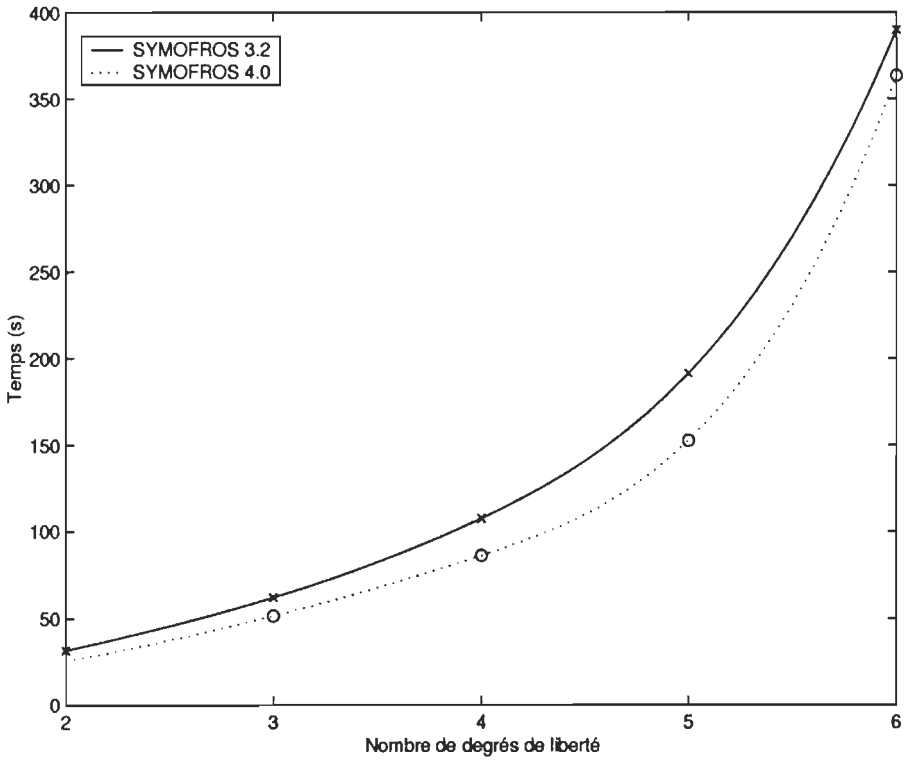

FIG. 3.3 – Temps de la génération symbolique pour **Robot2-i**

Modèle	Version 3.2	Version 4.0	v.4.0 / v3.2 (%)
Robot1	12.1 s	2.1 s	17.36 %
Robot2-2	14.2 s	5.0 s	35.21 %
Robot2-3	33.6 s	14.3 s	42.56 %
Robot2-4	60.0 s	26.4 s	44.00 %
Robot2-5	106.9 s	46.5 s	43.50 %
Robot2-6	210.1 s	99.0 s	47.12 %

TAB. 3.3 – Temps requis pour linéariser le modèle autour d'un point d'opération

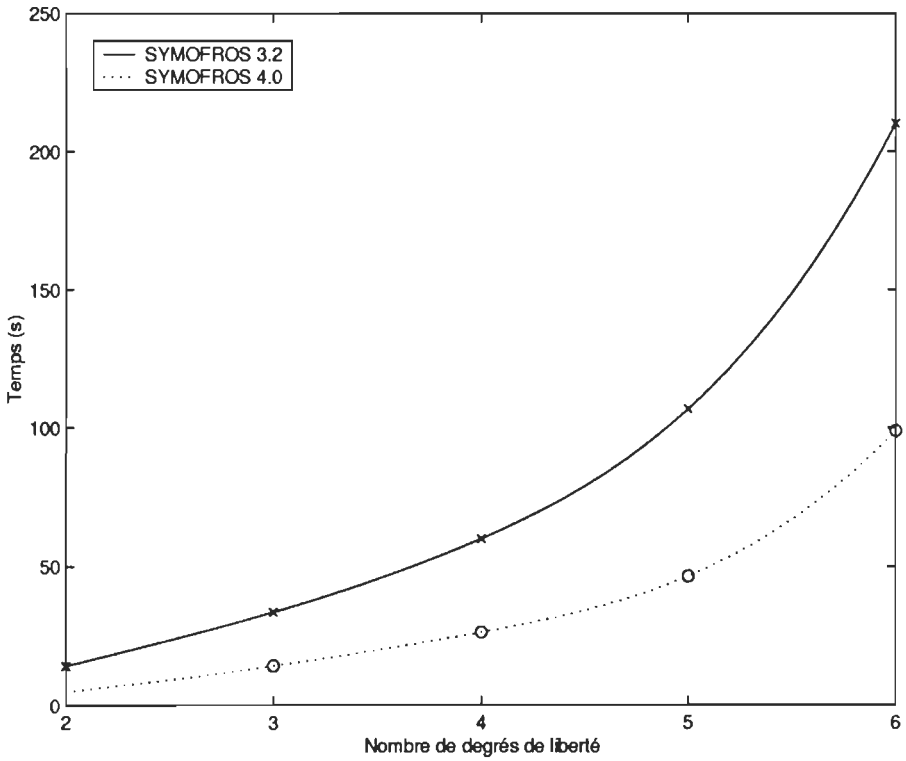

FIG. 3.4 – Temps nécessaire pour linéariser le modèle autour d'un point d'opération pour **Robot2-i**

Modèle	Version 3.2	Version 4.0	v.4.0 / v3.2 (%)
Robot1	4.0 s	4.7 s	117.50 %
Robot2-2	2.4 s	4.6 s	191.67 %
Robot2-3	9.0 s	16.3 s	181.11 %
Robot2-4	23.5 s	34.1 s	145.11 %
Robot2-5	55.7 s	74.3 s	133.40 %
Robot2-6	145.6 s	225.4 s	154.81 %

TAB. 3.4 – Temps requis pour la réduction du nombre de variables temporaires

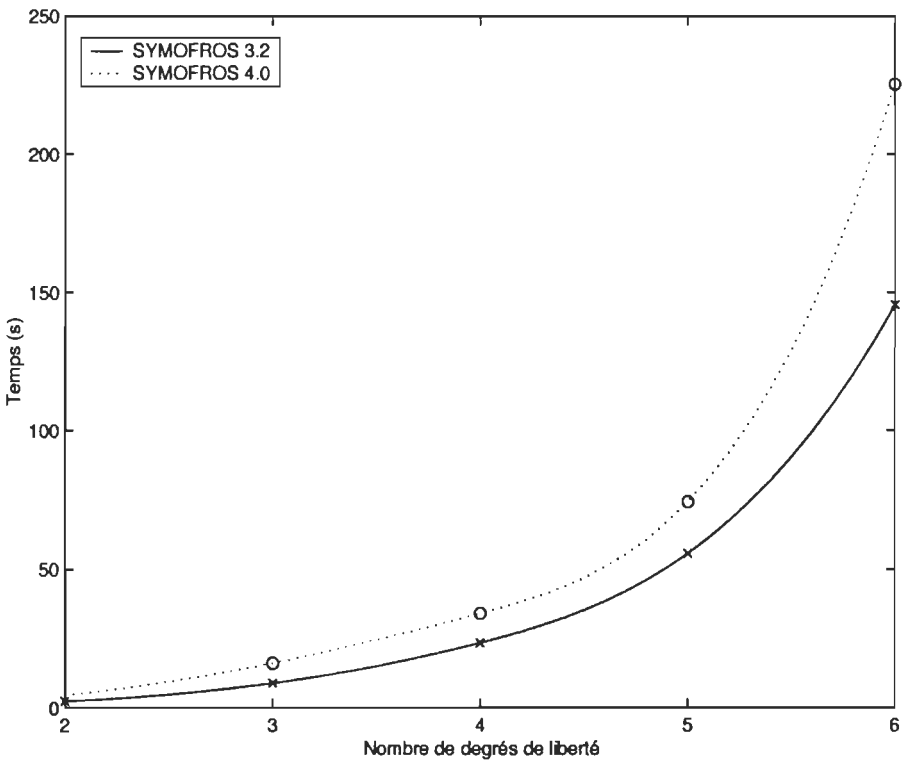

FIG. 3.5 – Temps nécessaire à la réduction des variables temporaires pour **Robot2-i**

Remarquons finalement que dans le cas du modèle **Robot2-6**, il y a saturation de mémoire ce qui explique pourquoi les résultats ne semblent pas suivre la tendance.

3.5.4 Mémoire requise

Il est très difficile de comparer la quantité de mémoire utilisée par les deux versions de SYMOFROS pour plusieurs raisons :

1. Les étapes de manipulation des expressions dans SYMOFROS 3.2 sont très différentes de celles de la nouvelle version, il est très difficile de comparer certaines étapes du processus.
2. Il y a 42 fonctions disponibles dans SYMOFROS 3.2 alors qu'il y en a 56 dans SYMOFROS 4.0.
3. Dans la version 4.0, il y a plus de variables créées pendant la génération des équations que dans la version précédente ce qui influence grandement la quantité de mémoire utilisée par la suite.
4. Plusieurs structures de données sont créées, utilisées et effacées pendant une étape du processus de génération ce qui rend difficile l'étude de la quantité de mémoire utilisée. Par exemple, à quel moment de la génération devons-nous prendre les mesures de la quantié de mémoire utilisée ?

Pour ces raisons, nous ne présenterons pas de comparaison de la quantité de mémoire utilisée par les deux versions de SYMOFROS. Mentionnons cependant que pour le modèle **Robot2-7**, il n'est pas possible de générer le modèle (avec l'ordinateur dont les caractéristiques ont été mentionnées précédemment) avec SYMOFROS 3.2 sans obtenir un problème de manque de mémoire alors qu'il est possible de générer ce modèle dans SYMOFROS 4.0.

3.5.5 Flexibilité et efficacité du code généré

Le regroupement dynamique des fonctions permet de réduire le nombre d'opérations nécessaires lors de la simulation en évitant les calculs redondants entre plusieurs fonctions. Il n'est cependant pas possible d'effectuer une comparaison entre la version 3.2 de SYMOFROS et la version 4.0 car le regroupement dépend de l'utilisation que fait l'usager de SYMOFROS 4.0.

3.6 Conclusion

Dans cette section, nous avons proposé une nouvelle approche afin d'optimiser les équations générées par SYMOFROS. Cette approche est basée sur une version modifiée de la procédure d'optimisation de Maple permettant d'optimiser les expressions à mesure qu'elles sont créées en plus de permettre un regroupement dynamique des fonctions par l'usager. Ce groupement peut donc être adapté au modèle et aux besoins de l'usager de façon à améliorer l'efficacité de la simulation. De plus, le regroupement des fonctions peut être modifié et il est possible de générer le fichier C associé au modèle sans devoir regénérer symboliquement les équations.

Nous avons aussi proposé une nouvelle méthode afin de dériver des expressions exprimées de façon récursive, c'est-à-dire à l'aide de variables temporaires, lors de la linéarisation du modèle autour d'un point d'opération. Cette nouvelle méthode permet de réduire le temps nécessaire à la génération des équations du modèle linéaire de façon significative.

Chapitre 4

Estimation des paramètres dynamiques

Les modifications proposées dans la section précédente ont permis d'améliorer certains aspects associés à la modélisation de structures mécaniques dans SYMOFROS. Pour une structure mécanique définie, l'usager obtient un ensemble d'expressions décrivant différents aspects du comportement de la structure dans un état et dans un environnement donné. Ces expressions dépendent entre autre de certaines caractéristiques physiques de chacun des membres de la structure qui sont difficiles à évaluer expérimentalement, comme l'inertie ou la position du centre de masse. Ces paramètres doivent être estimés afin de pouvoir effectuer une simulation. Dans la section 2.4, nous avons présenté les principales étapes ainsi qu'un bref résumé des méthodes proposées dans la littérature pour effectuer l'identification des paramètres dynamiques. Ces méthodes sont très spécifiques, c'est-à-dire qu'elles dépendent grandement de la topologie de la structure. Dans le cadre de ce travail, nous proposerons une méthode générale afin d'estimer les paramètres dynamiques d'une structure mécanique à l'aide des expressions générées par SYMOFROS.

Dans ce chapitre, nous présenterons une méthode afin de trouver un ensemble minimal de paramètres pour une structure formée de membres rigides et flexibles sans boucle cinématique fermée. Pour ce faire, nous proposerons tout d'abord un ensemble de paramètres dynamiques apparaissant de façon linéaire dans les équations de la dynamique afin de caractériser chaque membre rigide et flexible d'une structure mécanique. À partir de ces paramètres, nous effectuerons certains regroupements afin d'obtenir un ensemble de

paramètres identifiables et indépendants (ensemble de base). Cet ensemble de base nous sera ensuite utile afin d'estimer les valeurs de ces paramètres en supposant les trajectoires excitantes connues.

4.1 Choix des paramètres dynamiques

Un choix judicieux des paramètres dynamiques est une étape essentielle du processus d'estimation des paramètres. Nous proposons maintenant un ensemble de paramètres dynamiques linéaires pour les membres rigides et les membres flexibles.

4.1.1 Structure rigide

Les paramètres dynamiques choisis afin de représenter chacun des membres rigides de la structure sont :

Masse	m
Moments du premier ordre	r_x, r_y, r_z
Moments du second ordre (inertie)	$I_{xx}, I_{xy}, I_{xz}, I_{yy}, I_{yz}, I_{zz}$
Inertie du rotor	I_r

Les moments du premier ordre pour un membre donné sont les produits entre la masse et la position du centre de masse par rapport au référentiel attaché à la base du membre. Les moments du second ordre représentent les inerties du membre mesurées par rapport au référentiel de la base.

4.1.2 Structure flexible

Afin d'effectuer un choix adéquat des paramètres dynamiques pour une membrure flexible, nous devons tout d'abord effectuer certaines hypothèses afin de modéliser la structure. Dans notre cas, nous utiliserons les hypothèses présentées par Piedboeuf [Pie98] et utilisées dans SYMOFROS.

Les paramètres dynamiques à estimer pour une membrure flexible respectant les hypothèses précédentes peuvent être divisés en deux grandes catégories : les paramètres inertiels et les paramètres de rigidité. Les paramètres inertiels à estimer sont la densité linéaire (ρ) ainsi que les inerties par unité de longueur d'une section ($dI_{xx}, dI_{yy}, dI_{zz}$) mesurées au centroïde d'une section

de la poutre alors que les 6 paramètres associés à la rigidité de la membrure sont : la rigidité en flexion selon y (EI_y) et z (EI_z), la rigidité en torsion (GI_t) ainsi que les produits entre le coefficient d'amortissement associé au modèle de Voigt-Kelvin [PPN98] et les rigidités en flexion (κ_y, κ_z) et en torsion (κ_t).

L'ensemble de paramètres dynamiques contient donc les 10 paramètres suivants :

$$\rho, dI_{xx}, dI_{yy}, dI_{zz}, EI_y, EI_z, GI_t, \kappa_y, \kappa_z, \kappa_t$$

4.2 Ensemble de base

4.2.1 Formulation du problème

Soit $\delta_1, ..., \delta_p$, les p paramètres dynamiques d'une structure représentée par n coordonnées généralisées apparaissant de façon linéaire dans les équations de la dynamique (équation 2.1), alors il est possible d'écrire :

$$\boldsymbol{\phi}\ \boldsymbol{\delta} = \mathbf{Z} \tag{4.1}$$

où

$$\boldsymbol{\delta} = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_p \end{bmatrix}, \quad \boldsymbol{\phi} = \begin{bmatrix} \phi_{11} & \cdots & \phi_{1p} \\ \vdots & \ddots & \vdots \\ \phi_{n1} & \cdots & \phi_{np} \end{bmatrix}, \quad \mathbf{Z} = \begin{bmatrix} Z_1 \\ \vdots \\ Z_n \end{bmatrix}$$

Notons que la matrice $\boldsymbol{\phi}$ et le vecteur $\mathbf{Z}$ sont indépendants des paramètres $\delta_i \in \boldsymbol{\delta}$. La matrice $\boldsymbol{\phi}$ est appelée matrice de régression et peut être obtenue en calculant le jacobien de $\mathbf{M\ddot{q}} + \mathbf{g}$ par rapport à $\boldsymbol{\delta}$:

$$\boldsymbol{\phi} = \begin{bmatrix} \frac{\partial \mathbf{M\ddot{q}}+\mathbf{g}}{\partial \delta_1} & \cdots & \frac{\partial \mathbf{M\ddot{q}}+\mathbf{g}}{\partial \delta_p} \end{bmatrix}$$

La matrice de régression $\boldsymbol{\phi}$ est donc composée de n lignes et p colonnes. Le vecteur $\mathbf{Z}$ est obtenu en remplaçant les paramètres dynamiques par 0 dans les équations de la dynamique :

$$\mathbf{Z} = -\left(\mathbf{M\ddot{q}} + \mathbf{g}\right)|_{\delta_1 = ... = \delta_p = 0}$$

Par conséquent, $\mathbf{Z}$ est un vecteur de n lignes.

4.2.2 Élimination des paramètres non-identifiables

Il est ensuite facile d'éliminer les paramètres $\delta_j \in \boldsymbol{\delta}$ qui n'apparaissent pas dans les équations de la dynamique puisque ces paramètres correspondent aux colonnes j de la matrice $\boldsymbol{\phi}$ qui sont nulles. Ces paramètres ne sont pas identifiables et il est alors possible de réduire la taille de $\boldsymbol{\phi}$ et de $\boldsymbol{\delta}$ en enlevant les paramètres qui ne sont pas identifiables de $\boldsymbol{\delta}$ et en enlevant les colonnes nulles de la matrice $\boldsymbol{\phi}$. Le nombre de paramètres dynamiques à estimer passe donc de p à p', où $p' \leq p$ et l'équation 4.1 peut être reformulée sous la forme suivante :

$$\boldsymbol{\phi}' \, \boldsymbol{\delta}' = \mathbf{Z} \tag{4.2}$$

4.2.3 Dépendance entre les paramètres identifiables

La dépendance entre certains paramètres implique que chacun des paramètres ne peut pas être estimé indépendamment des autres paramètres. Par exemple, soit L une constante connue et soit :

$$\boldsymbol{\phi\delta} = \begin{bmatrix} \sin q_1 & L \sin q_1 \\ q_2 & Lq_2 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}$$

Il est possible de réécrire les équations précédentes sous la forme :

$$\sin(q_1)\,(\delta_1 + L\delta_2) = Z_1$$
$$q_2\,(\delta_1 + L\delta_2) = Z_2$$

Notons que quelque soit la trajectoire (ie : les valeurs de $\sin(q_1)$, q_2, Z_1 et Z_2), il est impossible de déterminer les valeurs de δ_1 et de δ_2 individuellement. Il est cependant possible d'évaluer la valeur de $\delta_1 + L\delta_2$.

Soit $\boldsymbol{\phi}_j$, la colonne j de $\boldsymbol{\phi}$ et $\mathbf{x}$, l'ensemble des états $(\mathbf{q}, \dot{\mathbf{q}})$, des accélérations $(\ddot{\mathbf{q}})$ et des entrées $(\mathbf{u})$. La méthode proposée afin de trouver les paramètres dépendants consiste à trouver les dépendances entre les colonnes de la matrice de régression $\boldsymbol{\phi}$ associée. Avant de présenter la méthode, nous devons tout

d'abord définir clairement en quoi consiste la dépendance entre les colonnes de $\boldsymbol{\phi}$.

Définition 1 *La colonne $\boldsymbol{\phi}_i$ est dépendante des colonnes $\boldsymbol{\phi}_j$ pour $j = 1, ..i-1$ si et seulement s'il existe des constantes $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_{i-1})$ tel que*

$$\boldsymbol{\phi}_i = \sum_{j=1}^{i-1} \alpha_j \boldsymbol{\phi}_j$$

Donc, il suffit de trouver les valeurs des éléments de $\boldsymbol{\alpha}$ s'ils existent. Notons cependant que $\boldsymbol{\phi}$ dépend de façon non-linéaire des variables $x_i \in \mathbf{x}$, c'est-à-dire que $\boldsymbol{\phi}$ s'exprime comme une combinaison des x_i, de leur puissance (x_i^k) ainsi que des fonctions trigonométrique $\sin(x_i)$ et $\cos(x_i)$.

Afin de trouver les valeurs des $\boldsymbol{\alpha}$, nous pouvons utiliser la procédure *solve* de Maple. Cependant, nous devons préalablement séparer l'expression en plusieurs sous-expressions car nous désirons que l'équation soit valable pour toutes les valeurs de $\mathbf{x}$. Voici un exemple qui permet de clarifier ce point.

Exemple 4.1 *Soit les constantes c_1, c_2, c_3, c_4 et x_1, x_2, deux variables. Soit*

$$\boldsymbol{\phi\delta} = \begin{bmatrix} c_1 x_1 & c_2 x_2 & c_3 x_1 + c_4 x_2 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} = \begin{bmatrix} Z_1 \end{bmatrix}$$

Il est évident que les deux premières colonnes de la matrice de régression $\boldsymbol{\phi}$ sont indépendantes car elles ne dépendent pas des mêmes variables. Le paramètres δ_3 est identifiable seulement en combinaisons linéaires si et seulement s'il existe (α_1, α_2) tel que

$$\alpha_1 \boldsymbol{\phi}_1 + \alpha_2 \boldsymbol{\phi}_2 = \alpha_1 c_1 x_1 + \alpha_2 c_2 x_2 = c_3 x_1 + c_4 x_2 = \boldsymbol{\phi}_3$$

Si nous utilisons la procédure solve de Maple afin de trouver la valeur de α_1 et de α_2, on obtient

expr := $\alpha_1 * c_1 * x_1 + \alpha_2 * c_2 * x_2 - c_3 * x_1 - c_4 * x_2$:
α := $\{\alpha_1, \alpha_2\}$:
solve({*expr*}, *alpha*);

$$\left\{\alpha_1 = \alpha_1, \alpha_2 = -\frac{\alpha_1 * c_1 * x_1 - c_3 * x_1 - c_4 * x_2}{c_2 * x_2}\right\}$$

Puisque le système consiste en une seule équation et deux inconnues, Maple isole tout simplement α_2 et l'exprime en fonction de α_1. Nous devons donc séparer l'équation précédente en plusieurs sous-équations. À chaque variable, nous associons une équation en utilisant le coefficient de cette variable à l'aide de la procédure coeff de Maple. Dans notre cas, nous cherchons les coefficients des variables x_1 et x_2. À l'aide des deux équations obtenues, il est alors possible de trouver les valeurs de α_1 et α_2 :

expr1 := coeff(expr, x_1);

$$expr1 = \alpha_1 * c_1 - c_3$$

expr2 := coeff(expr, x_2);

$$expr2 := \alpha_2 * c_2 - c_4$$

solve({expr1, expr2}, alpha);

$$\left\{\alpha_1 = \frac{c_3}{c_1}, \alpha_2 = \frac{c_4}{c_2}\right\}$$

Nous savons donc que la colonne 3 de la matrice de régression s'exprime comme une combinaison linéaire des colonnes 1 et 2. Par conséquent, les 3 paramètres δ_1, δ_2 et δ_3 ne peuvent pas tous être évalués individuellement.

Soit $\mathbf{y} = \{x_j \cup x_j^k \cup \sin^k(x_j) \cup \cos^k(x_j) \mid x_j \in \mathbf{x}, k \in N, k \neq 0\}$. De façon plus formelle, la méthode consiste à créer une équation pour chacun des termes apparaissant dans y et dans l'équation en plus de considérer le terme constant. Illustrons cette méthode avec un exemple théorique un peu plus complexe qui n'est cependant pas basé sur des équations obtenues d'un modèle réel.

Exemple 4.2 *Soit a, b, c_1 et c_2, des constantes et soit la matrice de régression suivante :*

$$\phi = \begin{bmatrix} x_1\sin(x_1) + x_2 & x_2^2\cos(x_1) + c_1 & ax_1\sin(x_1) + bx_2^2\cos(x_1) + ax_2 + c_2 \\ 0 & 0 & 0 \end{bmatrix}$$

Nous avons que $\mathbf{y} = \{x_1, \sin(x_1), \cos(x_1), x_2, x_2^2\}$. *Afin de déterminer si les colonnes sont dépendantes entre elles, nous cherchons les valeurs des constantes* α_1 *et* α_2 *tel que* $\alpha_1\phi_1 + \alpha_2\phi_2 = \phi_3$. *Nous obtenons donc l'ensemble d'équations suivantes :*

$$\left\{\begin{array}{rcl} x_1 & : & \alpha_1 \sin(x_1) - a\sin(x_1) = 0 \\ \sin(x_1) & : & \alpha_1 x_1 - a x_1 = 0 \\ \cos(x_1) & : & \alpha_2 x_2^2 - b x_2^2 = 0 \\ x_2 & : & \alpha_1 - a = 0 \\ x_2^2 & : & \alpha_2 \cos(x_1) - b\cos(x_1) = 0 \\ \textit{termes constants} & : & \alpha_1 c_1 - c_2 = 0 \end{array}\right.$$

Dans ce cas, il existe une solution qui est

$$\left\{\begin{array}{rcl} \alpha_1 & = & a \\ \alpha_2 & = & b \end{array}\right\}$$

avec la condition que $a = \frac{c_2}{c_1}$.

L'exemple précédent, qui est un exemple purement théorique montre que certaines contraintes sur les constantes peuvent apparaître lorsque l'on cherche à trouver les valeurs de α_1 et de α_2. Si on utilise la commande *solve* de Maple sur les équations précédentes, Maple ne trouve pas de solution. Dans ce cas, puisque nous utilisons la procédure *solve* de Maple et que nous n'avons aucun moyen de déterminer les relations entre les constantes dans SYMOFROS, nous considérerons le paramètre comme indépendant des paramètres précédents. Dans le cas où une telle contrainte serait respectée dans un modèle particulier et que cette contrainte permettrait d'établir une dépendance entre certains paramètres, l'ensemble de base comporterait certains paramètres dépendants, et donc l'ensemble de base généré par le logiciel ne serait pas réellement un ensemble de base. Cependant, les résultats obtenus lors de l'estimation à l'aide de l'ensemble qui n'est pas minimal seraient quand même corrects sauf que la taille du problème sera plus grande car on aura une redondance d'information issue de la dépendance entre les paramètres.

4.2.4 Regroupement des paramètres identifiables seulement en combinaisons linéaires

Dans la section 4.2.2, nous avons montré comment réduire la taille des équations en éliminant les paramètres qui n'apparaissent pas dans les équations de la dynamique. L'étape suivante consiste à regrouper les paramètres qui sont seulement identifiables en terme de combinaisons linéaires (par rapport à $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ et $\mathbf{u}$). Cette dépendance entre les paramètres dynamiques peut être déterminée en étudiant les dépendances entre les colonnes de la matrice de régression. Il est possible de réécrire l'équation 4.2 sous la forme suivante :

$$\left[\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_i, ..., \boldsymbol{\phi}_p\right] \begin{bmatrix} \delta_1 \\ \delta_2 \\ \cdots \\ \delta_i \\ \cdots \\ \delta_p \end{bmatrix} = \mathbf{Z} \tag{4.3}$$

ou sous la forme

$$\sum_{j=1}^{p} \boldsymbol{\phi}_j \delta_j = \sum_{j=1}^{i-1} \boldsymbol{\phi}_j \delta_j + \boldsymbol{\phi}_i \delta_i + \sum_{j=i+1}^{p} \boldsymbol{\phi}_j \delta_j = \mathbf{Z} \tag{4.4}$$

Afin de simplifier la notation, nous avons remplacé $\boldsymbol{\phi}'$ par $\boldsymbol{\phi}$ et $\boldsymbol{\delta}'$ par $\boldsymbol{\delta}$ malgré le fait que nous supposons que tous les paramètres qui n'apparaissent pas dans les équations de la dynamique ont été enlevés. Pour chaque colonne $\boldsymbol{\phi}_i$ de la matrice de régression, il est possible de vérifier si cette colonne est dépendante des colonnes précédentes. D'après la définition présentée préalablement, nous pouvons dire que la colonne $\boldsymbol{\phi}_i$ est dépendante des colonnes précédentes si et seulement s'il existe un vecteur de constantes α_j, pour $j = 1, ..., i-1$, tel que

$$\boldsymbol{\phi}_i = \sum_{j=1}^{i-1} \alpha_j \boldsymbol{\phi}_j \tag{4.5}$$

La condition $[\alpha_1, ..., \alpha_{i-1}] \neq [0, ..., 0]$ n'est pas nécessaire puisque les colonnes nulles de la matrice ont été enlevées de la matrice de régression.

La dépendance entre les colonnes est vérifiée à l'aide des variables dans $\mathbf{x}$ $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{u})$. Il est donc possible que pour certaines valeurs numériques de $\mathbf{x}$, certaines colonnes indépendantes deviennent dépendantes ou même nulles. Cependant, ce cas n'est pas considéré pour le moment et pourra seulement être considéré lors de la détermination des trajectoires excitantes. En supposant que la colonne $\boldsymbol{\phi}_i$ s'exprime comme une combinaison linéaire des colonnes précédentes et en remplaçant l'équation (4.5) dans l'équation (4.4), on obtient que :

$$\begin{aligned}
\mathbf{Z} &= \sum_{j=1}^{i-1} \boldsymbol{\phi}_j \delta_j + (\sum_{j=1}^{i-1} \alpha_j \boldsymbol{\phi}_j)\delta_i + \sum_{j=i+1}^{p} \boldsymbol{\phi}_j \delta_j \\
&= \sum_{j=1}^{i-1} \left(\boldsymbol{\phi}_j \delta_j + \alpha_j \boldsymbol{\phi}_j \delta_i\right) + \sum_{j=i+1}^{p} \boldsymbol{\phi}_j \delta_j \\
&= \sum_{j=1}^{i-1} \boldsymbol{\phi}_j \left(\delta_j + \alpha_j \delta_i\right) + \sum_{j=i+1}^{p} \boldsymbol{\phi}_j \delta_j
\end{aligned}$$

En exprimant la dernière équation sous forme vectorielle, on obtient :

$$\left[\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_{i-1}, \boldsymbol{\phi}_{i+1}, ..., \boldsymbol{\phi}_p\right] \begin{bmatrix} \delta_1 + \alpha_1 \delta_i \\ \delta_2 + \alpha_2 \delta_i \\ \cdots \\ \delta_{i-1} + \alpha_{i-1} \delta_i \\ \delta_{i+1} \\ \cdots \\ \delta_p \end{bmatrix} = \mathbf{Z} \tag{4.6}$$

L'équation précédente montre qu'il est très simple de regrouper les paramètres qui sont dépendants. Ce regroupement est appliqué pour chacune des colonnes de $\boldsymbol{\phi}$, de $\boldsymbol{\phi}_2$ jusqu'à $\boldsymbol{\phi}_p$, qui sont dépendantes des colonnes précédentes. L'algorithme peut être amélioré en exprimant une colonne $\boldsymbol{\phi}_i$ dépendante comme la combinaison linéaire des colonnes précédentes qui sont indépendantes entre elles. De façon plus formelle, l'algorithme peut être formulé de la façon suivante :

$$\boldsymbol{\phi}' = [\boldsymbol{\phi}_1]$$

$\boldsymbol{\delta}' = [\delta_1]$
Pour $i = 2$ à p
  $k = \dim(\boldsymbol{\delta}')$
  Si $\exists(\alpha_1, ..., \alpha_k)$ tel que $\boldsymbol{\phi}_i = \sum_{j=1}^{k} \alpha_j \boldsymbol{\phi}'_j$

$$\boldsymbol{\delta}' = \boldsymbol{\delta}' + \begin{bmatrix} \alpha_1 \delta_i \\ ... \\ \alpha_k \delta_i \end{bmatrix}$$

  Sinon

$$\boldsymbol{\phi}' = [\boldsymbol{\phi}' \ \boldsymbol{\phi}_i]$$

$$\boldsymbol{\delta}' = \begin{bmatrix} \boldsymbol{\delta}' \\ \delta_i \end{bmatrix}$$

  Fin si
Fin pour

Le vecteur $\boldsymbol{\delta}'$ obtenu représente alors un ensemble de base de nouveaux paramètres dynamiques qui sont des regroupements des paramètres utilisés initialement, $\boldsymbol{\phi}'$ représente la matrice de régression associée à ces nouveaux paramètres dynamiques alors que le vecteur $\mathbf{Z}$ reste inchangé. Le logiciel joint dans l'annexe B et permettant de trouver un ensemble minimal est basé sur cet algorithme. Ce logiciel permet de traiter le problème uniquement dans le cas où les équations ne sont pas exprimées de façon récursive.

4.2.5 Exemple de génération d'un ensemble de base

Nous présentons maintenant un exemple qui illustre comment le logiciel génère un ensemble minimal de paramètres. Nous montrons de plus que le résultat est exact, c'est-à-dire que l'ensemble de base est valable et minimal.

Exemple 4.3 *La structure considérée est une structure série composée de trois membres rigides numérotés de 1 à 3, en partant de la base. La longueur de ces membres sont l_1, l_2 et l_3. Les angles associés à chacune des articulations rotatives situés à la base de chacun des membres sont respectivement notés par θ_1, θ_2 et θ_3. Afin de simplifier la notation, s_i représentera* $\sin(\theta_i)$ *alors que c_i représentera* $\cos(\theta_i)$.

Dans le modèle initial, 5 paramètres dynamiques doivent être estimés pour chacun des membres i de la structure : la masse ($m^{(i)}$), le premier moment

d'inertie en x ($I_x^{(i)}$) ainsi que les trois second moments d'inertie principaux ($I_{xx}^{(i)}$, $I_{yy}^{(i)}$, $I_{zz}^{(i)}$). Nous montrerons que l'application développée permet de trouver l'ensemble minimal de paramètres.

La première étape consiste à transformer les équations de la dynamique afin d'isoler les 15 paramètres dynamiques. Il est alors possible d'éliminer 7 paramètres qui n'apparaissent pas dans les équations de la dynamique. Ces paramètres non-identifiables sont : $m^{(1)}$, $I_{xx}^{(1)}$, $I_{yy}^{(1)}$, $I_{xx}^{(2)}$, $I_{yy}^{(2)}$, $I_{xx}^{(3)}$, $I_{yy}^{(3)}$. Les seconds moments d'inertie $I_{xx}^{(i)}$, $I_{yy}^{(i)}$ n'apparaissent pas dans les équations de la dynamique car les membres effectuent des rotations autour de l'axe des z.

Nous pouvons donc écrire l'équation sous la forme :

$$\boldsymbol{\phi}\boldsymbol{\delta} = \mathbf{Z}$$

avec

$$\boldsymbol{\delta} = \left[I_x^{(1)}, I_{zz}^{(1)}, m^{(2)}, I_x^{(2)}, I_{zz}^{(2)}, m^{(3)}, I_x^{(3)}, I_{zz}^{(3)}\right]^t$$

$$\mathbf{Z} = \begin{bmatrix} \tau_1 - b_1\dot{\theta}_1 \\ \tau_2 - b_2\dot{\theta}_2 \\ \tau_3 - b_3\dot{\theta}_3 \end{bmatrix}$$

$$\boldsymbol{\phi} = \begin{bmatrix} s_1 g & \ddot{\theta}_1 & {l_1}^2\ddot{\theta}_1 + l_1 g s_1 & \phi_{14} & \ddot{\theta}_1 + \ddot{\theta}_2 & \phi_{16} & \phi_{17} & \ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3 \\ 0 & 0 & 0 & \phi_{24} & \ddot{\theta}_1 + \ddot{\theta}_2 & \phi_{26} & \phi_{27} & \ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & \phi_{37} & \ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3 \end{bmatrix}$$

où

$$\begin{aligned}
\phi_{14} &= g c_1 s_2 - l_1 s_2 \dot{\theta}_2^{\,2} + l_1 c_2 \ddot{\theta}_2 + 2 l_1 c_2 \ddot{\theta}_1 - 2 l_1 s_2 \dot{\theta}_1 \dot{\theta}_2 + g s_1 c_2 \\
\phi_{24} &= l_1 c_2 \ddot{\theta}_1 + l_1 s_2 \dot{\theta}_1^{\,2} + g c_1 s_2 + g s_1 c_2 \\
\phi_{16} &= l_1 l_2 c_2 \ddot{\theta}_2 + l_2^2 \ddot{\theta}_2 + 2 l_1 l_2 c_2 \ddot{\theta}_1 + l_2^2 \ddot{\theta}_1 + l_1^2 \ddot{\theta}_1 + l_2 g c_1 s_2 \\
&\quad - l_1 l_2 s_2 \dot{\theta}_2^{\,2} + l_2 g s_1 c_2 - 2 l_1 l_2 s_2 \dot{\theta}_1 \dot{\theta}_2 + l_1 g s_1 \\
\phi_{26} &= l_1 l_2 c_2 \ddot{\theta}_1 + l_2^2 \ddot{\theta}_1 + l_2^2 \ddot{\theta}_2 + l_1 l_2 s_2 \dot{\theta}_1^{\,2} + l_2 g s_1 c_2 + l_2 g c_1 s_2 \\
\phi_{17} &= -l_2 s_3 \dot{\theta}_3^{\,2} - 2 l_1 c_3 s_2 \dot{\theta}_2 \dot{\theta}_3 - 2 l_1 s_3 c_2 \dot{\theta}_1 \dot{\theta}_3 - 2 l_1 s_3 c_2 \dot{\theta}_1 \dot{\theta}_2
\end{aligned}$$

$$
\begin{aligned}
& -2l_1 s_3 c_2 \dot{\theta}_2 \dot{\theta}_3 - 2l_1 c_3 s_2 \dot{\theta}_1 \dot{\theta}_3 - 2l_1 c_3 s_2 \dot{\theta}_1 \dot{\theta}_2 - 2l_2 s_3 \dot{\theta}_2 \dot{\theta}_3 \\
& -2l_2 s_3 \dot{\theta}_1 \dot{\theta}_3 - l_1 s_3 c_2 \dot{\theta}_3^{\,2} - l_1 s_3 c_2 \dot{\theta}_2^{\,2} + 2l_2 c_3 \ddot{\theta}_2 + l_2 c_3 \ddot{\theta}_3 \\
& +g c_3 c_1 s_2 + g c_3 s_1 c_2 - l_1 c_3 s_2 \dot{\theta}_3^{\,2} + g s_3 c_1 c_2 - g s_3 s_1 s_2 + 2l_2 c_3 \ddot{\theta}_1 \\
& -l_1 s_3 s_2 \ddot{\theta}_2 + l_1 c_3 c_2 \ddot{\theta}_3 \\
& -l_1 s_3 s_2 \ddot{\theta}_3 + 2l_1 c_3 c_2 \ddot{\theta}_1 - 2l_1 s_3 s_2 \ddot{\theta}_1 + l_1 c_3 c_2 \ddot{\theta}_2 - l_1 c_3 s_2 \dot{\theta}_2^{\,2} \\
\phi_{27} \;=\; & -l_1 s_3 s_2 \ddot{\theta}_1 + l_1 c_3 c_2 \ddot{\theta}_1 + 2l_2 c_3 \ddot{\theta}_1 + 2l_2 c_3 \ddot{\theta}_2 + l_2 c_3 \ddot{\theta}_3 \\
& +l_1 c_3 s_2 \dot{\theta}_1^{\,2} + l_1 s_3 c_2 \dot{\theta}_1^{\,2} - 2l_2 s_3 \dot{\theta}_1 \dot{\theta}_3 - 2l_2 s_3 \dot{\theta}_2 \dot{\theta}_3 - l_2 s_3 \dot{\theta}_3^{\,2} \\
& +g s_3 c_1 c_2 + g c_3 s_1 c_2 + g c_3 c_1 s_2 - g s_3 s_1 s_2 \\
\phi_{37} \;=\; & -l_1 s_3 s_2 \ddot{\theta}_1 + l_1 c_3 c_2 \ddot{\theta}_1 + l_2 c_3 \ddot{\theta}_1 + l_2 c_3 \ddot{\theta}_2 + l_1 s_3 c_2 \dot{\theta}_1^{\,2} \\
& +l_2 s_3 \dot{\theta}_1^{\,2} + 2l_2 s_3 \dot{\theta}_1 \dot{\theta}_2 + l_2 s_3 \dot{\theta}_2^{\,2} + l_1 c_3 s_2 \dot{\theta}_1^{\,2} + g s_3 c_1 c_2 \\
& -g s_3 s_1 s_2 + g c_3 c_1 s_2 + g c_3 s_1 c_2
\end{aligned}
$$

L'application permet ensuite de déterminer symboliquement la dépendance entre les colonnes de la matrice de régression ϕ. Nous obtenons que la colonne 3 et la colonne 6 peuvent s'exprimer comme une combinaison linéaire des colonnes précédentes et peuvent s'écrire de la façon suivante :

$$\phi_3 = l_1 \phi_1 + l_1^2 \phi_2 \tag{4.7}$$

$$\phi_6 = l_1 \phi_1 + l_1^2 \phi_2 + l_2 \phi_4 + l_2^2 \phi_5 \tag{4.8}$$

Les autres colonnes de la matrice ϕ sont indépendantes entre elles. Illustrons maintenant comment s'effectue le regroupement.

La première étape consiste à initialiser ϕ' et δ' :

$$\phi' = [\phi_1], \delta' = [\delta_1]$$

Puisque la colonne 2 de la matrice de régression est indépendante de la colonne 1, nous modifions ϕ' et δ' :

$$\phi' = [\phi_1 \ \phi_2], \delta' = \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix}$$

Puisque la colonne ϕ_3 est dépendante des deux premières colonnes, nous obtenons alors :

$$\boldsymbol{\delta}' = \begin{bmatrix} \delta_1 + l_1\delta_3 \\ \delta_2 + l_1^2\delta_3 \end{bmatrix}$$

alors que la valeur de $\boldsymbol{\phi}'$ demeure inchangée. En appliquant le même procédé pour les autres colonnes de $\boldsymbol{\phi}$ on obtient l'ensemble de base suivant :

$$\boldsymbol{\delta}' = \begin{bmatrix} \delta_1 + l_1(\delta_3 + \delta_6) \\ \delta_2 + l_1^2(\delta_3 + \delta_6) \\ \delta_4 + l_2\delta_6 \\ \delta_5 + l_2^2\delta_6 \\ \delta_7 \\ \delta_8 \end{bmatrix} = \begin{bmatrix} I_x^{(1)} + l_1\left(m^{(2)} + m^{(3)}\right) \\ I_{zz}^{(1)} + l_1^2\left(m^{(2)} + m^{(3)}\right) \\ I_x^{(2)} + l_2 m^{(3)} \\ I_{zz}^{(2)} + l_2^2 m^{(3)} \\ I_x^{(3)} \\ I_{zz}^{(3)} \end{bmatrix}$$

et la nouvelle matrice de régression suivante :

$$\boldsymbol{\phi}' = [\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_4\ \boldsymbol{\phi}_5\ \boldsymbol{\phi}_7\ \boldsymbol{\phi}_8]$$

Remarquons que les paramètres de l'ensemble de base obtenus dans l'exemple précédent sont en fait les paramètres barycentriques tel que définis par Raucent [Rau90].

4.3 Estimation des paramètres

L'ensemble minimal de paramètres obtenu dans la section 4.2 contient des paramètres qui sont tous identifiables individuellement. L'étape suivante consiste à trouver une trajectoire excitante et à estimer les valeurs des paramètres expérimentalement. Dans cette section, nous nous intéressons seulement au problème de l'estimation des paramètres pour une trajectoire donnée. La méthode utilisée est illustrée par la figure 4.1.

Exemple 4.4 *Afin de tester la méthode des moindres carrés, nous utiliserons les données de l'exemple 7. Dans ce dernier exemple, nous avons obtenu un ensemble de base contenant 6 paramètres identifiables et indépendants.*

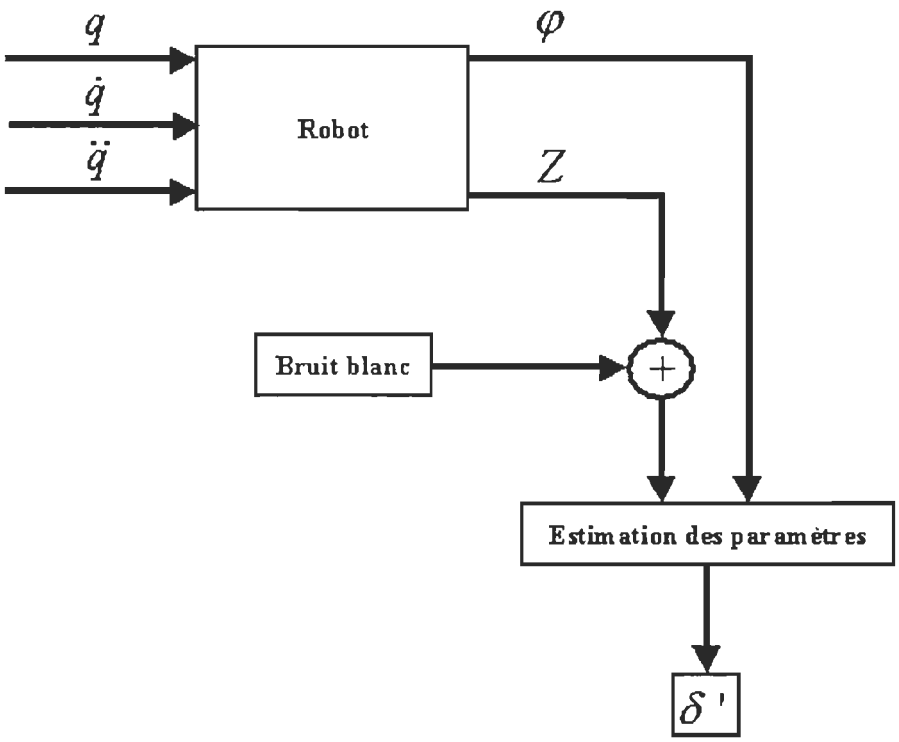

FIG. 4.1 – Estimation des paramètres dynamiques

Afin d'éviter toute confusion, nous renommerons les paramètres de la façon suivante :

Nom	*Paramètre*	*Valeur*
p_1	$I_x^{(1)} + l_1\left(m^{(2)} + m^{(3)}\right)$	*9.30*
p_2	$I_{zz}^{(1)} + l_1^2\left(m^{(2)} + m^{(3)}\right)$	*1.71*
p_3	$I_x^{(2)} + l_2 m^{(3)}$	*4.00*
p_4	$I_{zz}^{(2)} + l_2^2 m^{(3)}$	*0.31*
p_5	$I_x^{(3)}$	*0.30*
p_6	$I_{zz}^{(3)}$	*0.02*

Les trajectoires utilisées sont :

$$q_i = \frac{\pi}{4}\sin(\omega t), i = 1, 2, 3$$

Notons que la fréquence ω de la trajectoire est la même pour chacune des coordonnées généralisées. Le mouvement de la structure est alors périodique de période $2\pi\omega^{-1}$ secondes et la rotation de chacune des articulations est limitée entre -45 degré et 45 degrés ($-\frac{\pi}{4}$ et $\frac{\pi}{4}$ radians). À l'aide de cette trajectoire simple, vérifions les résultats obtenus à l'aide de la méthode des moindres carrés dans Matlab (à l'aide de la commande mldivide).

Posons $\omega = 1.0$ rad/s, nous obtenons alors que la période de la trajectoire est de 2π secondes. Nous échantillonnerons le signal à toutes les 0.25 seconde. Nous supposerons que le bruit sur la trajectoire $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ est négligeable par rapport aux bruits sur les mesures des éléments de $\mathbf{Z}$, c'est-à-dire sur les mesures des moments de force appliqués à la structure par le contrôleur afin que la structure suive la trajectoire désirée. Cette hypothèse est notamment utilisée par Swevers et al [SGT+97b].

Le tableau 4.1 illustre les résultats obtenus en considérant un bruit blanc d'écart-type 0.01 sur chacun des éléments de $\mathbf{Z}$ pour chaque échantillon. L'échantillonnage se fait pour 10 cycles, soit pendant 62.8s (20π) secondes. On refait cette expérience à 30 reprises et on calcule la moyenne et l'écart-type obtenus pour les paramètres.

	$\bar{p}_1$	$\bar{p}_2$	$\bar{p}_3$	$\bar{p}_4$	$\bar{p}_5$	$\bar{p}_6$
Valeur exacte	9.3000	1.7100	4.0000	0.3100	0.3000	0.0200
Moyenne	9.3004	1.7150	4.0000	0.3104	0.3000	0.0198
Écart-type	0.0080	0.0929	0.0012	0.0107	0.0002	0.0015

TAB. 4.1 – Résultats de l'estimation des paramètres pour un bruit blanc avec un écart-type 0.01

L'intensité du bruit a évidemment une influence directe sur la précision de l'estimation. Le tableau 4.2 illustre la moyenne des valeurs obtenues pour du bruit blanc de différentes intensités alors que le tableau 4.3 illustre l'écart-type de ces valeurs.

Les résultats obtenus sont acceptables pour des bruits de faible intensité. Nous remarquons cependant que tous les paramètres ne peuvent pas être mesurés avec autant de précision. Par exemple, l'écart-type obtenu pour le paramètre p_2 *(*σ_{p_2}*) est beaucoup plus élevé que l'écart-type obtenu pour les autres paramètres.*

4.4 Conclusion

Nous avons présenté une méthode automatique afin de trouver un ensemble de base de paramètres dynamiques pour une structure mécanique rigide ou flexible sans boucle cinématique dans le cas où les expressions ne sont pas exprimées de façon récursive. Cette méthode tire profit du calcul symbolique dans Maple ainsi que des équations disponibles dans SYMOFROS. La procédure présentée est générale et peut être utilisée à partir des équations de la dynamique ou à partir des équations associées au modèle énergétique.

Nous avons utilisé la méthode des moindres carrés afin d'estimer les valeurs numériques des paramètres de l'ensemble de base. Nous avons présenté les résultats obtenus pour une structure donnée en simulant les trajectoires et en ajoutant du bruit sur les mesures des moments de force.

Écart-type du bruit blanc	$\bar{p}_1$	$\bar{p}_2$	$\bar{p}_3$	$\bar{p}_4$	$\bar{p}_5$	$\bar{p}_6$
0.00	9.3000	1.7100	4.0000	0.3100	0.3000	0.0200
0.01	9.3004	1.7150	4.0000	0.3104	0.3000	0.0198
0.02	9.2990	1.6978	4.0000	0.3105	0.3001	0.0201
0.03	9.3027	1.7426	4.0005	0.3143	0.2998	0.0186
0.04	9.2975	1.6791	3.9999	0.3098	0.3000	0.0204
0.05	9.3086	1.8113	4.0002	0.3114	0.3001	0.0205

TAB. 4.2 – Moyenne des valeurs des paramètres estimés pour différents niveaux de bruit blanc

Écart-type du bruit blanc	σ_{p_1}	σ_{p_2}	σ_{p_3}	σ_{p_4}	σ_{p_5}	σ_{p_6}
0.01	0.0080	0.0929	0.0012	0.0107	0.0002	0.0015
0.02	0.0126	0.1472	0.0018	0.0167	0.0005	0.0030
0.03	0.0199	0.2313	0.0028	0.0242	0.0006	0.0036
0.04	0.0269	0.3115	0.0042	0.0376	0.0011	0.0063
0.05	0.0297	0.3428	0.0048	0.0433	0.0010	0.0059

TAB. 4.3 – Écart-type des valeurs des paramètres estimés pour différents niveaux de bruit blanc

Chapitre 5

Conclusion

Les travaux présentés dans ce document portent sur différents aspects reliés à la modélisation et à la simulation de structures mécaniques. Nous avons proposé des approches qui sont globales et qui pourraient être appliquées dans des domaines différents de la robotique. Deux sujets principaux ont été traités, soit l'amélioration de la génération symbolique du code lors du processus de modélisation ainsi que l'estimation des paramètres dynamiques.

Une nouvelle approche a été proposée afin de manipuler l'information lors de la génération symbolique des équations dans SYMOFROS. L'optimisation des expressions est maintenant effectuée lors de la génération des expressions plutôt qu'après. La nouvelle procédure d'optimisation donne aussi une plus grande flexibilité à l'utilisateur de SYMOFROS puisqu'il peut regrouper les fonctions selon ses besoins permettant ainsi d'améliorer l'efficacité de la simulation. Une nouvelle méthode afin de dériver une expression récursive a aussi été proposée. Cette approche permet d'améliorer la vitesse de la linéarisation du modèle autour d'un point d'opération, qui est l'une des tâches les plus longues du processus de génération symbolique dans SYMOFROS. Cependant, le processus de génération symbolique des équations demande une quantité importante de mémoire qui pourrait être réduite considérablement en tirant profit de la structure afin de réduire le nombre de variables temporaires.

Nous avons aussi présenté une méthode générale afin d'obtenir un ensemble minimal de paramètres dynamiques identifiables et indépendants. Cette méthode a été automatisée et est applicable pour une structure arborescente

rigide ou flexible quelconque lorsque les équations ne sont pas exprimées de façon récursive. Une procédure permettant l'estimation des paramètres pour une trajectoire donnée a aussi été développée. Cependant, pour un modèle le moindrement complexe, les équations doivent être exprimées de façon récursive et dans ce cas, la méthode proposée dans ce document n'est pas applicable.

Certains aspects n'ont pas été traités et pourront faire l'objet de futurs travaux :

1. La réduction du nombre de variables temporaires dans la table est très coûteuse en temps lors de la génération des équations. Ceci est entre autre une conséquence immédiate du fait que le nombre de variables temporaires dans la table globale lors de la génération des équations est énorme. Ce nombre de variables pourraient être réduit considérablement en créant des variables temporaires pour des expressions spécifiques en tenant compte de la structure des équations associée à une structure mécanique particulière dans SYMOFROS.
2. La génération automatique d'un ensemble de base pour une structure avec boucles cinématiques fermées.
3. La génération d'un ensemble minimal lorsque les équations sont exprimées récursivement. Ceci permettrait de traiter des problèmes plus complexes.

Le regroupement des fonctions pourrait aussi servir afin d'effectuer des simulations en parallèle en assignant un processeur à chacun des groupes de fonctions.

Bibliographie

[Ang97] J. Angeles. *Fundamentals of Robotic Mechanical Systems.* Springer-Verlag, 1997.

[Arm89] B. Armstrong. On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics. *The International Journal of Robotics Research*, 8(6) :28–48, 1989.

[Bla92] W. Blajer. A projection method approach to constrained dynamic analysis. *Journal of Applied Mechanics*, 59 :643–649, 1992.

[BN86] F. Bauer et J.A. Nohel. *Introduction to Differential Equations with Applications.* Harper et Row, Publishers, Inc., 1986.

[CDGP93] P. Chedmail, P. Dépincé, M. Gautier, et C.M. Pham. Identification of minimum set parameters of flexible robots. Dans *IMAC - Symposium on Mathematical et intelligence models in system simulation*, pages 75–78, 1993.

[Cra89] J.J. Craig. *Introduction to Robotics : Mechanics et Control.* Addison Wesley Publishing Company, Inc., 2e édition, 1989.

[DCB93] P. Dépincé, P. Chedmail, et F. Bennis. Minimum parameters of a flexible links robots - application to a one flexible link. Dans *1993 ICAR Tokyo Proc.C.*, 1993.

[dJB94] J. García de Jalón et E. Bayo. *Kinematic et Dynamic Simulation of Multibody Systems.* Springer-Verlag, 1994.

[EBO96] H. Elmqvist, D. Brck, et M. Otter. *Dymola — User's Manual.* Dynasim AB, Research Park Ideon, Lund, Sweden, 1996.

[Fer96] G. Ferretti. Systematic dynamic modelling of mechanical systems containing kinematic loops. *Mathematical Modelling of Systems*, 2(3) :212–235, 1996.

[FRS95] P. Fisette, B. Raucent, et J.C. Samin. Contribution to the identification of dynamic parameters of robot manipulators with closed-loop. Dans *ICIAM'95*, page 175, July 3-7 1995.

[FRS96] P. Fisette, B. Raucent, et J.C. Samin. Minimal dynamic characterization of tree-like multibody systems. *Nonlinear Dynamics*, 9 :165–184, 1996.

[GK88] W. Gautier et W. Khalil. A direct determination of minimum inertial parameters of robots. Dans *Proc. IEEE Conf. on Robotics et Automation*, pages 1682–1687, April 24-29 1988.

[GK90] W. Gautier et W. Khalil. Direct calculation of minimum set of inertial parameters of serial robots. *IEEE Transactions on Robotics et Automation*, 6(3) :368–372, 1990.

[GK92] W. Gautier et W. Khalil. Exciting trajectories for the identification of base inertial parameters of robots. *The International Journal of Robotics Research*, 11(4) :362–375, 1992.

[GKR95] W. Gautier, W. Khalil, et P.P. Restrepo. Identification of the dynamic parameters of a closed loop robot. Dans *Proc. IEEE Conf. on Robotics et Automation*, pages 3045–3050, 1995.

[GL96] G.H. Golub et C.F Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[Jou09] P.E.B. Jourdain. Note on an analogue of gauss' principle of least constraint. *The Quaterly Journal of Pure et Applied Mathematics*, XL :153–157, 1909.

[Kan61] T.R. Kane. Dynamics of nonholonomic systems. *Transactions of the ASME, Journal of Applied Mechanics*, pages 574–578, 1961.

[Kan83] T.R. Kane. Formulation of dynamical equations of motion. *American Journal of Physics*, pages 974–977, 1983.

[Khe96] N.A. Kheir. *Systems Modeling et Computer Simulation*. Marcel Dekker, Inc., 1996.

[KL83] T.R. Kane et D.A. Levinson. Multibody dynamics. *Transactions of the ASME, Journal of Applied Mechanics*, 50 :1071–1078, 1983.

[LFRS91] K. Lipinski, P. Fisette, B. Raucent, et J.-C. Samin. External identification of dynamic parameters of robot manipulators with a parallelogram mechanism. Dans *6th International Symposium*

on Measurement et Control in Robotics, pages 1169–1174, April 1991.

[Mar96] J.-P. Martineau. *Modélisation expérimentale des réducteurs Harmonic Drive, application à la détermination des paramètres minimaux d'un robot souple trois axes.* PhD thesis, École Centrale de Nantes, 1996.

[Mat97] The MathWorks Inc. *Using SIMULINK*, 1997.

[MDP00] E. Martin, M. Doyon, et J.-C. Piedboeuf. Hardware-in-the-loop simulation using symofros. Dans *ISR 2000, Proceedings of the 31st International Symposium on Robotics*, pages 474–480, 2000.

[MGH+96] M.B. Monagan, K.O. Geddes, K.M. Heal, G. Labahn, et S.M. Vorkoetter. *Maple V, Programming Guide.* Springer, 1996.

[MSW89a] P. Maes, J.C. Samin, et P.Y. Willems. Linearity of multibody systems with respect to barycentric parameters : Dynamics et identification models obtained by symbolic generation. *Mechanics of Structures et Machines*, pages 219–237, 1989.

[MSW89b] P. Maes, J.C. Samin, et P.Y. Willems. *Multibody System Handbook*, pages 246–264. W. Schiehlen (ed)., Springer-Verlag, 1989.

[OC97] M. Oliviers et G. Campion. A contribution to identification of robots with flexible arms. Dans *Proc. SYSID'97, 11th IFAC Symposium on System Identification*, pages 159–164, 1997.

[PDLL99] J.-C. Piedboeuf, M. Doyon, P. Langlois, et R. L'Archevêque. Symofros : A flexible dynamics modeling software. Dans *ISAIRAS 99*, 1999.

[Pie93] J.-C. Piedboeuf. Kane's equations or Jourdain's principle ? Dans *36th Midwest Symposium on Circuits et Systems*, Août 1993.

[Pie95a] J.-C. Piedboeuf. The jacobian matrix for a flexible manipulator. *Journal of Robotic Research*, 12(11) :709–726, 1995.

[Pie95b] J.-C. Piedboeuf. Symbolic modelling of flexible manipulators. *AAS/AIAA Astrodynamics Specialist Conference*, Août 1995.

[Pie96] J.-C. Piedboeuf. Modelling flexible robots with Maple. *Maple Tech : The Maple Technical Newsletter*, 3(1) :38–47, 1996.

[Pie98] J.-C. Piedboeuf. Recursive modeling of serial flexible manipulators. *The Journal of the Astronautical Sciences*, 46(1) :1–24, 1998.

[PPN98] M.-J. Potvin, J.-C. Piedboeuf, et J.A. Nemes. Comparison of viscoelastic models in simulating the transient response of a slewing polymer arm. *Transactions of the ASME, Journal of Dynamic Systems, Measurement, et Control*, 120 :340–345, 1998.

[Rau90] B. Raucent. *Identification des paramètres dynamiques des robots manipulateurs.* PhD thesis, Louvain-La-Neuve, 1990.

[RCB+91] B. Raucent, G. Campion, G. Bastin, J.-C. Samin, et P.-Y. Willems. On the identification of the barycentric parameters of robot manipulators from external measurements. Dans *Proc. IEEE Conf. on Robotics et Automation*, pages 1169–1174, April 1991.

[Ric98] J. Richalet. *Pratique de l'identification.* Hermes, 1998.

[RSG91] B. Raucent, J.-C. Samin, et R. Gorez. Influence of inertial parameters on the positional accuracy of a robot. Dans *ICAR'91*, pages 946–951, June 20-22 1991.

[Sco88] D. Scott. Can a projection method of obtaining equations of motion compete with Lagrange's equations ? *American Journal of Physics*, 5 :451–456, 1988.

[SGT+97a] J. Swevers, C. Ganseman, D.B. Tukel, J. De Schutter, et H. Van Brussel. Additional remarks related to the paper : Optimal robot excitation and identification. Technical report, Division Production Engineering, MAchine Design and Automation (PMA), Katholieke Universiteit Leuven, 1997.

[SGT+97b] J. Swevers, C. Ganseman, D.B. Tukel, J. De Schutter, et H. Van Brussel. Optimal robot excitation et identification. *IEEE Transactions on Robotics et Automation*, 13(5) :730–739, 1997.

ANNEXE A

Optimisation

ENTRÉES :

1. Expression à optimiser
2. Nom de la table de variables temporaires

SORTIES :

1. Structure de données (identique à l'expression passée en paramètres) contenant les expressions exprimées à l'aide des variables temporaires de la table.

UTILISATION :

1. Ouvrir Maple 5.1 et taper la commande *readlib(optimize);*
2. Charger la procédure d'optimisation dans Maple
3. Initialiser la table de variables temporaires *TableVarTemp* à l'aide de la commande *init_optimize(TableTempVar)*
4. Optimiser une expression *expr* à l'aide de la table de variables temporaires *TableVarTemp* : *optimize(expr, evaln(TableTempVar))*

NOTES

1. L'optimisation peut se faire à l'aide de plusieurs tables de variables temporaires de façon simultanée.
2. La procédure d'optimisation n'assigne pas de variables temporaires pour une expression de la forme : cst*var (où cst est une valeur numérique).
3. Les variables temporaires sont exprimées sous la forme z[i] plutôt que ti.
4. La table de variables temporaires ne comprend pas d'expressions redondantes (ceci est un résultat de l'utilisation de l'*option remember* de Maple)

```
# ------------------------------------------------------------ #
# 'Auteur: Brian Moore
#
# 'Nom: init_optimize
#
# 'Date: 12/1999
#
# 'Description:
#    Initialisation du nombre d'elements dans la table de
#    variables temporaires (Z_t[0]) et initialisation des
#    tables remember des procedures de optimize.
#
# 'Entrees
#    Nom de la table de variable temporaire
#
# 'Sorties
#    Aucune
#
# ------------------------------------------------------------ #


init_optimize := proc(a::name)

  a[0] := 0;
  `optimize/clear`(`optimize/prepro`);
  `optimize/clear`(`optimize/ass`);
  `optimize/clear`(`optimize/makeass`);
  `optimize/clear`(`optimize/postpro`);

 end:
```

```
# ------------------------------------------------------------ #
# 'Auteur: Brian Moore
#
# 'Nom: optimize
#
# 'Date: 12/1999
#
# 'Description:
#    Procedure optimize creee a partir de la commande optimize
#    de Maple.  Cette procedure permet d'optimiser des
#    expressions pendant la generation d'equations.
#    La procedure peut etre appele a plusieurs reprises
#    par rapport a une table de variables temporaires Z_t
#    declaree comme globale.
#
# 'Entrees
#    1) Expression algebrique a optimiser
#    2) Nom de la table de variables temporaires
#
# 'Sorties
#    1) Expression exprimee a l'aide des variables temporaires
#       de la table mentionnee comme Entree 2.
#
# ------------------------------------------------------------ #

optimize := proc(a, TableTempVar)
local s, opt, mod_expr;

  if nargs < 2 then
    ERROR(`There are less than 2 arguments in the call of optimize`)
  fi;

  if nargs > 2 then
    ERROR(`There are more than two arguments in the call of optimize`)
  fi;

  if not assigned(TableTempVar[0]) then
```

```
    ERROR('Element 0 of the table of temporary variable
                    must be initialized (default = 0)');
  fi;

  if type(a, {name, numeric}) then RETURN(a)
  elif not type(a, algebraic) then ERROR('invalid arguments', a)
  fi;

  mod_expr := factor(simplify(expand(a)));

  userinfo(1, optimize, 'optimizing an algebraic expression');
  s := 'optimize/prepro'(mod_expr);

  userinfo(2, optimize, 'finding common subexpressions');
  s := 'optimize/makeass'(s, TableTempVar);
end:
```

```
# ------------------------------------------------------------ #
# 'Auteur: Brian Moore
#
# 'Nom: `optimize/ass`
#
# 'Date: 12/1999
#
# 'Description:
#    Procedure optimize creee a partir de la commande
#    de Maple.  Assignation d'une variable temporaire a une
#    expression passee en parametre et mise a jour de la
#    table de variables temporaires.
#
# 'Entrees
#    1) Expression algebrique a ajouter dans la table
#    2) Nom de la table de variables temporaires
#
# 'Sorties
#    1) Retourne le nom de la variable temporaire associee
#       a l'expression.
#
# 'Notes
#     L'option remember permet d'eviter d'ajouter une
#     variable dans la table de variables temporaires si
#     elle en fait deja partie.
#
# ------------------------------------------------------------ #

`optimize/ass` := proc(expr, ass)
  local n, x;
  option remember;

  n := ass[0] + 1;
  ass[0] := n;
  x := z[n];
  ass[n] := simplify(expr);
  x;
end:
```

```
# ------------------------------------------------------------ #
# 'Auteur: Brian Moore
#
# 'Nom: `optimize/makeass`
#
# 'Date: 12/1999
#
# 'Description:
#    Procedure optimize creee a partir de la commande
#    de Maple.
#
# 'Entrees
#    1) Expression algebrique a ajouter dans la table
#    2) Nom de la table de variables temporaires
#
# 'Sorties
#
# 'Notes
#
# ------------------------------------------------------------ #

`optimize/makeass` := proc(expr, ass)
local t1, t2, i, base, pow, quo, rem;
option remember,
`Copyright (c) 1992 by the University of Waterloo. All rights
              reserved.`;

  if type(expr, {numeric, name}) then expr
  elif type(expr, `*`) then
    t1 := op(1, expr);
    if type(t1, numeric) then
      t1*procname(subsop(1 = 1, expr), ass);
    elif t1 = I then t1*procname(subsop(1 = 1, expr), ass)
    elif 3 < nops(expr) then
      rem := nops(expr);
      quo := iquo(rem, 2);
      t1 := procname(convert([op(1 .. quo, expr)], `*`), ass);
      t2 := procname(convert([op(quo + 1 .. rem, expr)], `*`),
```

```
                    ass);
      `optimize/ass`(t1*t2, ass)
    elif nops(expr) = 3 then
      t1 := procname(op(1, expr), ass);
      t2 := procname(op(2, expr), ass);
      t1 := `optimize/ass`(t1*t2, ass);
      t2 := procname(op(3, expr), ass);
      `optimize/ass`(t1*t2, ass)
    elif nops(expr) = 2 then
      t1 := procname(op(1, expr), ass);
      t2 := procname(op(2, expr), ass);
      `optimize/ass`(t1*t2, ass)
    fi
  elif type(expr, anything^integer) then
    base := op(1, expr);
    pow := op(2, expr);
    if pow = 2 then
      base := procname(base, ass);
      quo := `optimize/ass`(base^2, ass);
      quo
    elif pow < 0 then
      quo := procname(1/expr, ass);
      `optimize/ass`(1/quo, ass)
    else
      rem := `optimize/bins`(pow);
      quo := pow - rem;
      quo := procname(base^quo, ass);
      rem := procname(base^rem, ass);
      procname(rem*quo, ass)
    fi
  elif expr = I then expr
  elif type(expr, anything^fraction) then
    base := op(1, expr);
    pow := op(2, expr);
    if op(1, pow) = 1 then
      base := procname(base, ass);
      `optimize/ass`(base^pow, ass)
    elif 1 < pow then
```

```
    rem := procname(base^trunc(pow), ass);
    base := procname(base^frac(pow), ass);
    'optimize/ass'(base*rem, ass)
  elif pow < 0 then
    base := procname(base^(-pow), ass);
    'optimize/ass'(1/base, ass)
  else
    base := procname(base^(1/op(2, pow)), ass);
    procname(base^op(1, pow), ass)
  fi
elif type(expr, '+') then
  t1 := op(4, op(procname));
  if t1 <> NULL and assigned(t1[-expr, ass]) then
    RETURN(-t1[-expr, ass])
  fi;
  if 20 < nops(expr) then
    rem := nops(expr);
    quo := iquo(rem, 2);
    t1 := procname(convert([op(1 .. quo, expr)], '+'), ass);
    t2 := procname(convert([op(quo + 1 .. rem, expr)], '+'),
                   ass);
    RETURN('optimize/ass'(t1 + t2, ass))
  fi;
  t1 := op(1, expr);
  if type(t1, rational) then if t1 < 0 then t1 := -t1 fi
  elif type(t1, '*') and type(op(1, t1), rational) then
    t1 := op(1, t1); if t1 < 0 then t1 := -t1 fi
  else t1 := 1
  fi;
  for i from 2 to nops(expr) while t1 <> 1 do
    t2 := op(i, expr);
    if type(t2, rational) then if t2 < 0 then t2 := -t2 fi
    elif type(t2, '*') and type(op(1, t2), rational) then
      t2 := op(1, t2); if t2 < 0 then t2 := -t2 fi
    else t2 := 1
    fi;
    if t1 <> t2 then t1 := 1 fi
  od;
```

```
    if t1 <> 1 then RETURN(t1*procname(expr/t1, ass)) fi;
    t1 := map(procname, expr, ass);
    `optimize/ass`(t1, ass)
  elif type(expr, function) and
  member(`optimize/postpro`(op(0, expr)),
          eval(`optimize/noopt`, 1)) then
        `optimize/ass`(expr, ass)
  else t1 := map(procname, expr, ass); `optimize/ass`(t1, ass)
  fi;
end;
```

```
# ------------------------------------------------------------ #
# 'Auteur: Brian Moore
#
# 'Nom: `optimize/prepro`
#
# 'Date: 12/1999
#
# 'Description:
#    Procedure optimize creee a partir de la commande
#    de Maple.
#
# 'Entrees
#    1) Expression algebrique a ajouter dans la table
#    2) Nom de la table de variables temporaires
#
# 'Sorties
#
# 'Notes
#
# ------------------------------------------------------------ #

`optimize/prepro` := proc(x)
local a, i, n;
option remember,
`Copyright (c) 1992 by the University of Waterloo.
              All rights reserved.`;
  if type(x, numeric) then x
  elif type(x, symbol) then
    i := readlib(`tools/gensym`)(x);
    `optimize/postpro`(i) := x;
    i
  elif type(x, string) then x
  elif type(x, function) then
    n := `optimize/prepro`(op(0, x));
    a := seq(`optimize/prepro`(i), i = x);
    n(a)
  elif type(x, indexed) then x
  else map(`optimize/prepro`, x)
```

```
  fi
end;
```

ANNEXE B

Détermination d'un ensemble de base

```
# ------------------------------------------------------------ #
# 'Auteur: Brian Moore
#
# 'Nom: DynParEstimation
#
# 'Date: 04/2000
#
# 'Description:
#    Procedure principale afin de trouver un ensemble de base
#    de parametres.
#
# 'Entrees
#    1) Equations dans lesquelles les parametres apparaissent
#       de facon lineaire (Les equations sont egales a 0)
#    2) Liste des parametres dynamiques a estimer
#    3) Nom de la matrice de regression en sortie
#    4) Nom du vecteur de parametres en sortie
#    5) Nom du vecteur contenant les termes independants
#       des parametres en sortie
#
# 'Sorties
#    Assignation des Entrees 3, 4 et 5
#
# 'Conditions : Les equations sont exprimees de facon
#               non-recursive (sans variables temporaires)
#
# ------------------------------------------------------------ #

DynParEstimation := proc(DynEq, ParamList, PhiBasePar::name,
                         DeltaBasePar::name, ZBasePar::name)
  `Copyright (c) Brian Moore 2000.`;
  local IndependantCol,  # Liste des indices des colonnes de
                         # la matrice de regression qui sont
                         # independante

        Delta_par,       # Parametres
        Phi_par,         # Matrice de regression
        Z_par;           # Elements independants des parametres
```

```
  Delta_par := eval(ParamList);

  ## Si il y a au moins un parametre dynamique
  if evalb(nops(SysVar[DynParList]) > 0) then

  ## 1. Trouver les valeurs de Phi et de Z
    Phi_par := GetRegressionMatrix(eval(DynEq), Delta_par);
    Z_par := GetZMatrix(eval(DynEq), Delta_par);

  ## 2. Trouve l'ensemble des parametres identifiables
    GetIdentifiableSet(evaln(Phi_par), evaln(Delta_par));

    ## Si aucun parametre identifiable
    if vectdim(eval(Delta_par)) = 0 then
      print("Aucun parametre identifiable");
    else

    ## 3. Trouve un ensemble de base
      IndependantCol := GetBaseSet(Delta_par, Phi_par,
                                          DeltaBasePar);

    ## 4. Assignation des valeurs de sorties
      PhiBasePar := submatrix(evalm(Phi_par),1..rowdim(Phi_par),
                                           IndependantCol);
      DeltaBasePar := convert(eval(DeltaBasePar), vector);
      ZBasePar := eval(Z_par);
    fi:
  else
    print("Aucun parametre a estimer");
  fi:
end:
```

```
# ------------------------------------------------------------ #
# 'Auteur: Brian Moore
#
# 'Nom: GetRegressionMatrix
#
# 'Date: 04/2000
#
# 'Description:
#    Creation de la matrice de regression.
#
# 'Entrees
#    1) Equations dans lesquelles les parametres apparaissent
#       de facon lineaire (Les equations sont egales a 0)
#    2) Liste des parametres dynamiques a estimer
#
# 'Sorties
#    1) La matrice de regression Phi
#
# ------------------------------------------------------------ #


GetRegressionMatrix := proc(eq::{array}, dynPar::{list})
  `Copyright (c) Brian Moore 2000.`;
  local regMatrix;

  ## Calcul du Jacobien de l'equation par rapport aux parametres
  regMatrix := evalm(jaco(eq, dynPar));
  regMatrix := evalm(map(expand, regMatrix));

  RETURN(regMatrix);
end:
```

```
# ------------------------------------------------------------ #
# 'Auteur: Brian Moore
#
# 'Nom: GetZMatrix
#
# 'Date: 04/2000
#
# 'Description:
#    Creation du vecteur independant des parametres.
#
# 'Entrees
#    1) Equations dans lesquelles les parametres apparaissent
#       de facon lineaire (Les equations sont egales a 0)
#    2) Liste des parametres dynamiques a estimer
#
# 'Sorties
#    1) Le vecteur independant des parametres
#
# ------------------------------------------------------------ #

GetZMatrix := proc(eq::{array}, dynPar::{list})
  `Copyright (c) Brian Moore 2000.`;
  local ZMatrix;

  ZMatrix := evalm(subs([seq(dynPar[i] = 0,i=1..nops(dynPar))],
                                                    evalm(eq)));

  ZMatrix := map2(subs,SysVar[SubStatePlus], evalm(ZMatrix));

  RETURN(RETURN(evalm(-1 * ZMatrix)););
end:
```

```
# ------------------------------------------------------------ #
# 'Auteur: Brian Moore
#
# 'Nom: GetIdentifiableSet
#
# 'Date: 04/2000
#
# 'Description:
#    Elimine les parametres qui ne sont pas identifiables,
#    c'est-a-dire les parametres qui n'apparaissent pas dans
#    les equations.
#
# 'Entrees
#    1) Nom de la matrice de regression
#    2) Nom de la liste des parametres
#
# 'Sorties
#    Modification des deux entrees.
#
# ------------------------------------------------------------ #

GetIdentifiableSet := proc(regMatrix::{name}, dynPar::{name})
  `Copyright (c) Brian Moore 2000.`;
  local i, tmp, tmp_Delta;

  tmp := [];
  tmp_Delta := [];
  for i from 1 to coldim(eval(regMatrix)) do
    if iszero(col(eval(regMatrix),i)) then
      tmp := [op(tmp), i];
    else
      tmp_Delta := [op(tmp_Delta), dynPar[i]];
    fi:
  od:

  # Aucun parametre identifiable
  if nops(tmp) = coldim(eval(regMatrix)) then
    regMatrix := matrix([NULL]);
```

```
    dynPar := vector([NULL]);
  else
    for i from nops(tmp) to 1 by -1 do
      regMatrix := delcols(eval(regMatrix), tmp[i]..tmp[i]);
    od:
    dynPar := convert(tmp_Delta, vector);
  fi:
end:
```

```
# ------------------------------------------------------------ #
# 'Auteur: Brian Moore
#
# 'Nom: GetBaseSet
#
# 'Date: 04/2000
#
# 'Description:
#   Trouve un ensemble de base.
#
# 'Entrees
#    1) Liste des parametres
#    2) Matrice de regression
#    3) Nom de la nouvelle liste de parametres
#
# 'Sorties
#    Evaluation de l'entree 3 et retourne
#    1) Liste des indices des colonnes de la matrice de
#       regression qui sont independantes
#
# 'Note
#    Toutes les variables (etats, accelerations et entrees)
#    doivent etre representees sous la forme de variables
#    indexees de la forme X[] ou U[].
#
# ------------------------------------------------------------ #

GetBaseSet := proc(Delta_par, Phi_par, DeltaBasePar::name)
  `Copyright (c) Brian Moore 2000.`;

  local IndependantCol, i, j, k, tmp, tmp_Delta,
  isThereAPossibleSolution, eqList, eq, setOfIndVar, seqSolution;

  ## La premiere colonne est independante
  IndependantCol := [1];
  DeltaBasePar := matrix(1,1,[eval(Delta_par[1])]);

  ## Pour chacune des colonnes de la matrice de regression
```

```
  for i from 2 to coldim(evalm(Phi_par)) do
    isThereAPossibleSolution := true;
    eqList := [];
    tmp := evalm(sum('alpha[IndependantCol[j]]*col(evalm(Phi_par),
                IndependantCol[j])','j'=1..nops(IndependantCol)));

    ## Cette equation est egale a 0
    eq := evalm(col(evalm(Phi_par),i) - tmp);

    ## Creation d'ensemble d'equation afin de trouver les alpha
    for j from 1 to vectdim(eq) do
      if evalb(eq[j] <> 0) then
setOfIndVar := DynFindVar(eq[j], "set");
        for k from 1 to nops(setOfIndVar) do
  eqList := [op(eqList), coeff(eq[j], setOfIndVar[k])];
        od:
      fi;
    od:

    ## Verifie s'il y a au moins une equation de la forme
    ##     valeur numerique = 0
    ## dans ce cas, il n'y a pas de solution
    for j from 1 to vectdim(eqList) do
      if type(eqList[j],numeric) and evalb(eqList[j] <> 0) then
isThereAPossibleSolution := false;
      fi:
    od:

    ## S'il y a une solution possible
    if isThereAPossibleSolution then

      ## Trouve les valeurs des constantes alpha
      seqSolution := solve(convert(eqList, set),
                     convert([seq(alpha[IndependantCol[j]],
     j=1..nops(IndependantCol))],set));

      ## Pas de solution pour alpha => parametre independant
      if evalb(seqSolution = NULL) then
```

```
        print(i, "Independant column: no solution for alpha");

IndependantCol := [op(IndependantCol), i];
        DeltaBasePar := stack((DeltaBasePar), [Delta_par[i]]);

      ## Il existe une solution pour alpha
      else
        if evalb(nops([seqSolution]) > 1) then
          print("More than one solution: not implemented yet");
else

  ## Si la solution depend des variables =>
  ##                              parametre independant
  if has(seqSolution, X) or has(seqSolution, U) then
    IndependantCol := [op(IndependantCol), i];
            DeltaBasePar := stack((DeltaBasePar), [Delta_par[i]]);

          ## Si parametre dependant
  else
            tmp := evalm(transpose(convert([[seq(subs(seqSolution,
                 alpha[IndependantCol[j]])*Delta_par[i],
 j=1..nops(IndependantCol))]], matrix)));
            DeltaBasePar := evalm(DeltaBasePar + tmp);
  fi:
        fi:
      fi:
    ## Parametre independant
    else
      IndependantCol := [op(IndependantCol), i];
      DeltaBasePar := stack((DeltaBasePar), [Delta_par[i]]);
    fi:
  od:

  RETURN(IndependantCol);
end:
```

```
# ------------------------------------------------------------ #
# 'Auteur: Brian Moore
#
# 'Nom: DynFindVar
#
# 'Date: 04/2000
#
# 'Description:
#   Trouve les variables (X[] et U[]) apparaissant dans une
#   expression.
#
# 'Entrees
#    1) Expression sous forme d'une expression algebrique,
#       d'une liste, d'un vecteur ou d'une matrice.
#
# 'Sorties
#    1) Ensemble des variables apparaissant dans l'expression
#
# ------------------------------------------------------------ #

DynFindVar := proc(zexpr::{algebraic,list,vector,matrix})
  `Copyright (c) Brian Moore 2000.`;
  option remember;
  local icount1, icount2, inum_of_element, zindexed_var;

  zindexed_var := [];

  if type(zexpr, list) then
    for icount1 from 1 to nops(zexpr) do
      zindexed_var := [op(zindexed_var),
        op(DynFindVar(zexpr[icount1], sztype_output))];
    od;

  elif type(zexpr, vector) then
    for icount1 from 1 to vectdim(zexpr) do
      zindexed_var := [op(zindexed_var),
        op(DynFindVar(zexpr[icount1], sztype_output))];
    od;
```

```
elif type(zexpr, matrix) then
  for icount1 from 1 to rowdim(zexpr) do
    for icount2 from 1 to coldim(zexpr) do
      zindexed_var := [op(zindexed_var),
op(DynFindVar(zexpr[icount1,icount2], sztype_output))];
    od;
  od;

else
  inum_of_element := nops(zexpr);
  if 1 < inum_of_element then
    ## if X[]^n or U[]^n
    if type(zexpr, indexed^numeric) then
      if evalb(has(op(1,zexpr), X) or
                      has(op(1,zexpr), U)) then
        zindexed_var := [zexpr];
      fi:
    elif type(zexpr, function^numeric) then
      if evalb(has(op(1, zexpr), X) or
                      has(op(1,zexpr), U)) then
        zindexed_var := [zexpr];
      fi:
    else
      for icount1 to inum_of_element do
        zindexed_var := [op(zindexed_var),
  op(DynFindVar(op(icount1, zexpr), sztype_output))];
      od
    fi:
  elif (type(zexpr, function) and type(op(1, zexpr), indexed))
      or type(zexpr, indexed) then

    if evalb(has(zexpr, X) or has(zexpr, U)) then
      zindexed_var := [zexpr];
    fi:
  fi;
fi;
```

```
  RETURN(convert(zindexed_var,set));
end:
```