SYMBOLIC TREATMENT FOR THE EQUATIONS OF MOTION FOR RIGID MULTIBODY SYSTEMS

Bukoko C. Ikoki[1], Marc J. Richard[2], Mohamed Bouazara[3], Sélim Datoussaïd[4]
[1] *Dép. de génie mécanique, Université de Sherbrooke, 2500 de l'Université, Sherbrooke (Qc) Canada J1K 2R1*
[2] *Département de génie mécanique, Université Laval, Québec (Qc) Canada G1V 0A6*
[3] *Dép. de génie mécanique, U. du Québec à Chicoutimi, 555 de l'Université, Saguenay (Qc) Canada G7H 2B1*
[4] *Département de génie civil, Polytechnique de Mons, Rue du Joncquois, 7000 Mons, Belgique*
E-mail: marc.richard@gmc.ulaval.ca

Received January 2009, Accepted June 2009
No. 09-CSME-21, E.I.C. Accession 3107

ABSTRACT

The library of symbolic C++ routines is broadly used throughout the world. In this article, we consider its application in the symbolic treatment of rigid multibody systems through a new software KINDA (KINematic & Dynamic Analysis). Besides the attraction which represents the symbolic approach and the effectiveness of this algorithm, the capacities of algebraical manipulations of symbolic routines are exploited to produce concise and legible differential equations of motion for reduced size mechanisms. These equations also constitute a powerful tool for the validation of symbolic generation algorithms other than by comparing results provided by numerical methods. The appeal in the software KINDA resides in the capability to generate the differential equations of motion from the choice of the multibody formalism adopted by the analyst.

TRAITEMENT SYMBOLIQUE DES ÉQUATIONS DU MOUVEMENT POUR LES SYSTÈMES MULTICORPS RIGIDES

RÉSUMÉ

La bibliothèque de routines SymboliqueC++ est largement utilisée dans le monde. Dans cet article, nous envisageons son application au traitement symbolique des systèmes multicorps rigides au travers d'un nouveau programme KINDA (KINematic & Dynamic Analysis). Outre l'attrait que représente l'approche symbolique et l'efficacité de l'algorithme C++, les capacités de manipulations algébriques de SymboliqueC++ sont mises à profit afin de produire des équations différentielles de mouvement lisibles et concises pour des mécanismes de taille relativement réduite. Ces équations constituent également un instrument puissant pour la validation directe d'un algorithme symbolique de génération des équations par un procédé autre que la comparaison des résultats fournies par voie numérique. Le recours au logiciel symbolique KINDA nous a permis d'obtenir les équations différentielles du mouvement grâce notamment au choix du formalisme multicorps adopté par l'analyste.

1. INTRODUCTION

L'approche symbolique dans le traitement des systèmes multicorps présente l'avantage reconnu d'une supériorité en temps de calcul par rapport à l'approche numérique. Outre un tel avantage, la disponibilité des équations de mouvement sous une forme lisible présente un attrait particulier, à la fois d'un point de vue pédagogique et du point de vue de certaines applications comme l'optimisation et la commande tels que décrit par Fisette [1,2].

Il est utile de préciser ce que l'on entend par ''traitement symbolique''. Un tel traitement correspond essentiellement à la génération des équations de mouvement sous forme d'expressions mathématiques. Il fait appel à un formalisme donné dont il manipule les symboles pour produire les résultats escomptés. De tels symboles peuvent être des variables, des nombres, des fonctions ou encore une combinaison des trois correspondant à une expression.

L'outil logiciel KINDA [3] que nous présentons ici permet, grâce à la librairie SymboliqueC++ [4], de fournir les équations de mouvement sous une forme lisible en mettant en œuvre le formalisme de Newton-Euler avec paramètres barycentriques. Un aspect intéressant d'un tel atout est sans doute la possibilité de vérifier visuellement, grâce à la théorie de la dynamique des solides, les équations de mouvement fournies par le programme. Une telle vérification serait ardue avec des programmes ayant un moteur symbolique non adapté au multicorps ou encore ceux produisant des équations de mouvement sous forme symbolique condensée. Nous décrivons une telle forme comme celle associée à des variables de récursion [5] pour dérouler la séquence de l'exécution d'une expression mathématique sans jamais l'exprimer explicitement.

2. DESCRIPTION DES MANIPULATIONS SYMBOLIQUES

L'algèbre sur ordinateur connaît aujourd'hui un développement important grâce à des outils logiciels généraux que l'on rencontre sur le marché. Il s'applique à des domaines où la nécessité de représenter un phénomène par des lois mathématiques se présente, permettant ainsi de manipuler les expressions associées à ces lois dans le but de résoudre des équations ou de simplifier ces expressions. Les domaines d'application sont aussi variés que la physique, les mathématiques, la chimie ou l'ingénierie.

Il existe plusieurs moteurs symboliques. Le logiciel **Axiom** [6] est un programme moderne d'algèbre sur ordinateur présentant une conception modulaire qui tire profit des relations intrinsèques entre les objets mathématiques. Le programme **Derive** [7] exploite une panoplie d'algorithmes algébriques et numériques. Il présente cependant moins de fonction que la plupart des autres moteurs et il a une vocation plutôt académique car il est implanté dans certaines calculatrices ''Texas Instrument''. Le programme **Macsyma** [8], écrit en LISP et développé initialement par les chercheurs de la M.I.T., est considéré à ce jour comme l'un des plus performants sur le marché. Il est à l'origine de deux autres moteurs symboliques, Maple et Reduce, qui s'en sont largement inspiré. Le programme **Maple** a été créé en 1980 à L'Université de Waterloo [8]. En 1988, la firme Waterloo Maple Inc. a vu le jour dans le but de créer une interface pour Maple. Ce programme, écrit en C, tire sa grande flexibilité de son importante librairie de près de 3000 fonctions. Le programme **Mathematica**, aussi écrit en C, a été conçu par la firme Wolfram Research Inc. qui le distribue également depuis sa sortie en 1988 [8]. Finalement, le moteur **MuPAD** est développé depuis 1989 à l'Université de Padenborn en Allemagne [9]. Cet algorithme est doté d'un concept de programmation orienté objet qui permet la génération dynamique de nouveaux types de données ainsi que la possibilité de charger des codes objet écrit dans d'autres langages, comme le C++, pour l'exécution du programme.

Par ailleurs, il existe des outils symboliques dédiés spécifiquement pour le traitement des systèmes multicorps. Par exemple, le logiciel **MESA VERDE**, développé par l'équipe du professeur Jens Wittenburg dans les années 70 [10], est sans doute l'un des premiers programmes précurseurs pour les systèmes multicorps rigides. Son moteur est symbolique et il est présentement exploité par plusieurs firmes telles que Daimler-Benz. Le logiciel **AUTOLEV** [11] a été originellement conçu par David Levinson. Il utilise comme formalisme la méthode de "Kane" pour générer les équations de mouvement. Il s'agit en réalité d'un langage de calcul symbolique permettant de construire étape par étape les équations de mouvement au moyen de commandes qui requiert une connaissance rigoureuse de la dynamique dans la mesure où la génération des équations n'est pas le fruit d'un processus automatique. Le logiciel **NEWEUL**, développé par les chercheurs de l'Université de Stuttgart [12], a été initié en 1977 et possède son manipulateur symbolique propre. Toutefois, son aptitude à optimiser son code n'est pas clairement documentée. Comme son nom le suggère, il recourt au formalisme de Newton-Euler. Les équations générées sont écrites en Fortran pour être traités par le moteur MAPLE. Un autre logiciel propre aux multicorps rigides est **AUTOSIM** qui a été développé par Michael Sayers de l'Université du Michigan [13]. Il est basé sur le langage LISP dont il est considéré comme une extension. Il fait appel à un formalisme basé sur la méthode de "Kane" dans sa forme récursive pour générer les équations du mouvement. **AUTOSIM** recourt à une approche orientée objet qui crée des variables intermédiaires pour la gestion des expressions. Ses possibilités d'optimisation de code sont limitées car à la vue des équations qu'il génère, on note la présence de plusieurs doublons.

À ce jour, deux logiciels se distinguent. **DynaFlexPro** [14–18] de l'Université de Waterloo est un module du logiciel symbolique MAPLE, créé par la même Université qui a récemment été incorporé dans le logiciel **MapleSim** [19]. Il semble produire des résultats intéressants qui englobent les systèmes multicorps flexibles basés sur les principes d'orthogonalité et des travaux virtuels. Toutefois, le caractère général de MAPLE et la lourdeur du traitement qui le caractérise sont un handicap constant. Finalement, le logiciel **ROBOTRAN**, écrit en C, a été développé au début des années 90 par les chercheurs de l'Université Catholique de Louvain à Louvain-la-Neuve [2]. Ce logiciel utilise aux choix la méthode des puissances virtuelles ou alors le schéma de Newton-Euler récursif avec paramètres barycentriques, inspiré des travaux du Professeur Wisama Khalil [20] du laboratoire d'automatique de Nantes. De ce point de vue, il constitue un repère essentiel au présent travail.

Le moteur symbolique de KINDA est basé sur la bibliothèque SymboliqueC++ [4], distribuée sous licence GPL. Dans le passé, il avait été établi [21] que cette librairie présentait l'inconvénient de ne pas être bien adaptée au traitement des systèmes multicorps. Ce dernier étant basé sur la manipulation des expressions trigonométriques, il était difficile d'obtenir des expressions mathématiques concises avec SymboliqueC++ pour exploiter aisément les résultats fournis par la modélisation mathématique dans une optique d'analyse. Un autre inconvénient notable tenait sans doute au fait que les équations écrites par SymboliqueC++ ne pouvaient pas toujours être lues par cette librairie. À titre d'exemple, les valeurs réelles sans décimales sont écrites comme des nombres entiers et ne peuvent être interprétées que comme tel. Il se pose un problème au moment de l'interprétation par la librairie si dans une même expression des éléments vus comme des entiers sont combinés à des variables représentant des réels. C'est notamment le cas lorsque les équations de mouvement de base, c'est-à-dire celles représentées par les matrices masse et force le cas échéant, doivent subir des manipulations supplémentaires comme le calcul nécessaire à une résolution hybride symbolique-numérique [22]. Dans cet article, en conformité avec les termes de la licence susmentionnée, nous avons entrepris d'apporter les modifications nécessaires pour l'application aux systèmes multicorps rigides. La modélisation sous SymboliqueC++ des expressions algébriques

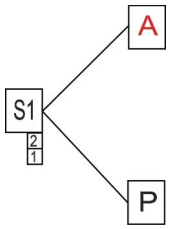

Fig. 1. Somme de deux termes.

est basée sur deux types principaux de classes d'objets. Il s'agit des classes somme et produit. À titre d'exemple, pour représenter une expression comme

$$E1 = 2 \times A + B \times \left(\cos^2 q_1 + \sin^2 q_1\right) \tag{1}$$

Le programme l'interprète d'abord comme une somme de deux termes X et Y

$$E1 = \underbrace{2 \times A}_{X} + \underbrace{B \times \left(\cos^2 q_1 + \sin^2 q_1\right)}_{Y} \tag{2}$$

À cet effet, un objet S1 de classe somme est créé pour symboliser la somme de ces deux termes comme indiqué sur la figure 1.

Avec d'une part, pour X, la variable A et pour Y, un produit d'un ensemble de facteurs. Les coefficients numériques de ces deux éléments, à savoir 2 et 1, sont spécifiés comme deux attributs de l'objet $S1$. Ces attributs sont repris dans les cases figurant sous le symbole S1 de la figure 1. Le produit (2) associé à la partie Y comprend deux arguments que sont une nouvelle somme $S2$ de deux fonctions *sin* et *cos* et la variable B. Les puissances respectives de ces deux facteurs, à savoir 1, sont indiquées comme attribut de l'objet $P1$ de classe produit sur la figure 2.

Les fonctions *sin* et *cos* référencées par l'objet $S2$ sont précédées par un objet P de classe produit qui indique la puissance des arguments. La structure de l'expression $E1$ représentée sur la figure 2 est analysée au moyen de méthodes associées aux objets de différentes classes pour appliquer les règles de manipulations algébriques programmées. Pour le cas d'espèce, l'objet $S2$ comporte deux arguments de type *sin* et *cos* à la deuxième puissance chacun et les facteurs associés aux deux termes sont 1. Le programme détectera ainsi une identité trigonométrique et appliquera la règle associée. Il remplacera les deux arguments de $S2$ par une seule branche qui pointera vers un nombre et libérera la mémoire allouée initialement aux deux branches précédentes P2 et P3. L'objet

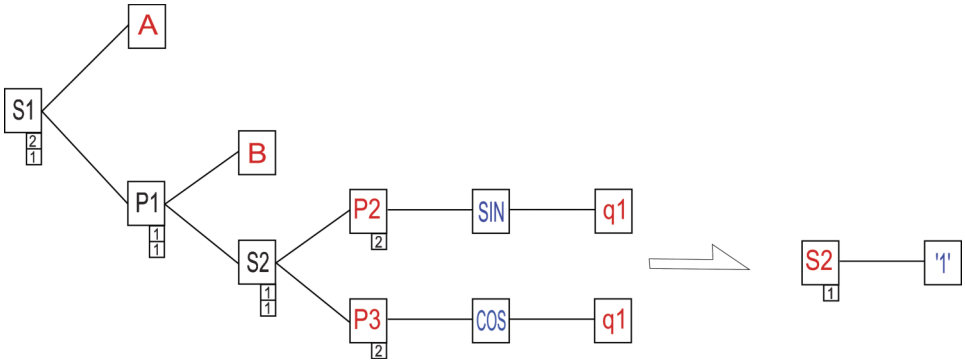

Fig. 2. Simplification de l'identité trigonométrique.

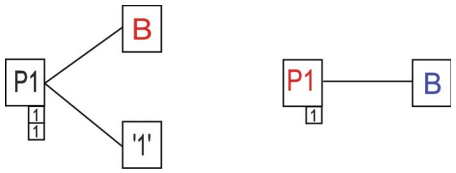

Fig. 3. Simplification de la multiplication par 1.

S2, argument de P1, sera analysé par ce dernier et sera remplacé par 1 dans un premier temps, voir figure 3. Bien entendu, la règle de base veut que la multiplication par 1 soit éliminée.

La branche relative à '1' sera supprimée avec l'indicateur de sa puissance et la mémoire associée libérée. Finalement nous obtenons plus précisément le résultat de la figure 4, ce qui correspond à l'expression $E1 = 2 \times A + B$.

Plusieurs règles de manipulations trigonométriques peuvent ainsi être programmées de manière à produire des simplifications appropriées. Il faut souligner à cet effet que la notion de simplification dans le calcul symbolique ne bénéficie pas d'une définition précise. Les outils symboliques généralistes l'implémentent suivant leurs spécificités. Il est donc important de recourir aux méthodes qui produisent des résultats conformes aux applications désirées.

À titre d'exemple, pour certains Systèmes d'Algèbre sur Ordinateur (SAO), pour $(a+b)^2$, on obtient $a^2 + 2 \cdot a \cdot b + b^2$. Ce qui d'un point de vue du multicorps est inacceptable, car on passe de deux opérations à cinq si on ne tient pas compte de la multiplication par deux. On peut aussi observer pour Maple 11 par exemple, que $2 \cdot \sin(q1) \cdot \cos(q1)$ ne peut être simplifié alors qu'il eut été souhaitable de remplacer une telle expression par $\sin(2 \cdot q1)$. Chaque programme disposant de sa propre stratégie de simplification fournira des résultats plus ou moins satisfaisants pour un même problème posé. Par exemple, pour l'expression $\sin^3(x) + \cos^3(x + \pi/6) + 3/4 \cdot \sin(3x)$, Maple et Mathematica obtiennent 0 alors que MuPAD obtient l'expression $1/2 \cdot (-\sin(x - y) - \sin(-x + y))$. Ce qui ne serait pas non plus souhaitable dans le cadre des traitements que nous entendons appliquer.

Ces exemples ne visent qu'à montrer que la simplification nécessaire à la génération d'un code acceptable exige le recours à un outil dédié aux systèmes multicorps. Outre la simplification des expressions calculées, un aspect important de la génération symbolique est l'optimisation du code en termes de redondance des expressions. Les expressions identiques qui apparaissent à plusieurs endroits du code généré doivent être remplacées par une variable unique de manière à ne les traiter qu'une seule fois.

Un exemple intéressant de manipulation algébrique est tiré de [1]. Soient les expressions $E1$, $E2$ et $E3$ suivantes

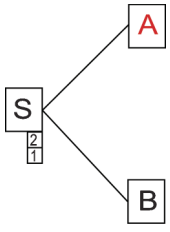

Fig. 4. Expression simplifiée.

Table 1. Exemple des simplifications.

	KINDA	ROBOTRAN	MAPLE 11
E1	$\sin q_8 \cdot \cos(q_2 + q_4) + \sin(q_5 + q_6) \cdot \cos q_8 \cdot \sin(q_4 + q_2)$	$C24 \cdot S8 + S24 \cdot S56 \cdot C8$	$\sin q_2 \cdot \cos q_4 \cdot \sin(q_5 + q_6) \cdot \cos q_8 + \cos q_2 \cdot \sin q_4 \cdot \sin(q_5 + q_6) \cdot \cos q_8 + \cos q_2 \cdot \cos q_4 \cdot \sin q_8 - \sin q_2 \cdot \sin q_4 \cdot \sin q_8$
E2	$\sin q_2 \cdot (\sin q_8 \cdot \cos(q_2 + q_4) + \sin(q_5 + q_6) \cdot \cos q_8 \cdot \sin(q_4 + q_2))$	$S2 \cdot (C24 \cdot S8 + S24 \cdot S56 \cdot C8)$	$-\sin q_2 \cdot (\sin q_2 \cdot \sin q_4 \cdot \sin q_8 - \cos q_2 \cdot \cos q_4 \cdot \sin q_8 - \cos q_2 \cdot \sin q_4 \cdot \sin(q_5 + q_6) \cdot \cos q_8 - \sin q_2 \cdot \cos q_4 \cdot \sin(q_5 + q_6) \cdot \cos q_8)$
E3	$\cos(q_2 + q_4) \cdot (-\cos(q_2 + q_4) \cdot \sin q_8 \cdot \cos(2. \cdot q_5 + 2. \cdot q_6) + \sin(q_2 + q_4) \cdot \sin(q_5 + q_6) \cdot \cos q_8)$	$C24 \cdot (-C24 \cdot C5656 \cdot S8 + S24 \cdot S56 \cdot C8)$	$-\cos(q_2 + q_4) \cdot (-\sin(q_2 + q_4) \cdot \sin(q_5 + q_6) \cdot \cos q_8 + 2 \cdot \cos(q_2 + q_4) \cdot \cos(q_5 + q_6)[2] \cdot \sin q_8 - \cos(q_2 + q_4) \cdot \sin q_8)$

$$
\begin{aligned}
E1 = {} & \cos q_2 \cdot \cos q_4 \cdot \cos(q_5 + q_6) \cdot \cos(q_5 + q_6) \cdot \sin q_8 \\
& + \cos q_2 \cdot \sin q_4 \cdot \sin(q_5 + q_6) \cdot \cos q_8 \\
& - \sin q_2 \cdot \sin q_4 \cdot \sin(q_5 + q_6) \cdot \sin(q_5 + q_6) \cdot \sin q_8 \\
& - \sin q_2 \cdot \sin q_4 \cdot \cos(q_5 + q_6) \cdot \cos(q_5 + q_6) \cdot \sin q_8 \\
& + \cos q_2 \cdot \cos q_4 \cdot \sin(q_5 + q_6) \cdot \sin(q_5 + q_6) \cdot \sin q_8 \\
& + \sin q_2 \cdot \cos q_4 \cdot \sin(q_5 + q_6) \cdot \cos q_8
\end{aligned} \tag{3a}
$$

$$
\begin{aligned}
E2 = {} & -\sin q_2 \cdot \sin q_2 \cdot \sin q_4 \cdot \sin(q_5 + q_6) \cdot \sin(q_5 + q_6) \cdot \sin q_8 \\
& + \cos q_2 \cdot \sin q_2 \cdot \cos q_4 \cdot \cos(q_5 + q_6) \cdot \cos(q_5 + q_6) \cdot \sin q_8 \\
& + \cos q_2 \cdot \sin q_2 \cdot \cos q_4 \cdot \sin(q_5 + q_6) \cdot \sin(q_5 + q_6) \cdot \sin q_8 \\
& - \sin q_2 \cdot \sin q_2 \cdot \sin q_4 \cdot \cos(q_5 + q_6) \cdot \cos(q_5 + q_6) \cdot \sin q_8 \\
& + \cos q_2 \cdot \sin q_2 \cdot \sin q_4 \cdot \sin(q_5 + q_6) \cdot \cos q_8 \\
& + \sin q_2 \cdot \sin q_2 \cdot \cos q_4 \cdot \sin(q_5 + q_6) \cdot \cos q_8
\end{aligned} \tag{3b}
$$

$$
\begin{aligned}
E3 = {} & -\cos(q_2 + q_4) \cdot \cos(q_2 + q_4) \cdot \cos(q_5 + q_6) \cdot \cos(q_5 + q_6) \cdot \sin q_8 \\
& + \cos(q_2 + q_4) \cdot \cos(q_2 + q_4) \cdot \sin(q_5 + q_6) \\
& \cdot \sin(q_5 + q_6) \cdot \cos q_8 \cdot \cos q_8 \cdot \sin q_8 \\
& + \cos(q_2 + q_4) \cdot \cos(q_2 + q_4) \cdot \sin(q_5 + q_6) \cdot \sin(q_5 + q_6) \cdot \sin q_8 \\
& \cdot \sin q_8 \cdot \sin q_8 \\
& + \cos(q_2 + q_4) \cdot \sin(q_2 + q_4) \cdot \sin(q_5 + q_6) \cdot \cos q_8 \cdot \cos q_8 \cdot \cos q_8 \\
& + \cos(q_2 + q_4) \cdot \sin(q_2 + q_4) \cdot \sin(q_5 + q_6) \cdot \cos q_8 \cdot \sin q_8 \cdot \sin q_8
\end{aligned} \tag{3c}
$$

Le tableau 1 présente un comparatif des résultats des simplifications. Notons que les résultats du logiciel ROBOTRAN sont toutefois à considérer par rapport aux variables auxiliaires créées

qui encapsulent des opérations antérieures. Il faut également tenir compte des opérations d'affectation associées.

3. TRAITEMENT MULTICORPS AVEC KINDA

3.1. Caractéristiques du programme

Deux approches sont généralement utilisées pour la modélisation physique des mécanismes. D'une part, on peut recourir à l'édition d'un fichier de données reprenant la description topologique du mécanisme, d'autre part, on peut faire usage d'une interface permettant la production d'un tel fichier. Cette dernière approche a été retenue pour ce programme, car il a l'avantage de guider l'utilisateur dans la description du mécanisme et de limiter ainsi les erreurs.

Il est intéressant de noter que cette dernière approche est sensiblement différente de celle utilisée dans les logiciels numériques qui font appel à des constructions géométriques. L'approche symbolique a la particularité de faire abstraction de telles constructions et des formes associées, à telle enseigne que la conception d'un système mécanique peut être ramenée au remplissage d'un simple formulaire.

L'approche en coordonnées relatives a été retenue. Le nombre minimal d'équations différentielles qu'elle procure est un atout attrayant. Le caractère ordinaire des équations différentielles obtenues est aussi un atout intéressant pour la résolution de telles équations.

Divers formalismes ont été conçus pour permettre un nombre limité d'opérations dans la génération des équations de mouvement. Renaud [5] propose la méthode de Newton-Euler avec paramètres barycentriques [10] pour la dynamique inverse des structures arborescente sous forme de chaine ($O(N)$). Fisette et Samin [1,2] présentent une méthode ($O(N^2)$) de Newton-Euler adaptée à la dynamique directe en fournissant les équations sous la forme dite semi-explicite propice pour les applications en robotique et utile pour le traitement des systèmes présentant des boucles cinématiques.

Wittenburg [10] explique toutefois, qu'il est possible en recourant à une approche récursive adaptée avec la méthode des puissances virtuelles de générer les équations de mouvement sous la forme matricielle $A\ddot{q} = B$ avec une complexité d'ordre N (nombre des corps) pour le calcul de A et $\boldsymbol{B}$.

3.2. Modélisation mathématique

Nous adoptons pour KINDA [3] les algorithmes de Newton-Euler pour la dynamique directe et inverse. Pour les systèmes ouverts, les équations de mouvement peuvent être données sous forme implicite, dans le cas de la dynamique inverse,

$$Q = f(q, \dot{q}, \ddot{q}, \alpha, \mathcal{F}, \mathcal{M}) \tag{4}$$

Pour la dynamique directe, elles peuvent être données sous forme explicite

$$\ddot{q} = f(q, \dot{q}, \alpha, \mathcal{F}, \mathcal{M}) \tag{5}$$

ou encore sous forme matricielle où les n équations se présentent comme suit

$$M(q)\ddot{q} + c(q, \dot{q}, \alpha, \mathcal{F}, \mathcal{M}) = Q(q, \dot{q}) \tag{6}$$

Dans ces équations

- $\boldsymbol{Q}$ représente la matrice ($n \times 1$) des efforts articulaires appliqués.
- $\boldsymbol{q}$ le vecteur des coordonnées généralisées.

– α l'ensemble des paramètres caractéristiques du système en ce compris la gravité.
– $\mathcal{F}$ et $\mathcal{M}$ l'ensemble des efforts appliqués sur les corps.
– $\boldsymbol{c}$ la matrice ($n \times 1$) des efforts dynamiques de gravité, des effets gyroscopiques et de Coriolis.
– M la matrice masse du système.

Pour les systèmes contraints, on introduit les multiplicateurs de Lagrange λ et la jacobienne $J = \partial h(q)/\partial q^T$ des contraintes pour ramener le mécanisme à un système ouvert équivalent.

$$M(q)\ddot{\boldsymbol{q}} + c(q, \dot{\boldsymbol{q}}, \alpha, \mathcal{F}, \mathcal{M}) + J^T\lambda = Q(q, \dot{\boldsymbol{q}}) \tag{7}$$

Ces n équations sont associées aux m équations de contraintes (scléronômes)

$$h(q) = 0 \tag{8}$$

et leurs dérivés

$$\begin{aligned} \dot{h}(q, \dot{\boldsymbol{q}}) &= J(q)\dot{\boldsymbol{q}} = 0 \\ \ddot{h}(q, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) &= J(q)\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(q)\dot{\boldsymbol{q}} = 0 \end{aligned} \tag{9}$$

Wehage [23] montre qu'il est possible pour un tel système algébro-différentiel de $n + m$ équations de le ramener à un système de n équations différentielles ordinaires. Pour cela, on fixe parmi les n coordonnées, $n - m$ coordonnées q_i indépendantes correspondant aux degrés de liberté du système. les m autres q_d sont dépendantes. En notant $\boldsymbol{F} = \boldsymbol{Q} - \boldsymbol{c}$, les équations de mouvement prennent alors la forme partitionnée suivante

$$M^{ii}(q)\ddot{\boldsymbol{q}}_i + M^{id}(q)\ddot{\boldsymbol{q}}_d + J_i^T\lambda = F^i(q, \dot{\boldsymbol{q}}) \tag{10a}$$

$$M^{id}(q)\ddot{\boldsymbol{q}}_i + M^{dd}(q)\ddot{\boldsymbol{q}}_d + J_d^T\lambda = F^d(q, \dot{\boldsymbol{q}}) \tag{10b}$$

$$h(q) = 0 \tag{10c}$$

$$J_d(q)\dot{\boldsymbol{q}}_d + J_i(q)\dot{\boldsymbol{q}}_i = 0 \tag{10d}$$

$$J_d(q)\ddot{\boldsymbol{q}}_d + J_i(q)\ddot{\boldsymbol{q}}_i = -\dot{\boldsymbol{J}}(q)\dot{\boldsymbol{q}} \tag{10e}$$

À partir de (10c) et (10d), en posant $b(q, \dot{\boldsymbol{q}}) = -\dot{\boldsymbol{J}}(q)\dot{\boldsymbol{q}}$ on montre que l'on peut réduire (10a), (10b) et (10e) à

$$M_r(q_i)\ddot{\boldsymbol{q}}_i = F_r(q_i, \dot{\boldsymbol{q}}_i) \tag{11}$$

où

$$\begin{aligned} M_r &= M^{ii} - M^{id}J_d^{-1}J_i - J_i^T J_d^{-T}\left[M^{id} - M^{dd}J_d^{-1}J_i\right] \\ F_r &= F^i - M^{id}J_d^{-1}b - J_i^T J_d^{-T}\left[F^d - M^{dd}J_d^{-1}b\right] \end{aligned}$$

J_i et J_d étant constitués respectivement des colonnes correspondant aux coordonnées indépendantes et dépendantes dans J. Les indices i et d des matrices masses partitionnées

correspondent aux lignes et aux colonnes associées respectivement aux coordonnées indépendantes et dépendantes dans la matrice masse de départ. Les coordonnées dépendantes sont déterminées à partir des coordonnées indépendantes par résolution du système non linéaire (10c). La méthode de Newton-Raphson est utilisée à cet effet. Elle est décrite par les itérations suivantes.

$$q_d^{k+1} = q_d^k - J_d^{-1} h(q) \tag{12}$$

La convergence est contrôlée par la norme $\|h(q)\|$. Les vitesses et les accélérations dépendantes sont obtenues à partir de (10d) et (10e). Soit,

$$\dot{q}_d = -J_d^{-1} J_i \dot{q}_i \tag{13a}$$

$$\ddot{q}_d = -J_d^{-1} J_i \dot{q}_i + J_d^{-1} b \tag{13b}$$

Les efforts de liaison sont obtenus par l'expression de λ, d'après [1]. À partir de (10b), nous avons

$$\lambda = J_d^{-T} \left[F^d - M^{id} \ddot{q}_i - M^{dd} \ddot{q}_d \right] \tag{14}$$

Les variables articulaires commandées introduisent des contraintes cinématiques qui sont levées par la même technique que celle décrite précédemment. Dans le cas où les contraintes géométriques et cinématiques sont combinées, les variables articulaires associées à ces dernières sont considérées comme un sous-ensemble des variables indépendantes. Une partition supplémentaire est opérée pour extraire les grandeurs réduites (M_r, F_r) appropriées.

3.3. Organisation des traitements

Les traitements des mécanismes avec KINDA [3] se déroulent en quatre phases: modélisation physique, génération (modélisation mathématique), le cas échéant présimulation et enfin simulation.

La modélisation physique consiste en la description du mécanisme dans une approche parent-enfant. Le système est considéré comme ouvert et les conditions de fermeture de boucles sont définies le cas échéant. Les corps sont définis par rapport à leur précédent, le premier étant le repère inertiel. Deux types de liaisons (rotoïde et prismatique), ou leur combinaison, peuvent connecter chaque corps enfant à son corps parent. Un repère local de référence est attaché à chacun des corps. Tous les points ainsi que les paramètres inertiels sont définis par rapport à ce dernier. Une configuration initiale identique est attribuée à chacun de ces repères. Des efforts peuvent être définis sur les corps. Les forces sont appliquées sur des points préalablement définis. D'autres points peuvent servir à la définition des boucles cinématiques et des contraintes associées.

Trois types de contraintes peuvent être définis. Celle réalisant la suppression d'une liaison rotoïde ou sphérique, celle pratiquant la coupure d'un corps et celle éliminant un bras entre deux liaisons rotoïdes ou sphériques. Entre deux corps peut être attaché un ensemble ressort-amortisseur. Les deux types de liaison de base peuvent comporter un amortisseur et/ou un ressort. Les différents paramètres décrivant le mécanisme sont des grandeurs symboliques ou numériques. À la fin de la modélisation, un fichier reprenant la topologie du système et les paramètres associés est créé. Un fichier de définition en langage SymboliqueC++ est également

créé. Ils y sont définis notamment les variables associées aux grandeurs symboliques, les matrices de rotation associées aux différentes liaisons rotoïdes, les matrices d'inertie de divers corps, etc… Tous les éléments nécessaires à l'algorithme de génération seront définis dans ce fichier automatiquement par le module de modélisation de KINDA.

La génération fournit les équations de mouvement sous forme des matrices M, $\boldsymbol{c}$ et $\boldsymbol{Q}$ pour la dynamique directe ou sous forme d'un vecteur correspondant à l'équation implicite pour la dynamique inverse. Pour les systèmes présentant des boucles cinématiques, il faut indiquer quelles sont les coordonnées indépendantes. Parmi ces dernières, on indique celles qui sont contrôlées et on précise les lois de la commande sous forme symbolique. Les équations de contraintes $\boldsymbol{h}(\boldsymbol{q})$ ainsi que leurs dérivées peuvent être générées. La jacobienne associée à ces dernières ainsi qu'aux contraintes introduites par les variables articulaires commandées est générée. La dérivée de cette dernière aussi. Une première partition est effectuée pour fournir la matrice jacobienne partitionnée nécessaire à la résolution du système (10). Cette partition peut être modifiée par l'utilisateur en définissant une matrice de permutation qui opérera les changements nécessaires. Un fichier représentant le modèle mathématique du système est ainsi créé avec toutes les grandeurs nécessaires à la résolution des équations de mouvement.

La présimulation peut être appliquée pour des systèmes relativement modestes. Elle utilise le fichier créé par le module de génération pour fournir symboliquement les matrices partitionnées nécessaires à la résolution de (10) pour les systèmes bouclés. Le système linéaire fournissant les expressions des accélérations généralisées à partir de (11) est résolu symboliquement. Pour les systèmes contrôlés ou contraints géométriquement, les multiplicateurs de Lagrange peuvent être déterminés symboliquement. L'utilisateur a cependant le choix de les générer ou pas, tant ils occasionnent un coût de calcul généralement important.

La simulation consiste à intégrer les accélérations généralisées produites par le module de présimulation lorsque cela s'applique. En fonction des conditions initiales, vitesses et positions seront déterminées. Un algorithme de type Runge-Kutta 4 est implémenté à cet effet. Des méthodes plus élaborées peuvent également être implémentées. En présence de contraintes, l'algorithme de Newton-Raphson permet de déterminer les coordonnées, vitesses et accélérations dépendantes. Les expressions des contraintes (10c) et de la Jacobienne générées symboliquement sont mises â profit pour un tel calcul. Les expressions symboliques des paramètres contrôlés sont simplement évaluées en fonction du temps. Les multiplicateurs de Lagrange associés aux efforts nécessaires pour produire de tels mouvements sont également évalués directement à partir de leur expression symbolique fournie par le module de présimulation. Ceux liés aux contraintes géométriques d'ouverture de boucle sont également évalués lorsque cela est demandé. Pour ce module de simulation, un exécutable du système est produit qui procède aux calculs de simulation nécessaire en fonction des paramètres symboliques auxquels on affecte les valeurs numériques souhaitées.

4. EXEMPLES DE TRAITEMENT DES SYSTÈMES

À titre d'exemple, nous nous proposons de déduire analytiquement les équations du mouvement d'un pendule oscillant à l'intérieur d'une rainure verticale, retenu par un ressort, tel que représenté dans la figure 5. Par la suite, ce mécanisme sera modélisé avec le logiciel symbolique KINDA.

Le bras S2 de longueur L et de masse m tourne autour du point A attaché à la glissière S1, de masse négligeable. Cette dernière coulisse suivant la direction x avec une compliance imposée

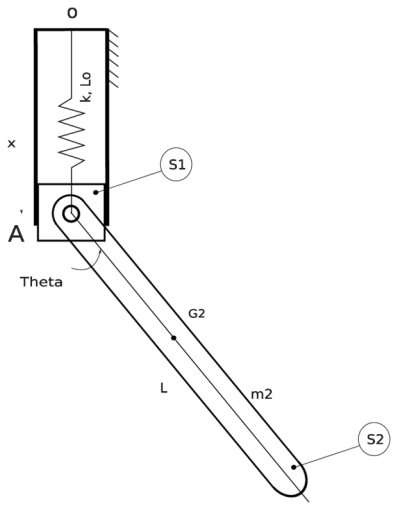

Fig. 5. Pendule coulissant.

par le ressort de raideur k. À partir de l'énergie cinétique T et de l'énergie potentielle V, on se propose de produire les équations de mouvement par le principe de Hamilton.

$$T = \frac{1}{2}m\left(\dot{x}^2 - \dot{x}\dot{\theta}L\sin\theta + \frac{L^2}{4}\dot{\theta}^2\right) + \frac{1}{2}I_{G2z}\dot{\theta}^2 \quad \text{avec} \quad I_{G2z} = \frac{mL^2}{12}$$

$$V = mg\left(h - x - \frac{L}{2}\cos\theta\right) + \frac{1}{2}kx^2$$

Par le principe de Hamilton, nous pouvons écrire

$$\int_{t_1}^{t_2} \delta\left(\frac{1}{2}m\left(\dot{x}^2 - \dot{x}\dot{\theta}L\sin\theta + \frac{L^2}{4}\dot{\theta}^2\right)\right.$$
$$\left. + \frac{1}{2}I_{G2z}\dot{\theta}^2 - mg\left(h - x - \frac{L}{2}\cos\theta\right) - \frac{1}{2}kx^2\right)dt = 0$$

Après développement, on a

$$\int_{t_1}^{t_2}\left[m\left(\dot{x} - \dot{\theta}\frac{L}{2}\sin\theta\right)\delta\dot{x} + \left(m\left(\dot{x}\dot{\theta}\frac{L^2}{4}\dot{\theta} - \dot{x}\frac{L}{2}\sin\theta\right) + \frac{1}{2}I_{G2z}\dot{\theta}^2\right)\delta\dot{\theta}\right.$$
$$\left. + (mg - kx)\delta x - \left(-m\dot{x}\dot{\theta}\frac{L}{2}\cos\theta + mg\frac{L}{2}\sin\theta\right)\delta\theta\right]dt = 0$$

Les déplacements virtuels étant les seuls pouvant être considérés comme arbitraires, on élimine $\delta\dot{x}$ et $\delta\dot{\theta}$ en intégrant par parties les expressions suivantes.

Pour $\delta\dot{x}$,

$$\int_{t_1}^{t_2}\left[m\left(\dot{x} - \dot{\theta}\frac{L}{2}\sin\theta\right)\delta\dot{x}\right]\,dt$$
$$= \left[m\left(\dot{x} - \dot{\theta}\frac{L}{2}\sin\theta\right)\right]\delta x\Big|_{t_1}^{t_2} - \int_{t_1}^{t_2} m\frac{d}{dt}\left(\dot{x} - \dot{\theta}\frac{L}{2}\sin\theta\right)\delta x\,dt$$

Puisque $\delta x = 0$ pour $t = t_1$ et $t = t_2$, cette dernière intégrale vaut

$$-\int_{t_1}^{t_2} m\left(\ddot{x} - \ddot{\theta}\frac{L}{2}\cos\theta\right)\delta x\, dt$$

Également, pour $\delta\dot{\theta}$,

$$\int_{t_1}^{t_2}\left[\left[m\left(\frac{L^2}{4}\dot{\theta} - \dot{x}\,L\sin\theta\right) + I_{G2z}\dot{\theta}\right]\delta\dot{\theta}\right]\, dt =$$

$$\left[m\left(\frac{L^2}{4}\dot{\theta} - \dot{x}\,L\sin\theta\right) + I_{G2z}\dot{\theta}\right]\delta\theta\Big|_{t_1}^{t_2} - \int_{t_1}^{t_2}\left[m\left(\frac{L^2}{4}\dot{\theta} - \dot{x}\,L\sin\theta\right) + I_{G2z}\dot{\theta}\right]\,\delta\theta dt$$

$$= -\int_{t_1}^{t_2}\left[m\left(\frac{L^2}{4}\ddot{\theta} - \ddot{x}\,L\sin\theta - \dot{x}\dot{\theta}L\cos\theta\right) + I_{G2z}\ddot{\theta}\right]\,\delta\theta\, dt$$

On a finalement

$$\int_{t_1}^{t_2} -\left[m\left(\ddot{x} - \ddot{\theta}\frac{L}{2}\sin\theta - \dot{\theta}^2\frac{L}{2}\cos\theta + mg - kx\right)\delta x\right.$$

$$\left. -\left[m\left(\frac{L^2}{4}\ddot{\theta} - \ddot{x}L\sin\theta - \dot{x}\dot{\theta}L\cos\theta\right) - I_{G2z}\ddot{\theta} - m\dot{x}\dot{\theta}\frac{L}{2}\cos\theta + mg\frac{L}{2}\sin\theta\right]\delta\theta\right]\, dt$$

$$= 0$$

Pour obtenir les équations de mouvement, on fait appel au caractère arbitraire des déplacements virtuels δx et $\delta\theta$.

En imposant $\delta\theta = 0$, cette dernière intégrale devant se vérifier pour tout δx, elle s'annulera lorsque

$$m\left(\ddot{x} - \ddot{\theta}\frac{L}{2}\sin\theta - \dot{\theta}^2\frac{L}{2}\cos\theta\right) - mg + kx = 0 \tag{15}$$

Alternativement, quand $\delta x = 0$, on obtient

$$m\frac{L^2}{4}\ddot{\theta} - m\ddot{x}\,L\sin\theta + I_{G2z}\ddot{\theta} + mg\frac{L}{2}\sin\theta = 0 \tag{16}$$

4.1. Traitement du pendule coulissant avec KINDA

Le corps de référence 0 est le sol tel que représenté sur la figure 6. Un repère lui est automatiquement attaché en (0, 0, 0). Par rapport à ce dernier, on précise la direction de la gravité (ici dans la direction 1). On en indique la valeur, ici la variable symbolique *g*. On indique l'emplacement d'un joint à utiliser pour le corps enfant, ici (0, 0, 0).

On ajoute un premier corps de type fictif représenté sur la figure 7. On indique pour cela son parent, le corps 0. On précise le type de liaison avec ce dernier, ici prismatique.

Par défaut, la connection se fait au niveau du repère local du corps 1. On précise la direction de la liaison prismatique, ici 1. On indique un emplacement de joint pour le corps enfant. Dans

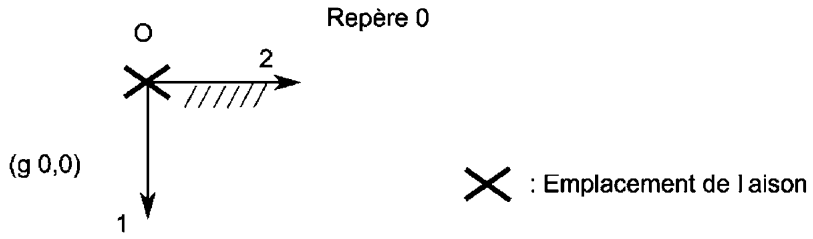

Fig. 6. Configuration du repère inertiel.

la définition du joint, on indique qu'un ressort est incorporé. On indique sa raideur et sa longueur naturelle, ici k et L_o. Le deuxième corps, montré en figure 8, est ajouté au premier. On indique le point de jonction sur le précédent, ici (0, 0, 0), attaché au corps 1. Le type de joint est rotoïde. On indique l'emplacement du centre de masse. Celui-ci est donné par rapport au repère local et est conforme à l'orientation initiale du corps, soit (L/2, 0, 0). La masse du corps 2 est renseignée. Aucun autre point que le centre de masse ne doit être créé. Il n'y a pas non plus d'emplacement de joint pour un corps enfant.

Avec ces informations, KINDA enregistre la topologie du mécanisme dans un fichier. Le programme produit un second fichier des paramètres en langage SymboliqueC++ qui seront appliqués à l'algorithme de génération.

Le fichier des paramètres produit par le module de modélisation est utilisé pour la génération. Un des paramètres que contient ce fichier est celui qui renseigne sur la structure du mécanisme. Ici, il s'agit d'une structure 2D ouverte avec 2 degré de liberté (ddl). En fonction de ces informations, le module de génération pose les questions qui détermineront l'analyse à effectuer. Les matrices M, c et Q seront produites automatiquement en fonction des informations récupérées dans le fichier des paramètres. Pour la dynamique inverse, les équations associées sont également générées dans le fichier de sortie.

Cas I: système libre. Puisque le système est ouvert, le programme demande le nombre des coordonnées généralisées qui seront commandées, ici 0. Le système est donc libre. Cette information sera reprise par le module de présimulation.

Cas II: une de deux coordonnées est commandée. Le nombre de coordonnées commandées à indiquer est 1. Le système demande d'indiquer la loi de la commande. Le programme génère une Jacobienne associée à la contrainte cinématique. Il réalise une partition de cette dernière qui sera utilisée dans la présimulation.

Cas III: les deux coordonnées sont commandées. Le système demande d'indiquer les lois de commande.

Les équations de base s'affichent sur l'écran de la figure 9. Il est intéressant d'observer les équations produites par l'algorithme de Newton-Euler sans les paramètres barycentriques. Ces

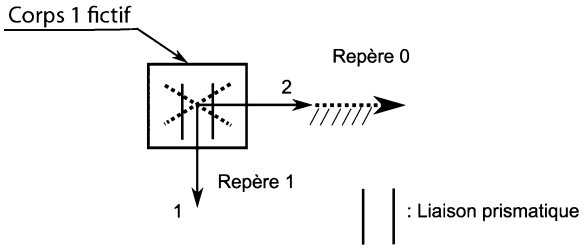

Fig. 7. Ajout du premier corps.

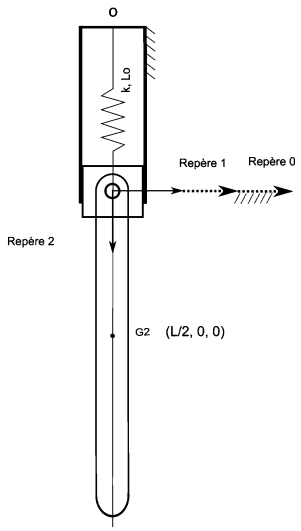

Fig. 8. Ajout du deuxième corps.

dernières sont affichées sur l'écran de la figure 10. On note la facilité que procure le recours aux paramètres barycentriques à une meilleure simplification des équations de mouvement.

Dans la phase de présimulation, les équations nécessaires à la simulation sont générées.

Cas I: système libre. L'expression des accélérations généralisées est écrite dans le fichier de sortie.

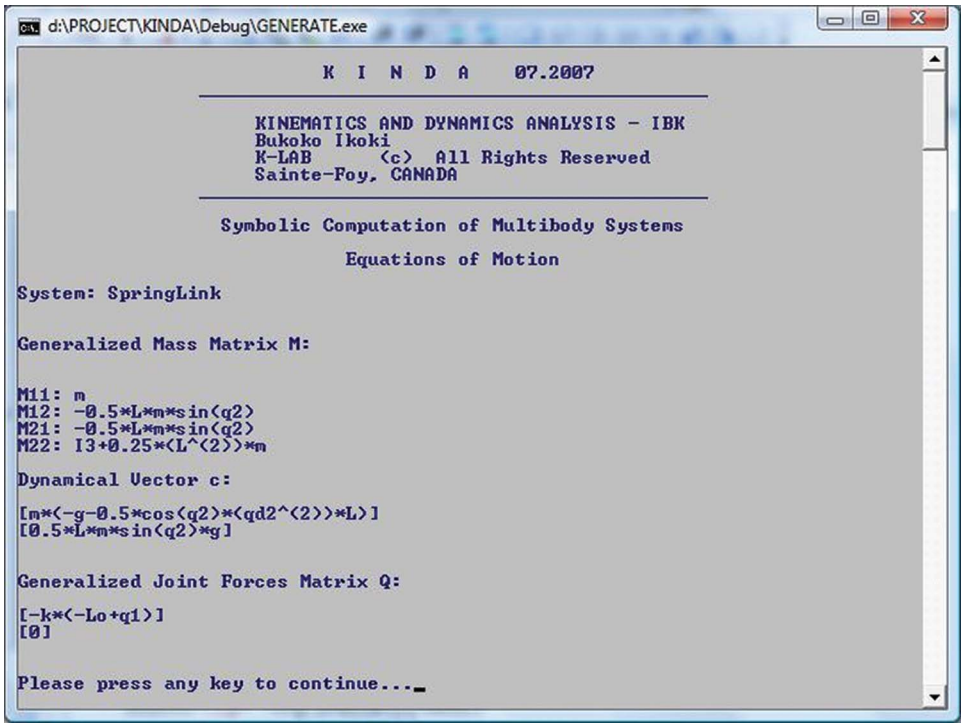

Fig. 9. Équations de mouvement de base générées par KINDA.

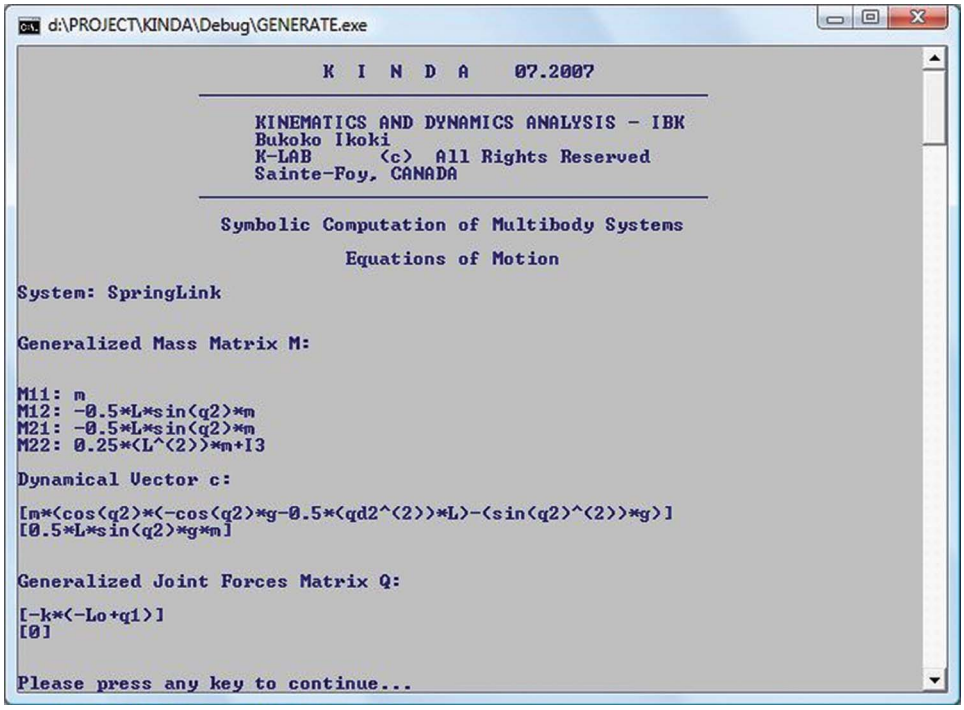

Fig. 10. Équations de mouvement sans paramètres barycentriques.

$$\ddot{q}_1 = -\frac{k}{m}(-L_o - q_1) + \left(g + L\left(0.5\cos q_2 \dot{q}_2^2 + \frac{0.5L\sin q_2^2\left(0.25m\cos q_2\dot{q}_2^2 L - 0.5k(-L_o+q_1)\right)}{mL^2\left(-0.25\sin q_2^2 + 0.25\right) + I_3}\right)\right)$$

$$\ddot{q}_2 = \frac{L\sin q_2\left(0.25m\cos q_2\dot{q}_2^2 L - 0.5k(-Lo+q1)\right)}{mL^2\left(-0.25\sin q_2^2 + 0.25\right) + I_3}$$

Cas II: une de deux coordonnées est commandée. L'expression de l'accélération généralisée de la coordonnée libre est produite. Celle de l'effort nécessaire à la commande est également produite.

$$\ddot{q}_2 = -2\frac{-0.5L\sin q_2 g - \ddot{q}_1}{L\sin q_2}$$

$$\lambda_1^c = m\left(\ddot{q}_1 + L\left(-0.5\sin q_2\ddot{q}_2 - 0.5\cos q_2\dot{q}_2^2\right) - g\right) + k(-L_o + q_1)$$

Alternativement, on a

$$\ddot{q}_1 = -\frac{k(-Lo+q_1) + m\left(g + L\left(0.5\cos q_2\dot{q}_2^2 + 0.5\sin q_2\ddot{q}_2\right)\right)}{m}$$

$$\lambda_2^c = Lm\sin q_2(-0.5\ddot{q}_1 + 0.5g) + \left(I_3 + 0.25L^2 m\right)\ddot{q}_2$$

Cas III: les deux coordonnées sont commandées. Seules les efforts nécessaires à la commande sont générés. Les résultats sont données ci-après.

$$\lambda_1^c = Lm \sin q_2(-0.5\ddot{q}_1 + 0.5g) + (I_3 + 0.25L^2 m)\dot{q}_2$$

$$\lambda_2^c = m(\ddot{q}_1 + L(-0.5 \sin \dot{q}_2\ddot{q}_2 - 0.5 \cos q_2\dot{q}_2^2) - g) + k(-L_o + q_1)$$

Dans tous les trois cas, des valeurs numériques du système pour tous les paramètres symboliques utilisés peuvent être précisées. Elles sont utilisées comme valeur par défaut au moment de la simulation.

Grâce au fichier de sortie de présimulation contenant les expressions nécessaires, dans le cas I, le module de simulation demandera automatiquement les conditions initiales, en figure 11, pour les coordonnées 1 et 2. Grâce à l'expression de l'accélération généralisée, l'intégration de cette dernière est réalisée pour obtenir vitesse et position généralisée.

Dans le cas II, seule une de deux coordonnées est concernée par le traitement précédent. La coordonnée commandée est simplement évaluée avec ses dérivées. Dans le cas III, les paramètres commandés (position, vitesse et accélération) sont évalués en fonction du temps. Les multiplicateurs de Lagrange associés aux efforts nécessaires à la commande sont également évalués.

Finalement, une routine Matlab est générée pour réaliser automatiquement le tracé des différentes courbes. La figure 12 trace les coordonnées généralisées (q_1, q_2), les vitesses généralisées ($\dot{q}_1$, $\dot{q}_2$) et les accélérations généralisées ($\ddot{q}_1$, $\ddot{q}_2$) du pendule coulissant.

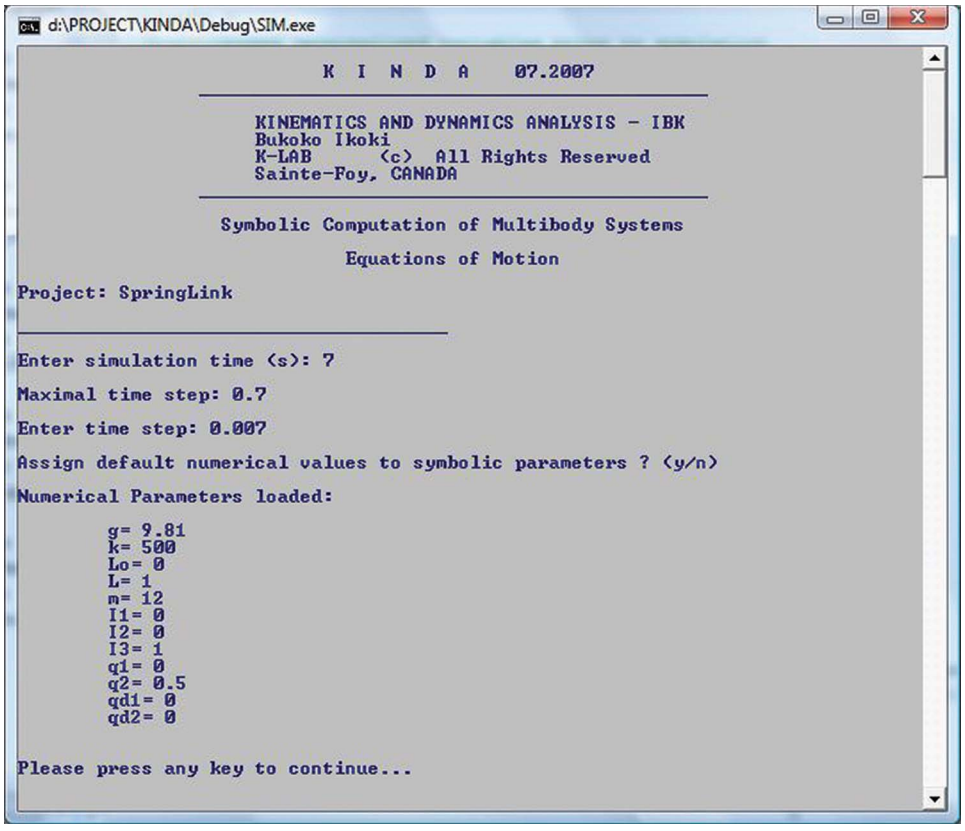

Fig. 11. Paramètres de simulation (pendule coulissant).

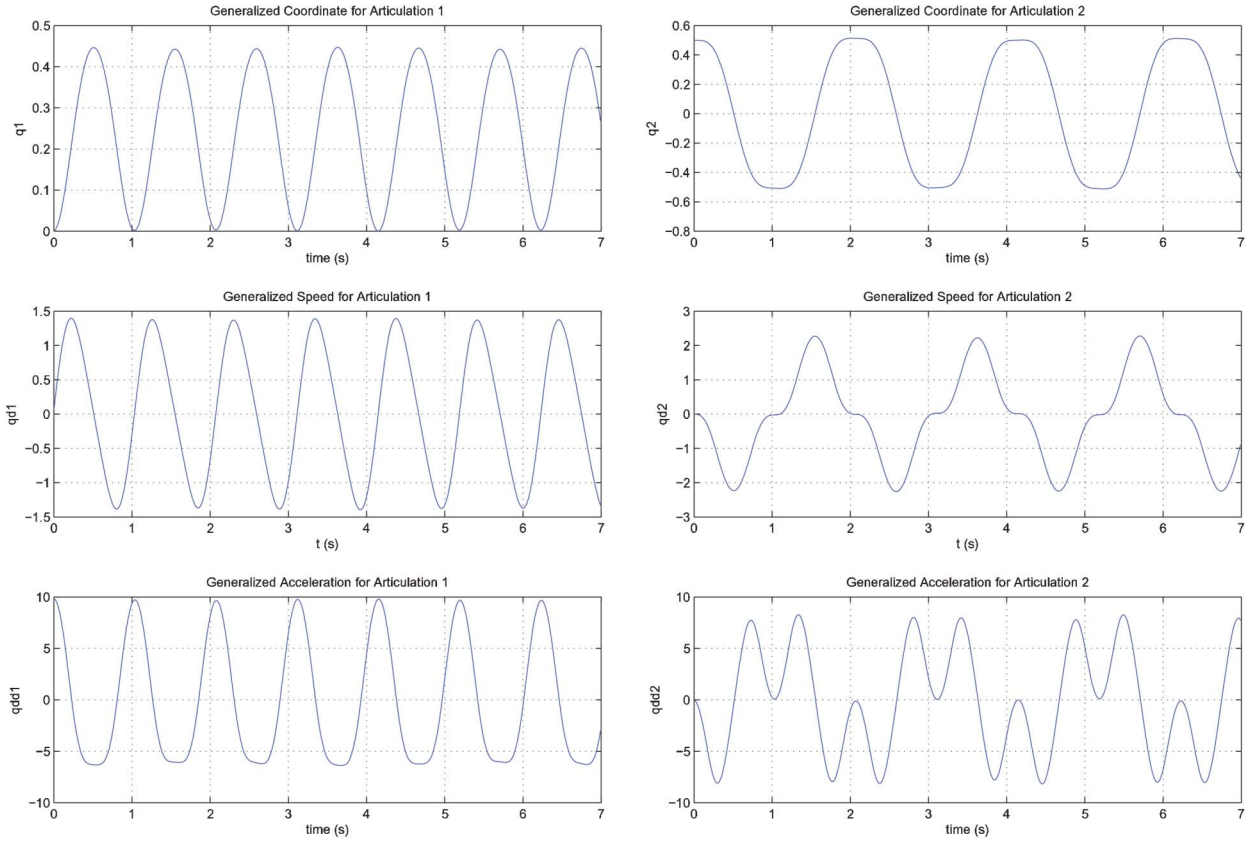

Fig. 12. Positions, vitesses et accélérations des coordonnées généralisées (pendule coulissant).

5. CONCLUSION

La conception du logiciel KINDA nous a permis de présenter une application pour le traitement des systèmes multicorps rigides à partir de la bibliothèque de routines de calcul algébrique assisté par ordinateur SymboliqueC++. Ce nouveau logiciel se situe encore dans une phase expérimentale et vise l'intégration dans une stratégie globale d'optimisation des systèmes multicorps. De ce point de vue, il tire le meilleur profit des différentes méthodes existantes afin de générer des codes pouvant produire des analyses avec une célérité propice aux applications des plus exigeantes comme l'optimisation. Pour la modélisation, il est doté d'un moteur symbolique dédié qui exploite le formalisme Newton-Euler de type récursif avec paramètres barycentriques en utilisant un système de coordonnées relatives. Les expressions traitées sont véritablement manipulées au sens algébrique du terme, rendant ainsi la nécessité d'un recourt aux variables intermédiaires inutile. Rapellons que son moteur symbolique lui offre la possibilité de gérer de manière automatique les doublons. Outre l'exportation des codes générés, KINDA est aussi doté d'un module de simulation.

Les capacités de manipulation du logiciel KINDA sont considérables car nous avons apporté les modifications nécessaires pour arriver à réaliser le traitement des systèmes dynamiques multicorps. Le recours à SymboliqueC++ nous a permis de produire des équations de mouvement offrant une lisibilité appréciable, grâce notamment au choix du formalisme multicorps adopté. Ultimement, une telle lisibilité a servi à la validation du programme KINDA grâce à la théorie de la dynamique des solides, en comparant directement les résultats fournis

par KINDA et ceux calculés à la main. Une telle démarche aurait été ardue dans une approche produisant des équations sous forme symbolique condensée avec des variables de récursion. Ces dernières restent cependant utiles pour des systèmes de taille importante. L'autre approche de validation aurait sans doute été le recours à la comparaison des courbes produites avec ceux des outils numériques populaires. Cependant, une telle démarche est discutable dans la mesure où de tels outils ne sauraient être considérés comme des étalons, tant les résultats fournis (quand ils peuvent l'être) par ces derniers sont tributaires des méthodes numériques de résolution qui sont elles-mêmes liées à la nature du mécanisme étudié.

RÉFÉRENCES

1. Fisette, P. and Samin, J.C., *Symbolic Modeling of Multibody Systems*, Kluwer Academic Publishers, 2003.
2. Fisette, P., Postiau, T., Saat, L. and Samin, J.C., ''Fully Symbolic Generation of Complex Multibody Models,'' *Mechanics of Structures and Machines*, vol. 30, pp. 31–82, 2002.
3. Ikoki, B.C., *Traitement symbolique des systèmes multicorps avec KINDA*, mémoire de maîtrise, Université Laval, Québec, Québec, Canada, 2008.
4. Shi, T.K., Steeb, W.H. and Hardi, Y., *Symbolic C++*, Second extended and revised edition, Springer-Verlag, 2000.
5. Renaud, M., ''Quasi-minimal Computation of the Dynamic Model of a Robot Manipulator utilizing the Newton-Euler Formalism and the Notion of Augmented Body,'' *IEEE Journal of Robotics and Automation*, pp. 1677–1682, 1987.
6. Jenks, R.D., *AXIOM - The Scientific Computation System*, Springer-Verlag, 1992.
7. Kutzler, B. and Kokol-Voljc, V., *Introduction to Derive*, Kutzler Kokol-Voljc OEG, Austria, 2000.
8. Cohen, J.S., *Computer Algebra and Symbolic Computation; Elementary Algorithm*, A.K. Peters Ltd, 2002.
9. Creutzig, C. and Oevel, W., *MuPad Tutorial*, Springer, Scientific Interfaces, 2e edition, 412 p., 2004.
10. Wittenburg, J., *Dynamics of Multibody Systems*, Springer-Verlag Berlin, 2008.
11. Kane, T. and Levinson, D., *Autolev User's Manual*, http://www.autolev.com/, 154 p., 2005.
12. Schiehlen, S., *Multibody Systems Handbook*, Springer-Verlag, 432 p., 1990.
13. Sayers, M.W., *Symbolic Computer Methods to Automatically Formulate Vehicle Simulation Codes*, Ph.D. Thesis, University of Michigan, 1990.
14. Shi, P. and McPhee, J., ''Dynamics of Flexible Multibody Systems Using Virtual Work and Linear Graph Theory,'' *Multibody Systems Dynamics*, Springer, vol. 4, pp. 355–381, 2000.
15. Schmitke, C., Morency, K. and McPhee, J., ''Using Graph Theory and Symbolic Computing to Generate Efficient Models for Multibody Vehicle Dynamics,'' *Journal of Multibody Dynamics*, Professional Engineering Publishing, vol. 222, no. 4, pp. 339–352, 2008.
16. McPhee, J., Schmitke, C. and Redmond, S., ''Dynamic Modelling of Mechatronic Multibody Systems With Symbolic Computing and Linear Graph Theory,'' *Mathematical and Computer Modelling of Dynamical Systems*, Taylor and Francis Publishing, vol. 10, no. 1, pp. 1–23, 2004.
17. Schmitke, C. and McPhee, J., ''Forming Equivalent Subsystem Components to Facilitate the Modelling of Mechatronic Multibody Systems,'' *Multibody Systems Dynamics*, Springer, vol. 14, no. 1, pp. 81–110, 2005.
18. Vogt, H.S., Schmitke, C., Jalali, K. and McPhee, J., ''Unified Modelling and Real-Time Simulation of an Electric Vehicle,'' *International Journal of Vehicle Autonomous Systems*, vol. 6, no. 3, pp. 288–307, 2008.

19. MapleSim-High Performance Multi-Domain Modeling and Simulation, http://www.maplesoft.com/products/maplesim/, Maplesoft, consulté le 8 mars 2009.
20. Khalil, W. Kleinfinger, J.F., ''Minimum Operations and Minimum Parameters of the Dynamic Models of Tree Structure Robots,'' *IEEE Journal of Robotics and Automation*, RA-3, vol. 6, pp. 517–526, 1987.
21. Richard, M.J., McPhee, J. and Anderson, R., ''Computerized Generation of Motion Equations Using Variational Graph Theoretic Methods,'' *Applied Mathematics and Computation*, vol. 192(1), pp. 135–156, 2007.
22. Geike, T. and McPhee, J., On the Automatic Generation of Inverse Dynamic Solutions for Parallel Manipulators, *WORKSHOP on Fundamental Issuesand Future Research Directions for Parallel Mechanisms and Manipulators*, University of Waterloo, 2002.
23. Wehage, R.A., *Generalized Coordinate Partitioning in Dynamical Analysis of Mechanical Systems*, Ph.D. Thesis, University of Iowa, 1980.