THESE de DOCTORAT de l'UNIVERSITE d'EVRY-VAL D'ESSONNE

Spécialité :

MECANIQUE

présentée par

Nicolas SEGUY

pour obtenir le titre de

DOCTEUR de l'UNIVERSITE d'EVRY-VAL d'ESSONNE

Modélisation modulaire de systèmes articulés : Conception orientée-objet de plates-formes de simulation

---

Soutenue le 5 Décembre 2003

devant le jury composé de :

M. A. BARRACO	Professeur à l'ENSAM	Rapporteur
M. A. CURNIER	Professeur à l'EPFL	Rapporteur
M. C. VALLEE	Professeur à l'Université de Poitiers	Rapporteur
Mme M. PASCAL	Professeur à l'Université d'Evry	Directrice de Thèse
M. Z.-Q. FENG	Professeur à l'Université d'Evry	Examinateur
M. P. JOLI	Maître de Conférence à l'Université d'Evry	Examinateur

A ma famille

A Sandrine,

A Baptiste,

A mes enfants à venir…

Remerciements

Ce travail de thèse a été réalisé au sein du Laboratoire Systèmes Complexes à l'Université d'Evry. Je tiens à exprimer dans cette page mes remerciements à toutes les personnes qui ont contribuées à son aboutissement.

*Je remercie **Madeleine Pascal**, Professeur à l'Université d'Evry, d'avoir accepté de diriger ce travail de recherche. Merci pour vos conseils, vos relectures attentives et enfin pour votre patience.*

*Je remercie **André Barraco**, Professeur à l'Ecole Nationale Supérieure des Arts et Métiers, **Alain Curnier**, Professeur à l'Ecole Polytechnique Fédérale de Lausanne et **Claude Vallée**, Professeur à l'Université de Poitiers, d'avoir accepté d'être les rapporteurs de cette thèse.*

*Mes plus vifs remerciements vont à **Pierre Joli**, Maître de Conférence à l'Université d'Evry, pour sa disponibilité. Merci pour le temps passé, sans se décourager, jours après jours, parfois même soirs et week-end. J'espère que la fin de ce travail est un point de départ pour une longue collaboration. Que Janice soit associée à ces remerciements.*

*Merci à **Zhi-Qiang Feng**, Professeur à l'Université d'Evry, membre du Laboratoire de Mécanique d'Evry, pour sa disponibilité, pour ses connaissances encyclopédiques, pour ses quantités de références toujours appropriées, pour ses avis toujours éclairés... Merci de m'avoir permis de 'm'insérer' dans FER/Mech.*

*Je remercie **Florent Chavand**, Professeur à l'Université d'Evry, directeur du Laboratoire Systèmes Complexes, d'avoir permis la réalisation de cette thèse.*

Que tous les membres du laboratoire soient ici associés à ces remerciements: doctorants, maîtres de conférence, professeurs ou collaborateurs 'industriels' qui ont participé de près, de loin ou même pas du tout, à la réalisation de ce travail, à ceux de l'étage ou du rez-de-chaussée, pour les matchs permanents-doctorants, pour l'aventure des JJCR'14,….

Merci enfin à mes proches, pour leur soutien, toujours présents même dans les moments les plus difficiles de ces années passées. Merci à mes amis toujours fidèles, Pierre, Sophie, mes compagnons des 'Agafousses'. Merci à 'Bob' (très respectueusement), Professeur à l'Université de Toulouse, qui m'a suivi et encouragé même de loin.

Merci à Sandrine, pour avoir assuré l'essentiel, pour avoir été patiente, pour Baptiste, pour Gudule et les autres,….

Notations

Notations associées au système global

q : Vecteur des coordonnées généralisées du système global.

n : Dimension de q.

m : Nombre total de contraintes algébriques.

n_ℓ : Nombre total de liaisons.

n_c : Nombre total de corps.

d : Degrés de liberté du système.

ϕ_{cs} : Contraintes de corps solides dites *internes*.

ϕ_ℓ : Contraintes de liaisons dites *externes*.

C : Matrice jacobienne des contraintes de dimension $[m \times n]$.

H : Jacobienne des contraintes *internes*.

J : Jacobienne des contraintes *externes*.

μ : Multiplicateurs de Lagrange de corps solides du système complet.

σ : Multiplicateurs de Lagrange de liaisons du système complet.

J_{ij} : Jacobienne de la liaison *i* s'exerçant sur le corps C_j.

Notations associées au corps C_j

q_j : Vecteur des coordonnées du corps C_j.

n_j : Dimension de q_j.

n_{ℓ_j} : Nombre de liaisons appliquées au corps C_j.

n_{cs_j} : Nombre de contraintes algébriques de corps solide associées au corps C_j.

$n_{c\ell_j}$: Nombre de contraintes algébriques de l'ensemble des liaisons associées au corps C_j.

M_j : Matrice d'inertie associée au corps C_j de dimension $[n_j \times n_j]$.

F_{ℓ_j} : Somme des efforts de liaisons appliqués au corps C_j.

H_j : Jacobienne des contraintes de corps solide associées au corps C_j, de dimension $[n_{cs_j} \times n_j]$.

J_j : Jacobienne des contraintes de liaisons de dimension $\left[n_{\ell j} \times n\right]$.

μ_j : Multiplicateurs de Lagrange de corps solide du corps C_j de dimension n_{cs_j}.

σ_j : Multiplicateurs de Lagrange de liaisons du corps C_j, de dimension $n_{c\ell_j}$.

Notations associées aux liaisons

n_{c_i} : Nombre de contraintes algébriques de la liaison i.

Introduction générale

1. Contexte de l'étude

La simulation de systèmes multicorps, comme la simulation de systèmes physiques au sens large, exige de réaliser une grande quantité de calculs afin d'obtenir à partir des équations du mouvement, l'évolution temporelle des grandeurs décrivant le système. L'outil informatique est donc particulièrement bien adapté au calcul des positions, vitesses et accélérations par résolution numérique des équations différentielles issues des lois de la mécanique. La recherche dans le domaine a donc initialement porté sur une adaptation au traitement numérique des représentations et des formalismes pour une automatisation et une systématisation de la génération des équations du mouvement, sous forme numérique ou symbolique, et de leur résolution. Une grande partie des études concerne également le contrôle des coûts numériques afin de rendre acceptable les temps de résolution.

Dans un souci de réduction des coûts de développement d'un produit, les nouveaux outils de simulation se dotent de nouvelles fonctionnalités afin de faciliter et donc d'accélérer la mise en œuvre des résolutions. L'environnement de travail des logiciels est ainsi rendu plus convivial non seulement pour la gestion des fichiers de données (modèles ou résultats) mais également pour la construction de ces modèles et la visualisation de ces résultats. La construction de pièces ou de mécanismes s'effectue alors virtuellement au travers d'interfaces graphiques et une part de l'analyse des résultats de simulation est sous forme d'animation 3D de la scène modélisée. Grâce à ces interfaces, l'utilisateur est donc libéré de toute contrainte de programmation et se concentre donc sur les compétences nécessaires à son métier.

Cependant, les interactions en dynamique avec le monde virtuel restent limitées. Aucun événement extérieur ne peut agir en ligne sur la scène afin de modifier le scénario de simulation. Le concepteur doit en effet suivre le cycle traditionnel de conception qui passe successivement par les trois étapes de pré-traitement, traitement et post-traitement qui réalisent respectivement la construction des pièces et des mécanismes, le calcul puis la visualisation des résultats. Ces étapes sont dissociées et des modifications sur la scène ne sont possibles qu'après analyse des résultats obtenus dans la dernière étape.

Ces outils de simulation sont donc rigides par deux aspects. Ils se situent en effet à un stade déjà trop avancé du processus de conception et ne dispensent pas d'une étude préalable, éventuellement sur papier, pour penser la forme des pièces et des assemblages. De plus, ils figent les modèles et il n’est alors pas possible d’assembler, que ce soit de manière interactive ou non, des modèles développés par différentes équipes de simulation sans une centralisation et une gestion des données extrêmement complexes.

C'est de ce constat que s'est imposée l'idée d'une étude en vue de la réalisation d'un outil d'aide à la conception de systèmes mécaniques pour la construction rapide et intuitive de maquettes digitales avec interaction en dynamique.

2. Objectifs

2.1 Problématique

Inclure des propriétés d'interactivité à un logiciel de simulation implique de pouvoir faire réagir la simulation à tout événement extérieur produit par exemple par l'utilisateur. Autoriser des interactions avec un système multicorps peut donc se traduire par la possibilité de:

- supprimer des contraintes pour libérer des liaisons entre solides et modifier la topologie du mécanisme.
- imposer des contraintes cinématiques telles celles induites par la manipulation directe d'un objet de la scène.
- appliquer des efforts.
- modifier l'environnement et créer de nouvelles possibilités de collisions.

Les paramètres du système, les formalismes employés ainsi que les méthodes de résolution doivent donc contenir, dans leur définition même, la possibilité de prendre en compte ces événements. L'outil de simulation doit de plus être modifié afin d'inclure les interactions avec l'environnement extérieur dans le processus de traitement qui réalise de manière simultanée, et non plus successivement, la résolution et l'animation.

2.2 Approche modulaire

Dans les formes classiques de résolution de problèmes de dynamique, le mécanisme à étudier est vu comme un unique système conduisant à des équations différentielles associées résolues de manière globale. La structure initiale du prototype fige ainsi pour toute la durée de la simulation les paramètres représentatifs du système tels que les tailles des vecteurs d'état ou des matrices représentatives. La suppression d'une liaison correspond à l'annulation de multiplicateurs de Lagrange ou bien de lignes et de colonnes dans une matrice. Le formalisme doit donc contenir à l'avance l'ensemble des possibilités d'évolution. La modélisation ne possède donc pas sous cette forme les propriétés de flexibilité nécessaires pour définir un outil interactif.

Cette flexibilité est rendu possible si le système global est vu comme un ensemble de corps ou de groupements de corps élémentaires modélisés séparément et liés par des liaisons. Chacune de ces entités élémentaires doit posséder toutes les caractéristiques nécessaires à son existence propre, indépendamment du reste de l'environnement. Cette modélisation introduit la notion de composant *mécanique*, constitué non seulement des caractéristiques physiques et des paramètres nécessaires à sa représentation mais également des méthodes locales de résolution de la dynamique (schéma d'intégration, pas de la simulation).

Le système global n'existe quant à lui que par l'existence de relations entre ces entités élémentaires. Des changements dans la structure de la chaîne cinématique peuvent donc être réalisés à tout instant par des modifications de ces relations sans affecter la définition des composants *mécaniques* eux-mêmes.

Un système multicorps se définit alors par un assemblage de composants mécaniques: composants *corps* et composants *liaison*. La modélisation est modulaire.

Le paramétrage choisi pour représenter le composant *corps* doit donc être compatible avec la modularité et donc déterminer la position du corps quelle que soit l'architecture de la chaîne cinématique. Ce choix doit de plus définir de manière simple les composants *liaison*

qui sont des relations algébriques entre les coordonnées de sous-systèmes différents, nommées contraintes *externes*. Cette simplicité est assurée si le nombre de paramètres est supérieur aux degrés de liberté du sous-système nécessitant l'introduction de relations dites de contraintes *internes*. La dynamique d'un sous-système est donc représentée par un système d'équations algébro-différentielles nommées équations *locales* de la dynamique.

Les relations de contraintes *internes* et *externes* sont traitées séparément. Dans un premier temps, les composants *liaison* calculent les contraintes *externes*, par une expression explicite des violations de contraintes, donnant les efforts de liaison. Puis, dans un deuxième temps, le système algébro-différentiel des équations *locales* de la dynamique est résolu, connaissant ces efforts de liaison. Au sein de ces composants mécaniques, le calcul des contraintes internes peut s'effectuer cette fois à partir d'une expression implicite des violations de contrainte.

Les méthodes de résolution de contrainte, à partir de formulations implicites ou explicites, conduisent à de l'instabilité numérique. Il est indispensable de se doter de critères de stabilité permettant une mise en œuvre fiable et automatique en fonction du pas de temps de simulation sans qu'il soit nécessaire d'avoir une connaissance approfondie des composants eux-mêmes.

Cette modélisation modulaire convient alors à une programmation orientée objets dont les propriétés d'encapsulation permettent de lier directement les objets informatiques aux composants mécaniques et donc aux objets réels, corps seuls ou assemblages de corps ainsi qu'aux liaisons. Le changement dans la topologie ne se traduit alors, sans calcul supplémentaire, que par l'ajout ou la suppression d'une instance de la classe définissant les liaisons.

Afin d'accélérer les temps de calculs et même si ce n'est pas l'objet de ce mémoire, la modélisation proposée laisse envisager la possibilité de résolution sur plusieurs processeurs ou une distribution vue comme un partage de ressources entre ordinateurs distants liés par un réseau.

2.3 Plan du mémoire

Le rapport présente, dans une première partie, une vue d'ensemble des logiciels de simulation et des méthodes de résolution modulaire déjà existantes. Ce bilan permet de définir les besoins pour la problématique propre à la construction d'un outil interactif de simulation

pour l'assemblage virtuel de mécanismes en terme de modélisation mécanique et de construction informatique.

La deuxième partie est un catalogue de diverses représentations, formalismes nécessaires à l'obtention des équations du mouvement, méthodes de résolution et schémas numériques d'intégration exposant ainsi les outils généralement employés en dynamique des systèmes multicorps.

Le troisième chapitre décrit les choix à effectuer sur les coordonnées et formalismes respectant l'indépendance entre les corps afin de définir les composants mécaniques et donc de construire un modèle sous une forme modulaire.

Associée à cette modélisation modulaire, les méthodes de résolution propres, respectant les propriétés de modularité, sont présentées dans le quatrième chapitre.

Les concepts de la programmation orientée objet, adaptés à la modélisation proposée, ainsi que les développements informatiques réalisés au sein du concept FER/System sont exposés au cinquième chapitre.

Le sixième et dernier chapitre présente les résultats de simulation de quelques systèmes 2D et 3D afin de valider l'algorithme général de résolution. Pour le premier exemple simulant un pendule double, les trajectoires du système sont comparées à celles obtenus par des méthodes classiques (représentation minimale ou résolution centralisée). Puis, un système soumis à des changements de topologie est également proposé afin de mettre en évidence l'apport de la méthode en terme de modularité et donc de flexibilité. Enfin, le dernier exemple est une simulation interactive pendant laquelle les contraintes de liaisons sont supprimées dynamiquement au travers de l'interface graphique afin de montrer les possibilités de l'association de la programmation orientée objet à une résolution modulaire.

CHAPITRE I

Vers une approche modulaire

I.1 Introduction: outils de simulation

Le but de l'utilisation des logiciels de simulation et de l'amélioration de leur interface est essentiellement de diminuer de manière notable les coûts de développements en réduisant les temps de cycles de conception de nouveaux produits. Ce cycle de développement classique de l'ingénieur en mécanique consiste en une analyse statique, cinématique et dynamique puis en un affinage de la conception en fonction des résultats obtenus [Dix99].

L'outil informatique permet, grâce à la rapidité des calculs, de tester à moindres coûts un grand nombre de solutions aidé en cela par des outils d'optimisation. Les outils de visualisation et d'animation 3D produisent des représentations suffisamment réalistes, appelées maquettes virtuelles pour s'affranchir de la réalisation de prototypes réels, réduisant donc également les coûts de développement. Ces interfaces affranchissent de plus le concepteur ou l'ingénieur des problèmes liés à l'informatique et à la programmation. Les détails d'implémentation sont masqués et le concepteur agit sur ce monde virtuel au travers de l'interface uniquement par les paramètres propres à sa tâche.

L'étude et la conception de logiciels de simulation sont des activités pluridisciplinaires regroupant les domaines de la dynamique des systèmes multicorps, l'informatique, l'analyse numérique, l'infographie, l'animation, la mécatronique, l'étude des contacts, la détection de collision,…. L'ensemble des activités, propres à la dynamique des systèmes multicorps sont présentés dans un article de synthèse [Sch97] sur les origines, l'état de l'art et les perspectives de la dynamique des systèmes multicorps.

L'évolution des logiciels permet également d'envisager de nouvelles applications telles que l'étude des tâches d'assemblages et de désassemblages virtuels, les études ergonomiques,…, [Jay97], [Buc98], [Gom99].

Après avoir décrit la structure classique d'un outil de simulation, ce chapitre présente les propriétés recherchées pour la résolution de problèmes par une approche modulaire et plus particulièrement pour la construction de logiciels de simulation de la dynamique de systèmes multicorps.

I.1.1 Description

Quel que soit le domaine d'étude, dynamique des systèmes multicorps rigides ou flexibles, dynamique des structures, l'outil informatique et les logiciels de calculs sont devenus un outil indispensable pour l'analyse mécanique. De nombreux logiciels commerciaux ou plus académiques, permettent de simuler le comportement d'assemblages de corps rigides ou flexibles liés par des liaisons mécaniques. Ces outils se composent généralement de trois modules organisés comme présentés sur la figure I-1.

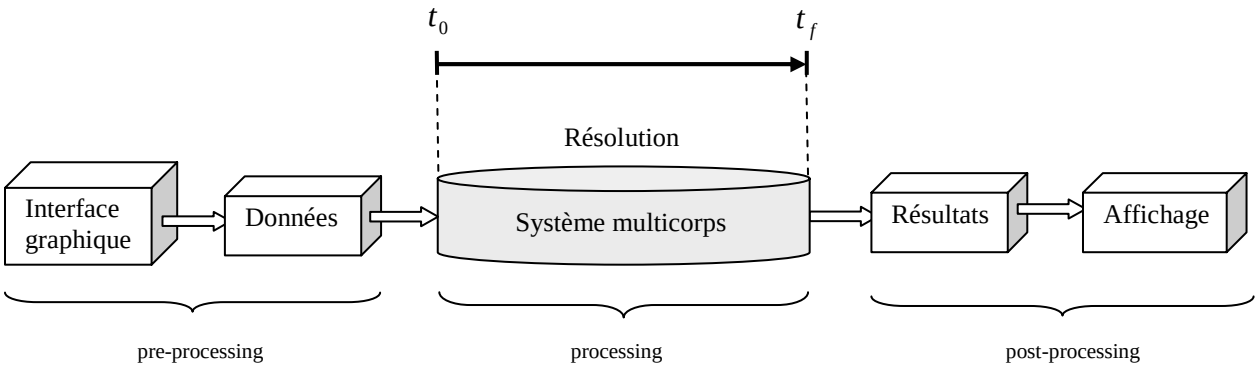


Figure I-1 : Structure classique d'un outil de simulation

Tout d'abord, au travers d'une interface graphique appelée modeleur, l'utilisateur construit les différents corps du système en leur attribuant leurs caractéristiques physiques (dimensions, inertie,…) puis définit les liaisons entre ces corps afin de construire la topologie du système. Dans une deuxième étape, les équations du mouvement sont construites et résolues dans une phase purement calculatoire à partir des données fournies par l'étape de pre-processing. Enfin, dans une dernière étape, des outils permettent de visualiser les résultats, éventuellement sous forme d'animation.

Outre le modeleur et l'environnement d'animation, ces logiciels diffèrent principalement par la manière de traiter le problème dynamique, par exemple, le type de coordonnées ou le formalisme employé pour établir les équations du mouvement. Il est à noter que ces étapes sont découplées et n'offrent donc pas la possibilité à un opérateur d'agir en ligne sur le système. L'ensemble des données de simulation sont fixées une fois pour toute dans la phase

de conception et leur modification ne se fait qu'après avoir complètement exécuté le cycle, c'est-à-dire après analyse des résultats.

I.1.2 Exemples de développement

Le marché offre de nombreux outils de simulation, désormais aboutis, offrant toutes les fonctionnalités nécessaires à l'étude complète de systèmes multicorps: Adams [Www1] (Adams/view, Adams/solver), Solidworks [Www3] et les modules de résolution associés (Motion works, Cam works), Ideas [Www4], Catia [Www2], DADS [Www6],….

Les modeleurs de ces différents produits tournent autour d'un même principe. Les volumes des différentes pièces sont construits de manière incrémentale par l'ébauche d'esquisses de formes de base ensuite extrudées pour créer ou enlever de la matière.

Grâce à des modules optionnels, le logiciel est muni de fonctionnalités plus spécifiques de résolutions pour l'analyse statique, cinématique et dynamique utilisant éventuellement, suivant le mode de résolution, des outils de maillage.

D'autres fonctionnalités sont également développées telles que la production automatique de dessins d'ensembles [Che02] ou la synthèse des systèmes multicorps [Rég97], correspondant au calcul optimisé des dimensions d'un mécanisme pour la réalisation d'une tâche donnée.

I.1.3 Simulation et réalité virtuelle

Pour tous ces logiciels, l'environnement graphique 3D est très convivial et permet non seulement de créer les corps et de les assembler graphiquement mais aussi de visualiser les résultats. Ces outils de simulation utilisent des techniques de réalité virtuelle. La réalité virtuelle qui peut être vue comme une extension de l'infographie 3D consiste à donner à l'utilisateur l'impression que l'environnement est réel, parfois même qu'il fait lui-même partie de la scène. Cette sensation d'immersion nécessite alors d'augmenter les possibilités d'interaction entre les mondes réels et virtuels.

Classiquement, comme pour les logiciels énumérés précédemment, l'interaction (création, manipulation et visualisation) consiste en une combinaison d'entrées provenant de la souris et du clavier. L'utilisation de la souris, dont les déplacements sont 2D, est combinée à celle du clavier pour modifier le plan actif ou le point de vue de l'utilisateur. Une solution peut être

l'utilisation d'autres types d'interfaces telles que auditives, tactiles [Hof98] ou vocales en plus de l'interface visuelle traditionnelle. La combinaison des informations provenant de ces différentes sources fait alors intervenir des méthodes de traitement multimodales [Chu97], [Gao00]. Ainsi des outils plus élaborés se développent et des interfaces spécifiques sont mises au point : vision stéréoscopique, data gloves [Stur94], [Ket01].

Il est à noter également que le domaine de l'animation fait aussi appel à des techniques de dynamique des systèmes multicorps ([Gas94], [Par97]) avec cependant quelques techniques propres concernant la gestion de la complexité (faces cachées, points de vue, niveaux de détail, …) [Noc01], [Val99] ou pour gérer l'interactivité [Wit90]. Cependant dans toutes ces applications, les modèles dynamiques sont souvent simplifiés à l'extrême au profit du temps de calcul afin de satisfaire la contrainte temps-réel (ou réaliste) pour un rendu haptique acceptable (retour d'effort). Dans le cas où un rendu visuel suffit, il est possible d'augmenter la complexité des modèles mais en général l'interaction se situe entre un objet prédéfini de la scène et l'opérateur. L'accent est mis en effet sur la détection de collision et donc la génération automatique des contraintes [Red02].

I.2 Objectifs d'une approche modulaire

Cette section présente les objectifs liés à l'utilisation d'approches modulaires ainsi que différents domaines utilisant une approche modulaire dans la résolution des problèmes. A l'analyse de ces applications, il apparaît que des techniques modulaires sont employées pour répondre à deux types d'exigences :

- gérer des niveaux de complexité que les méthodes centralisées classiques ne sont pas aptes à traiter.
- apporter de nouvelles propriétés telles que la flexibilité ou la possibilité de distribution qu'un traitement classique ne possède pas.

I.2.1 Gestion de la complexité

Cette complexité peut avoir plusieurs origines:

- le système global, dans sa construction ou dans sa résolution, nécessite un couplage de modèles provenant de domaines différents.

- le nombre de paramètres utilisés pour représenter le système est très important (dynamiques des structures, systèmes poly-articulés complexes,…).
- les temps de résolution, pour un système donné, sont très importants.
- le système est lui-même physiquement distribué (réseaux informatiques, coopération de robots mobiles,…)

Concernant le premier point, l'exemple de la construction d'une automobile est significatif. Les études nécessitent en effet l'expertise de mécaniciens dans des spécialités aussi diverses que l'aérodynamique, la dynamique des structures ou la thermodynamique entre autres. Le produit final est quant à lui la mise en commun de la résolution des problèmes liés à chacun des domaines. De même, l'étude et la simulation de systèmes mécatroniques sont issues de couplages entre modèles mécaniques, électriques et lois de commande. En micro-robotique, l'intégration de systèmes multi-physiques fait intervenir des processus physiques très divers (mécanique, phénomènes piezo-électriques, …[Fat97]).

Pour la simulation de systèmes poly-articulés, d'autres raisons, que la possibilité d'assemblages virtuels de composants mécatroniques, motivent l'utilisation d'une approche modulaire. En effet, deux inconvénients limitent la modélisation centralisée à partir de coordonnées généralisées relatives (coordonnées articulaires) pourtant efficace d'un point de vue numérique :

- le traitement des boucles cinématiques, qui est possible, mais s'avère très difficile.
- la modélisation de jeux et de flexibilité dans les articulations, dans ce cas impossible.

Même si l'approche modulaire est plus coûteuse en temps de calculs car elle conduit à un plus grand nombre d'équations à résoudre, ceci est atténué par la possibilité de distribuer et de calculer en parallèle des sous-systèmes relativement simples.

Le problème de la simulation peut donc s'avérer très complexe ou impossible, l'outil numérique atteignant ses limites, le nombre de paramètres utilisés impliquant des temps de calculs prohibitifs, les relations entrant en jeu entraînent des instabilités numériques, etc.... Une idée naturelle, face à la complexité, consiste à décomposer le problème afin d'identifier des sous-problèmes plus simples à résoudre. La complexité se situe alors dans la nature des interactions entre les différents composants dont il faut contrôler la stabilité. Cette complexité

est alors considérée comme une propriété émergente issue d'un échange d'informations entre composants autonomes.

Un ensemble d'outils et de concepts ont ainsi été développés. Dans le domaine de l'informatique, l'intelligence artificielle (I.A.) a évolué vers l'intelligence artificielle distribuée (I.A.D.) puis vers les systèmes multi-agents, et dans le domaine de la mécanique, les techniques de sous-structuration statiques et dynamiques (analyse modale) ou les techniques de parallélisation pour les méthodes numériques se sont développées.

I.2.2 Propriétés

Une modélisation modulaire permet, en plus de la gestion de la complexité, de conférer au modèle des propriétés nouvelles.

Pour la plupart des logiciels, l'interaction n'intervient qu'au niveau graphique au moment de la construction des pièces et des assemblages. Le système est donc fermé et en général adapté à une seule classe de problème. Cependant, un modèle global construit par assemblage de modules élémentaires peut être modifié très simplement par le remplacement d'un de ces modules. Le modèle est flexible et peut donc être modifié au cours de la simulation.

Ces modules, auxquels sont affectées des tâches locales spécifiques, peuvent alors être distribués sur plusieurs processeurs ou ordinateurs d'un même réseau afin de partager des ressources ou des compétences distantes.

I.2.3 Bilan

Pour les logiciels présentés, il n'existe pas de possibilités d'une vraie interaction en dynamique. Une manipulation directe et intuitive de la maquette digitale n'est pas possible. Ainsi, même si les logiciels sont définis comme interactifs, ils ne le sont en fait que pour la phase de construction des pièces et des mécanismes. Les phases d'assemblage et de désassemblage sont de ce fait très peu étudiées. Dans l'état actuel des développements, les logiciels de simulation accordent donc une grande importance à la réalisation fine des pièces à assembler au travers de modeleurs élaborés mais n'incluent pas la phase préliminaire du processus de conception pour s'affranchir de l'ébauche sur papier.

La modélisation modulaire laisse envisager une possibilité d'étude en amont pour réaliser des assemblages et des manipulations intuitives de maquettes virtuelles. Le système multicorps construit par assemblage d'entités élémentaires peut facilement être modifié par

suppression ou remplacement d'une de ces entités. La modularité fait ainsi gagner en flexibilité.

Dans une phase avancée, elle facilite l'intégration d'un nouveau composant sans qu'il soit nécessaire d'effectuer une ré-analyse du modèle complet. Elle préserve aussi l'intégrité et le secret de la modélisation de chaque composant [Wan03].

Toute la difficulté de ces méthodes réside alors dans la connexion des solutions modulaires sans altérer la cohérence du système global.

Pour que l'outil soit de plus interactif, il nécessaire que l'organisation de la résolution soit différente de celle classique présentée sur la figure I-1 et tende vers une architecture proche de celle de la figure I-2.

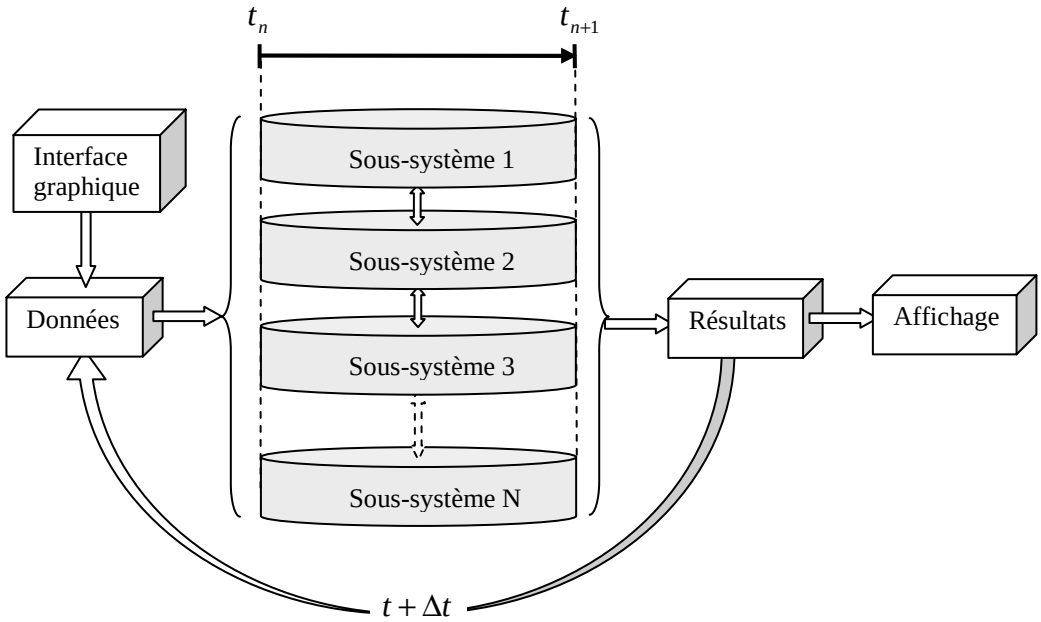


Figure I-2 : Approche modulaire et interactive

Sous cette forme, la résolution ne s'effectue que sur un pas de temps $\Delta t = t_{n+1} - t_n$ et de façon indépendante pour chacun des sous-systèmes 1 à N. Au travers de l'interface, qui se trouve dans la boucle de résolution, l'opérateur peut agir sur la scène:

- en modifiant les relations entre sous-systèmes ou en permettant de modifier les relations entre les sous-systèmes.
- en appliquant de nouvelles contraintes cinématiques, contraignant ainsi les déplacements afin de simuler par exemple la saisie d'un objet par l'opérateur.
- en exerçant des efforts sur les éléments du système.

Il est à noter que des modifications peuvent également être apportées:

- par détection de collision et donc ajout de contraintes cinématiques supplémentaires.
- par calcul d'efforts limites dans les articulations afin de simuler la coupure éventuelle d'une liaison.

Ce type de modélisation modulaire permet de tirer profit de l'évolution de l'architecture parallèle des ordinateurs. Chaque tâche ou groupement de tâches indépendantes peut être ainsi distribuée à des processeurs différents afin d'améliorer les temps de calcul. De manière plus visionnaire, avec l'avènement des réseaux informatiques, il est possible d'envisager qu'au travers de ces ressources physiquement distribuées, des portions de modèles de mondes virtuels soient connectées (Figure I-3).

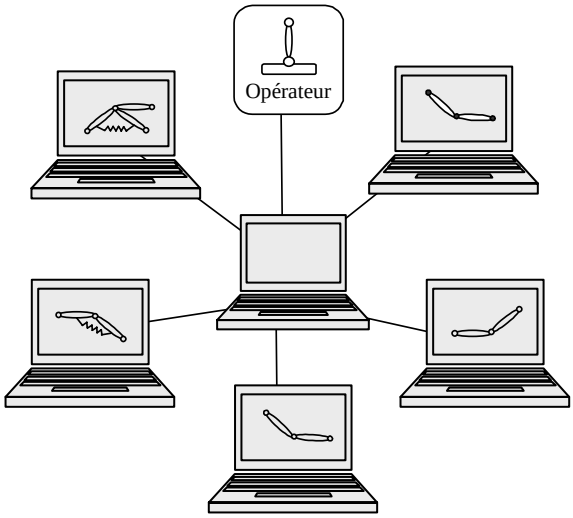


Figure I-3 – Modèle modulaire résolu de manière distribuée

L'existence du méta monde virtuel ainsi créé ne repose alors pas uniquement sur la définition des entités élémentaires qui peuvent être hétérogènes dans leur forme ou dans leur modèle mais également sur l'identification et la forme des interactions mises en jeu.

I.3 Exemples d'approches modulaires

De nombreux domaines de la mécanique utilisent des approches modulaires pour la résolution de problèmes complexes. Sont répertoriées ici quelques techniques répondant aux critères généraux suivants:

- une décomposition possible du problème global en problèmes plus simples.
- une identification d'entités élémentaires auxquelles sont affectées des tâches ou des comportements. Ces entités ont alors toute forme possible: virtuelles, physiques, mathématiques…
- l'union des résolutions produites par chacune des entités (liens, contraintes, dialogues,…) permet de résoudre le système global.

I.3.1 Méthodes de sous-structuration

En calcul des structures, les pièces à étudier sont maillées afin de mettre en œuvre des méthodes éléments finis [Imb95]. La structure peut alors être organisée en plusieurs niveaux. Des sous-structures sont formées à partir des éléments de base, les nœuds, puis elles-mêmes assemblées pour former d'autres structures, etc… pour constituer enfin la structure globale de la pièce. Pour chaque sous-structure, les nœuds internes sont différenciés des nœuds externes également appelés nœuds de liaison. Ce type de décomposition en sous-structures présente plusieurs avantages:

- des vérifications de modèles peuvent être réalisées par sous-structures, ce qui sur le plan pratique, permet de distribuer le travail à des personnes ou ordinateurs différents.
- toute modification d'une sous-structure n'est que locale et n'affecte donc pas la structure globale.
- pour des sous-structures identiques, l'analyse peut n'être effectuée qu'une seule fois entraînant une diminution des temps d'étude.

Une fois cette décomposition réalisée, la résolution peut être effectuée de deux manières:

- par intégration directe du système différentiel.
- par projection sur une base de modes propres.

Les différentes méthodes de synthèse modale proviennent des différents choix de modes de sous-structures et de procédures de couplage qui définissent les chargements et les déplacements des nœuds d'interfaces.

[Cip81a], [Cip81b] et [Kim99] ont appliqués ce type de méthode pour des systèmes multicorps.

I.3.2 Programmation par contraintes

Ce domaine aborde la résolution de problèmes dont les différentes entités élémentaires sont décrites par un ensemble de variables liées par des relations appelées contraintes. Deux types de problèmes distincts sont traités:

- la satisfaction de contraintes qui consiste à trouver un état des variables vérifiant les contraintes.
- le maintien de contraintes qui est la recherche des transformations continues du système faisant passer d'un état initial cohérent (respectant les contraintes) à un état final les respectant également.

Pour illustrer par une analogie mécanique, la satisfaction de contraintes peut être vue comme la recherche de conditions initiales ou également le problème de la conception. Le maintien de contraintes correspond au problème de cohérence au cours de l'évolution temporelle du système, c'est le problème de la simulation.

Un des domaines d'application est celui des interfaces graphiques interactives. Le but est de structurer un ensemble de composants graphiques élémentaires liés par des relations. De même, la construction de pièces ou d'assemblages dans les modeleurs 3D (Ideas, SolidWorks,…) résulte de la construction de relations entre entités géométriques [Doh95].

Les solutions classiques sont les méthodes de backtracking ('générer - tester') ou de propagation de contraintes [Kum92]. La technique 'générer - tester' s'applique à des systèmes décrits par un ensemble de variables appartenant à des domaines finis. Comme son nom

l'indique, elle consiste tout d'abord à générer l'ensemble des combinaisons de variables puis à tester celles vérifiant les contraintes. C'est une méthode dont le coût numérique est très important puisque toutes les combinaisons possibles sont calculées alors que seules quelques-unes respectent les contraintes. Dans les techniques de propagation de contraintes, l'espace de recherche est diminué à l'aide de l'étude du graphe des contraintes. Des techniques hybrides ont également été développées car la méthode de propagation ne permet pas dans certains cas d'effectuer une résolution complète. Dans le cas des domaines continus, les méthodes de Newton, de relaxation de Newton-Raphson, du simplexe restent les plus classiques.

Ces différentes approches offrent peu d'interactivité et la définition du système est statique. Graps [Bel97] propose une méthode distribuée de résolution par propagation de gradients afin de répondre à l'aspect dynamique du processus de construction et d'animation d'interfaces graphiques interactives.

I.3.3 Parallélisation

Les techniques de parallélisation sont des méthodes de décomposition des calculs. Le but est de reformuler les problèmes afin de dégager un certain nombre de calculs pouvant être réalisés indépendamment et simultanément [Fis98]. Ce sont des sous-tâches, chacune affectée à un processeur afin d'accélérer les temps de résolution. La parallélisation peut être réalisée à trois niveaux [Cog97]:

- au niveau **formalisme**, elle consiste, au moment de l'obtention des équations de la dynamique, à trouver des méthodes parallèles pour calculer les éléments des équations de la dynamique, par exemple la matrice d'inertie ou son inverse.

- par la **topologie**. Le but est de tirer partie de la géométrie parallèle du système mécanique et à le décomposer en sous-systèmes formant des chaînes parallèles.

- par le **code**. La technique consiste à paralléliser le graphe de tâche directement produit par le code, indépendamment de ce qu'il représente. Ce graphe est composé de sommets et d'arcs représentant respectivement les unités de calculs et les relations entre les données. La partition est alors l'affectation de groupes d'arcs à des processeurs différents. Les arcs traduisent alors les communications entre les processeurs. Les différentes variantes de cette méthode sont utilisées dans de nombreux domaines : la parallélisation des calculs associés aux éléments-finis, aux multiplications de matrices et de vecteurs, à la simulation de réseaux de neurones,...[Hen00]. Les travaux menés ont

pour but l'étude des techniques de répartition des calculs mais également l'évaluation des échanges entre processeurs afin de minimiser les échanges et les temps de communication entre processeurs.

[Fij95] présente une vue d'ensemble de ces méthodes dans le cadre de la dynamique des systèmes multicorps. Les études ont été menées dans le but d'améliorer l'efficacité de l'obtention des équations de la dynamique, qui dans le cas de la dynamique peut se résumer au calcul de la matrice d'inertie d'un système ou de son inverse. Les améliorations se sont traduites par la diminution sensible du nombre d'opérations nécessaires aux différents calculs. De $O(N^3)$ jusqu'à $O(N^2)$ ([Wal82]) puis O(N) ([Fea83]), ces premiers types de calculs ont produit des algorithmes séries pour lesquels il a été prouvé que la limite asymptotique est en O(N) pour un système à N corps. L'utilisation des méthodes parallèles, favorisée par les progrès techniques sur les processeurs, est apparue comme une voie possible d'amélioration [Bae87c], [Lee88], [Mcm92],[And98]. Ce nouveau type d'algorithmes est moins efficace sur le plan de la complexité (ou du nombre d'opérations) mais leur haut degré de parallélisation assure des calculs plus rapides, une fois distribués sur plusieurs processeurs. Ces algorithmes font l'objet de développements propres: les plus efficaces d'entre eux ne proviennent en effet pas d'une mise en parallèle d'algorithmes séries.

Citons entre autres études dans cette direction l'algorithme de complexité en O(Log(n)), 'Divide & Conquer' ([Fea 99a] et [Fea99b]) qui consiste à construire l'équation de la dynamique par assemblage successif deux à deux de corps simples. C'est une méthode de construction des équations de la dynamique du système multicorps. La résolution ne tire quant à elle aucun partie de la construction parallèle. L'algorithme parallèle CFA (Constraint Force Algorithm) développé par [Fij95] est également d'une complexité en O(Log(n)).

I.3.4 Représentation par blocs

Cette approche de la modélisation des systèmes multicorps est basée sur une représentation par blocs de la structure articulée [Küb99], [Küb00]. Le but est ici également d'apporter des propriétés de flexibilité dans la construction des mécanismes afin de passer très simplement d'un modèle à un autre. Chaque bloc symbolise un système multicorps (Figure I-4) éventuellement constitué d'un unique corps.

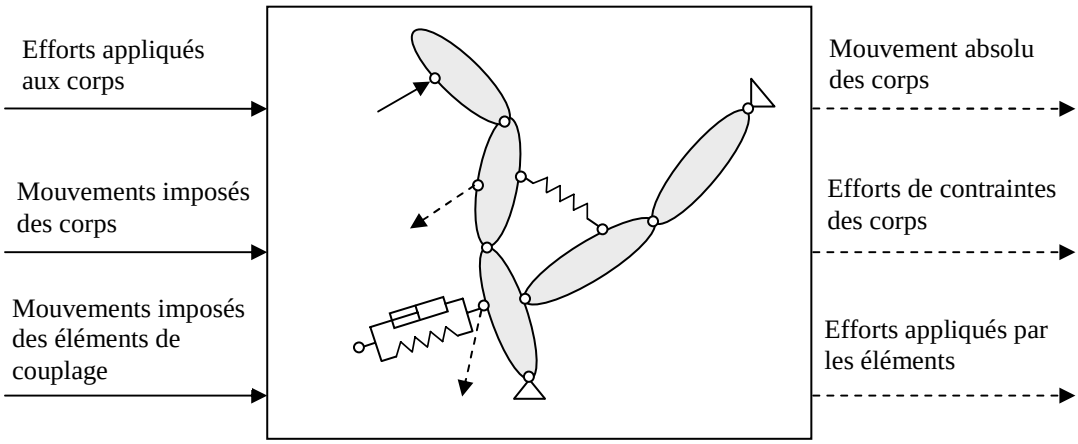


Figure I-4 – Exemple de représentation d'un système multicorps

Il est représenté schématiquement par un bloc de la forme suivante :

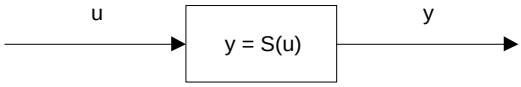


Figure I-5 – Schéma bloc d'un système multicorps

où :

- u est un vecteur d'entrée formé:

  - du vecteur des forces extérieures appliquées au système.

  - des vecteurs position et vitesse des corps du système dont les déplacements sont imposés.

  - des vecteurs position et de vitesse des éléments de couplage (ressorts, amortisseurs).

- y est un vecteur de sortie formé:

  - du mouvement absolu des corps (position, vitesse et accélération).

  - des efforts de contraintes associées aux positions et vitesses imposées.

  - des efforts de réaction appliqués par les éléments de couplage.

- S est l'ensemble des équations différentielles décrivant le comportement du système.

Le système global résulte de la connexion de ces blocs élémentaires communicants au travers de ces quantités identifiées en entrées et en sorties. Ces interconnections sont représentées par des équations de couplage. u et y sont définis par:

$$u = \begin{bmatrix} u^1 \\ : \\ u^N \end{bmatrix} \quad \text{et} \quad y = \begin{bmatrix} y^1 \\ : \\ y^N \end{bmatrix} \qquad (I.1)$$

et sont liés par:

$$u = L \cdot y \quad \text{où} \quad L_{ij} \in \{0,1\} \qquad (I.2)$$

La construction d'un système multicorps peut donc par exemple se mettre sous la forme d'un schéma bloc représenté sur la Figure I-6.

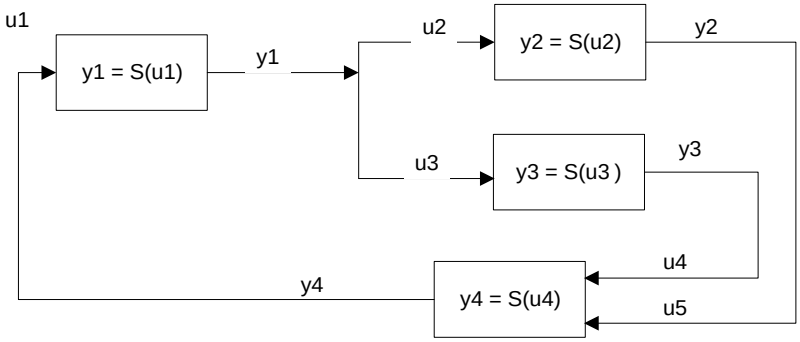


Figure I-6 – Exemple de couplage de sous-systèmes

Le formalisme adopté se prête bien à la modélisation de système articulés sans boucles cinématiques car le traitement des informations (entrées – sorties) s'effectue explicitement et simultanément après intégration numérique de chaque composant. Dans le cas de boucles cinématiques, il est nécessaire d'envisager un calcul implicite de ces informations afin d'éviter des problèmes de stabilité numérique obligeant alors à centraliser toutes les relations algébriques traduisant le couplage entre ces entrées et ces sorties.

Les sous-systèmes peuvent ainsi être modélisés ou modifiés indépendamment. Un outil de simulation modulaire tel que SIMULINK peut alors être utilisé pour assembler les différents blocs et constituer le système global. Cette méthode ne permet pas, dans l'état actuel, d'utiliser des méthodes de résolution différentes pour la résolution dynamique de chacun des blocs du système.

I.3.5 Algorithme de *'gluing'*

Tout aussi récemment, de nouvelles techniques basées sur une approche modulaire de la modélisation dynamique ont été définies dans [Tse01].

Dans cette approche, le modèle est déjà distribué en plusieurs composants développés au sein de différentes équipes de simulation. L'algorithme de *'gluing'* consiste alors à simuler l'ensemble des composants tout en maintenant la cohérence de l'ensemble du modèle. La contrainte imposée est de ne rien connaître des modèles développés au sein de chaque composant pour des raisons de confidentialité ou de complexité (principe du fonctionnement en boite noire).

L'idée est d'élaborer un élément d'interface entre deux composants qui assure la coordination des données au niveau de la liaison. Plusieurs stratégies sont alors possibles en fonction du type d'informations échangées entre cet élément de coordination et les modèles. En notant, $\boldsymbol{X}$ le vecteur des quantités cinématiques et $\boldsymbol{T}$ le vecteur des efforts à l'interface des modèles qui doivent être assemblés, les méthodes suivantes sont définies :

- méthode $\boldsymbol{T}$ - $\boldsymbol{T}$ (Figure I-7) : l'élément de coordination calcule le vecteur des efforts de réaction $\boldsymbol{T}$ qui minimise les violations de contraintes, aussi nommées conditions de compatibilité. C'est cette information $\boldsymbol{T}$ réactualisées qui est transmise à chaque modèle pour le pas de temps suivant.

- méthode $\boldsymbol{X}$ - $\boldsymbol{X}$ (Figure I-8) : l'élément de coordination calcule le vecteur des quantités cinématiques $\boldsymbol{X}$ qui minimise l'erreur sur le non-équilibre des efforts de réaction. Le résultat est transmis aux modèles en liaison pour le pas de temps suivant.

- méthode $\boldsymbol{X}$ - $\boldsymbol{T}$: c'est une méthode mixte où l'élément de coordination transmet pour un modèle l'information d'effort $\boldsymbol{T}$ et pour l'autre l'information sur les quantités cinématiques $\boldsymbol{X}$.

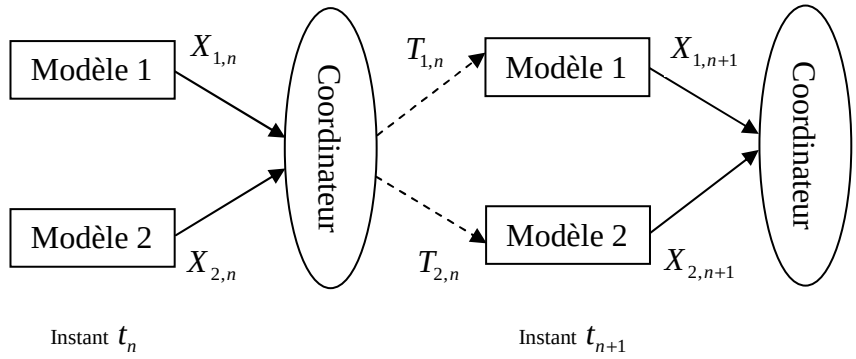


Figure I-7 – Méthode ***T*-*T***

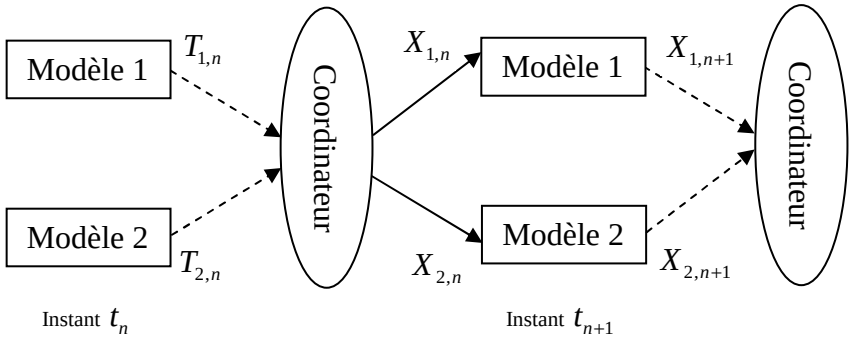


Figure I-8 – Méthode ***X* - *X***

Pour ces méthodes, la définition du coordinateur joue un grand rôle. La mise en œuvre de la méthode ***T* - *T*** pour deux composants, exposées dans [Wan03], est décrite dans la suite. Elle consiste à déterminer, dans un premier temps, pour chaque composant, la matrice de flexibilité locale K_1 et K_2 aux nœuds de la liaison et de déterminer ensuite la matrice de raideur (matrice de *'gluing'*) de la liaison.

Cette matrice permet ensuite par une procédure itérative de calculer les efforts de réaction. Soient *F,* le vecteur suffisant pour définir les efforts à l'interface et *e* une mesure des violations de contrainte.

L'erreur *e* peut prendre la forme générale suivante :

$$e = e(F) \qquad \text{(I.3)}$$

Puisque $e = 0$ indique que les conditions de compatibilité sont vérifiées, le but de l'algorithme est de faire tendre e vers zéro. Un développement limité de e au premier ordre conduit à :

$$e \approx e(F^{(i)}) + \left(\frac{\partial e}{\partial F} \right)_{F=F^{(i)}} (F - F^{(i)}) = 0 \qquad (I.4)$$

La résolution par une méthode de Newton-Raphson permet de calculer *F* par la formule itérative suivante :

$$F^{(i+1)} = F^{(i)} + \Lambda(-e^{(i)}) \qquad (I.5)$$

avec

- $\Lambda = \left(\frac{\partial e}{\partial F} \right)^{-1}_{F=F^{(i)}}$, la matrice de *'gluing'* ou matrice lambda
- $e^{(i)} = e(F^{(i)})$

La matrice de flexibilité de chaque composant servant au calcul de la matrice de raideur de l'interface peut être obtenue numériquement en calculant la réponse de chaque composant à des forces unités imposées suivant la direction des contraintes algébriques et appliquées aux nœuds de la liaison. Il n'est alors pas nécessaire de connaître le modèle interne de chaque composant.

Cette méthode s'applique bien à des problèmes de statique ou de dynamique linéaire mais nécessite par contre des ajustements dans sa mise en œuvre pour des problèmes de dynamique non-linéaire même pour des exemples aussi simples et élémentaires qu'un double pendule. En effet, la matrice de raideur (matrice de *'gluing'*) peut être mal conditionnée autour de positions dites singulières et un nombre d'itérations important peut s'avérer nécessaire avant convergence à cause de fortes non-linéarités géométriques.

I.3.6 Méthodes impulsionnelles

Développée par Mirtich [Mir96] en extension des travaux de [Hah88], cette méthode, issue du domaine de la dynamique moléculaire, a été développée pour des systèmes prédominés par des interactions de type collision dont l'exemple caractéristique est le comportement de pièces contenues dans un bol vibrant. Elle est donc particulièrement bien adaptée à des systèmes dont les corps sont soumis à des contacts unilatéraux. L'extension au traitement de contacts bilatéraux est toutefois possible.

Le principe consiste, à l'aide d'un détecteur de collision, à calculer pour la scène, le temps minimum avant le prochain impact entre deux solides. Entre deux collisions, les corps sont considérés libres et ont donc, dans le champ de pesanteur une trajectoire balistique. La dynamique de chaque corps peut donc être résolue indépendamment laissant alors la possibilité de paralléliser les calculs. Au moment d'une collision, une impulsion $\vec{p}$ est appliquée au premier solide et par réaction une impulsion $-\vec{p}$ au deuxième, impulsion qui dépend de la nature du contact défini entre les solides.

Ses particularités concernent tout d'abord la détection de collision. La première hypothèse porte sur la non-pénétration des corps rigides en contact. Cette détection, qui s'effectue classiquement à des instants discrets, est réalisée ici de manière continue supprimant ainsi les problèmes de pénétration ou d'éventuels ratés de collision. Elle permet de plus de prendre en compte l'évolution de la simulation telles que des modifications du type de contact (glissement, adhérence) ou de modèles de contact. Les différents types de contacts sont en effet traités à l'aide d'une unique méthode.

L'emploi d'impulsions permet également d'éviter les problèmes de stabilité numérique liés aux méthodes de pénalité qui nécessitent l'introduction de ressorts de forte raideur et conduisent à des problèmes raides, mal conditionnés. La difficulté réside aussi dans le choix des caractéristiques physiques de ces éléments de liaison.

Elle est employée dans le cadre de systèmes multi-corps rigides de taille modérée, le but étant de réaliser des simulations à une vitesse interactive c'est-à-dire proche du temps-réel. Les performances recherchées doivent donc conduire à une précision suffisante pour produire une animation réaliste mais possédant malgré tout un bon pouvoir prédictif.

Cette méthode possède des limites dans certaines applications bien particulières. Parmi les limitations, retenons l'exemple d'un système constitué d'une superposition de blocs à

l'équilibre, de même, que celui d'un solide posé sur une rampe même soumis à du frottement sec. Sous l'effet des impulsions, les corps ne restent dans ce cas pas à l'équilibre. De plus, les travaux présentés n'évoquent à l'heure actuelle pas de notion d'interactivité.

La méthode est donc moins efficace que les méthodes de pénalité pour des contacts prolongés ou simultanés. Une interrogation pourrait également être formulée sur le coût numérique et la stabilité liés au traitement de liaisons bilatérales par une succession de chocs. La méthode est donc adaptée à un type de problème bien particulier et le traitement de problèmes plus généraux requiert alors l'utilisation de méthodes mixtes, impulsionnelles et de pénalité.

I.4 Conclusion

Divers domaines utilisent donc des techniques modulaires pour gérer un grand nombre de paramètres ou coupler des systèmes distribués.

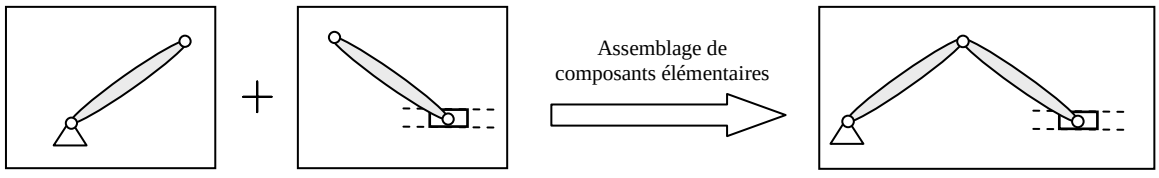
Les techniques de parallélisation visent à organiser les calculs afin de les répartir sur plusieurs processeurs et donc d'accélérer les temps de résolution. Mais ce qui prédomine dans la plupart des méthodes exposées, c'est la création d'entités indépendantes pour lesquelles il faut définir les interactions. Le but est d'apporter plus de flexibilité dans la construction des modèles ou de leur résolution. L'idée est de gérer ces entités en boites noires pour des raisons de confidentialité (algorithme *'gluing'*) ou pour leurs possibilités de réutilisation comme dans le cas de la représentation bloc ou pour les méthodes de sous-structuration.

Cette dernière méthode est utilisée en calcul des structures et fait appel à des techniques de décomposition de domaines associées à des techniques de résolution en effort et/ou en déplacement au niveau des interfaces. Si l'intégration numérique des composants peut s'effectuer indépendamment sur un pas de temps, la résolution au niveau des interfaces se fait de manière globale. Les composants ne sont alors plus autonomes mais dépendent d'un processus centralisé. Pour les systèmes multicorps constitués de corps rigides, l'application d'une telle méthode, qui pose de toutes façons des problèmes de mise en œuvre, est beaucoup moins avantageuse. Dans le cas de structures déformables, le nombre de contraintes algébriques introduites reste en effet faible devant le nombre de degrés de liberté du système alors que dans le cas de systèmes multicorps, ils sont du même ordre de grandeur. Le coût numérique du processus centralisé pour le calcul des efforts aux interfaces peut être plus

important que le coût numérique de l'intégration des équations de mouvement de tous les composants.

Pour notre étude, le but est donc de construire un système multicorps comme un assemblage d'entités indépendantes directement liées aux modèles physiques, corps ou sous-systèmes, tels que dans la représentation par blocs ou dans les méthodes de *'gluing'*. L'idée d'éléments d'interface, absente de la méthode par blocs de Kübler mais définie dans les méthodes de *'gluing'*, assurant la cohésion de l'ensemble des sous-systèmes pour former le système global, est intéressante car elle permet de rattacher directement ces éléments aux liaisons physiques. Cette formulation augmente encore la flexibilité, qui ne concerne donc plus seulement les éléments eux-mêmes, mais également les relations et rejoint en cela la méthode impulsionnelle de [Mir96]. Elle est, en effet, apte à traiter tout type de contact, c'est-à-dire, tout type de relation entre entités élémentaires.

La solution que nous proposons de développer dans le chapitre 4 consiste alors à séparer le traitement des liaisons du reste de la résolution rejoignant en cela le principe général de traitement des systèmes évolutifs présentés dans [Bar93]. Ces efforts de liaison sont définis de manière explicite à l'aide de fonctions de pénalité. La solution est simple et nécessite uniquement une solution locale au niveau de l'interface. Il n'y a plus de processus centralisé. La résolution est rapide et permet d'envisager l'interactivité en cours de simulation. La difficulté réside alors dans l'identification des paramètres associés à ces fonctions (paramètres de pénalité) afin d'assurer la stabilité numérique en fonction du pas de temps de simulation, sans qu'il soit nécessaire d'avoir une connaissance exhaustive des composants connectés. Dans notre approche, la possibilité d'assembler plusieurs composants pour construire un nouveau composant est conservée. Dans ce cas, la résolution au niveau des interfaces, devenues internes à l'élément, pourra être effectuée suivant un processus centralisé à l'intérieur même du nouveau composant Figure I-9.



a) Calcul explicite des efforts de liaisons

b) Calcul implicite des efforts de liaisons

Figure I-9 – Formes centralisée et modulaire d'un système bielle – manivelle

Pour illustrer ce principe, les exemples présentés utilisent une modélisation des solides à l'aide des coordonnées naturelles [Gar86], c'est-à-dire par l'intermédiaire de deux points et deux vecteurs, soit douze coordonnées. Ainsi, un solide est un composant défini par quatre composants élémentaires devant satisfaire six relations algébriques de contraintes internes.

Pour la résolution de ces contraintes, notre choix s'est porté sur la méthode itérative de Bayo [Bay88] basée sur une formulation du lagrangien augmenté. Cette méthode à l'avantage de traiter des problèmes de singularité, de converger très rapidement et d'avoir un formalisme très proche du traitement des contraintes externes à l'aide de fonctions de pénalité.

CHAPITRE II

Modélisation dynamique

II.1 Introduction

Le domaine de la dynamique des systèmes multicorps a connu de grandes avancées dans les deux dernières décennies. Le challenge de la simulation temps-réels, favorisé par l'explosion des capacités des ordinateurs personnels, ont poussés les études aussi bien sur la génération automatique des trajectoires que sur l'intégration numérique des équations du mouvement.

Ces études sont à l'interface des domaines suivants :

- la simulation
- l'animation
- la dynamique interactive
- les interfaces haptiques
- le prototypage rapide
- les interfaces graphiques interactives

Ces différents domaines présentent des problématiques communes : reproduire virtuellement des comportements physiques avec plus ou moins de précision, de réalisme, d'immersion dans le monde virtuel. La simulation concerne les domaines de l'ingénierie ou de la recherche. Le but est de produire une évolution temporelle fidèle de la scène modélisée avec une grande précision à partir des lois de la physique. A l'extrême opposé, les concepteurs d'animations créent une succession d'images virtuelles agencées de manière à donner l'illusion de la réalité comme pour la technique des dessins animés. Des différences apparaissent également par leur niveau d'interactivité avec l'utilisateur. De simple spectateur face à une animation ou un résultat de simulation classique, il devient acteur devant un outil de prototypage rapide ou une interface graphique interactive (IGI) et est immergé dans le cas des interfaces haptiques.

Dans ces domaines, les principales avancées ont été motivées par le désir de réduire significativement les temps de traitement avec comme but ultime la possibilité d'effectuer les calculs en temps-réels.

Ce chapitre présente de manière générale les principales méthodes utilisées en dynamique des systèmes multicorps. Il n'a toutefois pas la prétention de faire une présentation détaillée et

exhaustive mais donne une vue d'ensemble du domaine constituant ainsi une introduction au chapitre suivant plus particulièrement dédié à la problématique de la modélisation modulaire.

II.2 Généralités

La dynamique des systèmes multicorps trouve ses fondements sur les éléments de la mécanique classique. C'est un domaine largement étudié pour lequel il existe de nombreux articles et ouvrages de référence [Spo89],[Sha01], [Sch97],…

Effectuer la simulation d'un système, que ce soit d'un système multicorps ou autre, consiste à calculer l'évolution temporelle des grandeurs caractéristiques qui le décrivent. Ce travail s'effectue en trois étapes:

- **la modélisation** est l'identification ou le choix des paramètres représentatifs, mis en place sous certaines hypothèses et liés par des équations traduisant les lois de la physique. Ce modèle prend alors généralement la forme de systèmes d'équations différentielles ordinaires, d'équations aux dérivées partielles ou d'équations algébro-différentielles.
- **la résolution** : dans la plupart des cas, il n'est pas possible de trouver une solution analytique de ces systèmes. La résolution consiste alors, en utilisant les moyens informatiques, à trouver les positions, vitesses, accélérations et efforts à chaque instant de la simulation grâce à des méthodes numériques. Cette phase de résolution passe éventuellement aussi par des transformations sur les formes mathématiques obtenues lors de la modélisation, généralement des équations différentielles, afin d'obtenir des propriétés de stabilité ou de précision.
- **l'analyse des résultats** qui se présentent sous la forme de simples fichiers de données ou d'animations présentant des rendus plus ou moins réalistes. Leur analyse nécessite également de comparer la précision, la rapidité, la stabilité des méthodes.

Ces étapes sont également nommées pre-processing, processing et post-processing.

Pour la simulation de systèmes multi-corps rigides, il est nécessaire d'effectuer des choix [Füh90] sur :

- les coordonnées nécessaires à la représentation des mouvements du système
- le formalisme servant à construire les équations du mouvement
- la méthode numérique de résolution assurant efficacité, fiabilité et précision.

II.3 Modèles

La simulation nécessite la construction de trois types de modèles [Gil97]:

- le modèle géométrique donne la forme des corps et des assemblages. Il définit la topologie et est utilisé pour la détection de collision.
- le modèle de contact, nécessaire à la ‘résolution’ d'éventuelles collisions.
- le modèle dynamique dont la résolution donne, à chaque instant, l'état du système (position, vitesse, accélération).

II.3.1 Coordonnées

De nombreux paramétrages sont possibles pour représenter la position d’un assemblage de corps rigides. Ce choix est primordial puisqu’il influence directement la structure des équations de la dynamique et donc les méthodes à mettre en œuvre pour résoudre le problème de la dynamique. Il conditionne donc aussi les propriétés de convergence et de stabilité. Deux questions principales doivent alors être posées concernant le nombre et le type de coordonnées, soit :

- le nombre de coordonnées doit-il être minimal ou surabondant ?
- doit-on se repérer par rapport à un référentiel fixe ou mobile, c’est-à-dire, les coordonnées sont-elles absolues ou relatives?

II.3.1.1 Nombre de coordonnées

Le nombre de paramètres utilisés pour la représentation du système doit être au moins égal au nombre de degrés de liberté du système pour pouvoir totalement le définir. Deux cas se présentent alors :

- leur nombre est strictement égal aux degrés de liberté du mécanisme.

- les paramètres sont en nombre surabondant.

Dans le premier cas, les coordonnées sont donc indépendantes. Les équations de la dynamique forment ainsi un système d'équations minimal. Des techniques classiques de résolution d'équations différentielles ordinaires peuvent être mises en œuvre afin de trouver les positions à chaque instant.

Dans le deuxième cas, des équations algébriques de contraintes supplémentaires doivent être introduites. Elles définissent les relations entre les coordonnées dépendantes. Le système à résoudre est donc formé des équations différentielles auxquelles sont ajoutées ces équations algébriques: il est donc de type algébro-différentiel. Il est à noter que les conditions initiales doivent de plus vérifier ces équations de contraintes.

II.3.1.2 Type de coordonnées

II.3.1.2.1 Coordonnées relatives

Chacun des corps est repéré par sa position relative par rapport à un corps auquel il est lié dans la chaîne cinématique. Le paramétrage de Denavit-Hartenberg [Den55] modifié par Khalil [Kha99] est un exemple adapté aux systèmes multicorps qu'ils soient à structure arborescente ou fermée. Ce paramétrage est systématique et adapté à une génération automatique de la géométrie du système multicorps. Ce type de coordonnées prend en compte la nature de la liaison considérée.

Il est à noter que ces coordonnées deviennent dépendantes dans le cas de bouclages cinématiques.

II.3.1.2.2 Coordonnées absolues

La position est définie par rapport à un référentiel galiléen. Il existe plusieurs possibilités de représentation absolue telles que:

- les paramètres définissant classiquement la position et l'orientation d'un repère lié au solide par rapport à un repère fixe. La position peut par exemple être donnée en coordonnées cartésiennes, cylindriques ou sphériques et l'orientation définie par les angles d'Euler, les quaternions, etc…
- les positions cartésiennes absolues de points particuliers du corps affectés des masses reconstituant son inertie. Nikravesh définit de telles coordonnées, les *'point*

coordinates' [Nik94] et donne des exemples d'utilisation. La représentation 4 masses des corps rigides utilisées dans le code Plexus [Jol93] ou les coordonnées naturelles [Gar87], sont d'autres exemples de ce type de coordonnées.

II.3.2 Equations du mouvement

Pour obtenir les équations de la dynamique d'un système mécanique, différents formalismes sont possibles: Newton-Euler, Lagrange, travaux virtuels, …

II.3.2.1 Equations de Newton-Euler

Les équations de Newton-Euler proviennent directement des théorèmes généraux de la mécanique. Pour un système multicorps, construire le modèle dynamique consiste donc à écrire le principe fondamental de la dynamique pour chaque corps de la chaîne cinématique. L'équation associée au corps C_i est donc :

$$\begin{cases} \vec{F}_i = m_i \dot{\vec{v}}_{G_i} \\ \vec{N}_i = \mathbf{J_i} \dot{\vec{\omega}}_i + \vec{\omega}_i \wedge \left(\mathbf{J_i} \vec{\omega}_i \right) \end{cases} \qquad \text{(II.1)}$$

où :

- m_i est la masse du corps C_i.
- $\vec{F}_i$ et $\vec{N}_i$ sont la résultante et le moment au centre d'inertie G_i des forces extérieures appliquées au corps C_i.
- $\vec{v}_{G_i}$ est la vitesse absolue du centre d'inertie du corps C_i.
- $\vec{\omega}_i$ est la vitesse absolue de rotation du corps C_i.
- $\mathbf{J_i}$ est la matrice d'inertie au centre d'inertie du corps C_i.

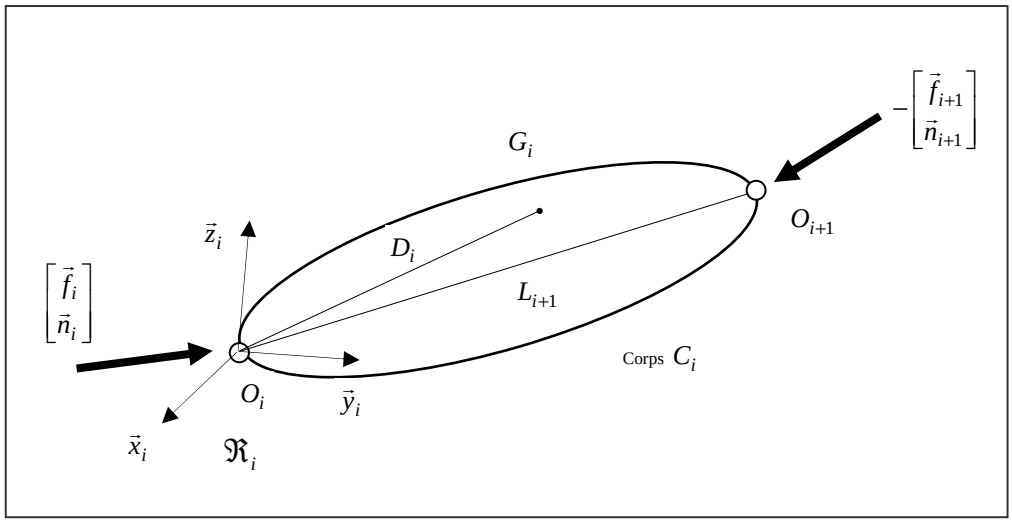


Figure II-1 – Notations pour les équations de Newton - Euler

L'expression de $\vec{F}_i$ et $\vec{N}_i$ peut être obtenue en réalisant un bilan des efforts extérieurs appliqués au corps C_i, conformément aux notations introduites dans la Figure II-1, soit :

$$\begin{cases}\vec{F}_i = \vec{f}_i - \vec{f}_{i+1} + m_i\vec{g} \\ \vec{N}_i = \vec{n}_i - \vec{n}_{i+1} + \left(\vec{D}_i - \vec{L}_{j+1}\right) \wedge \vec{f}_{i+1} - \vec{D}_i \wedge \vec{f}_i \end{cases} \qquad (\text{II.2})$$

avec

- $\Re_i = \left(O_i, \vec{x}_i, \vec{y}_i, \vec{z}_i\right)$ est le repère attaché au corps C_i, $\vec{z}_i$ étant l'axe de l'articulation avec le corps C_{i-1}.
- $\vec{L}_i = \overrightarrow{O_{i-1}O_i}$ est le vecteur formé des origines des repères $\Re_{i-1}$ et $\Re_i$.
- $\vec{D}_i = \overrightarrow{O_iG_i}$ est le vecteur formé par l'origine du repère $\Re_i$ et le centre de gravité du corps C_i.
- G_i est le centre d'inertie du corps C_i.
- $\vec{f}_i$ et $\vec{n}_i$ sont la force et le couple exercés au point O_i par le corps C_{i-1} sur le corps C_i.

Le principe fondamental de la dynamique fait donc intervenir dans l'expression des efforts extérieurs $\vec{F}_i$ et $\vec{N}_i$, l'expression des efforts de liaisons qui sont des quantités inconnues. L'obtention des équations du mouvement et l'élimination de ces inconnues s'effectuent par projections successives sur les axes de mouvement de manière récursive en parcourant l'ensemble des corps de la chaîne cinématique. Nous décrivons brièvement le principe décrit par Bae et Haug [Bae87a] pour une chaîne cinématique simple, le cas des boucles étant traité dans [Bae87b] par les mêmes auteurs.

Le système (II-2) peut s'écrire sous la forme algébrique suivante :

$$M_i \dot{Y}_i - Q_i = 0 \qquad \text{(II.3)}$$

avec :

- $M_i = \begin{bmatrix} m_i & 0 & 0 & \\ 0 & m_i & 0 & \mathbf{0} \\ 0 & 0 & m_i & \\ & \mathbf{0} & & \mathbf{J_i} \end{bmatrix}$
- $Y_i = \begin{Bmatrix} v_{Gi} \\ \omega_i \end{Bmatrix}$
- $Q_i = \begin{Bmatrix} F_i \\ N_i - \vec{\omega}_i \wedge (\mathbf{J_i} \vec{\omega}_i) \end{Bmatrix}$

Entre deux corps C_i et C_j reliés par une articulation (prismatique ou rotoïde), il est possible, en utilisant le principe de composition des vitesses, d'écrire une relation cinématique de la forme :

$$Y_j = B_{ij1} Y_i + B_{ij2} \dot{q}_{ij} \qquad \text{(II.4)}$$

où q_{ij} représente la variable articulaire entre les corps C_i et C_j, B_{ij1} et B_{ij2} étant les deux matrices coefficients, caractéristiques de cette composition.

Cette équation conduit à une relation sur les vecteurs de déplacements virtuels :

$$\partial Z_j = B_{ij1}\partial Z_i + B_{ij2}\partial q_{ij} \qquad (\text{II.5})$$

De même les accélérations du corps C_j sont reliées à celle du corps C_i par :

$$\dot{Y}_j = B_{ij1}\dot{Y}_i + B_{ij2}\ddot{q}_{ij} + D_{ij} \qquad (\text{II.6})$$

avec $D_{ij} = \dot{B}_{ij1}Y_i + \dot{B}_{ij2}\dot{q}_{ij}$.

En appliquant le principe des travaux virtuels pour une chaîne cinématique simple constitué de *n* solides, nous obtenons l'équation :

$$\sum_{i=1}^{n} \partial Z_i^T (M_i \dot{Y}_i - Q_i) = 0 \qquad (\text{II.7})$$

pour tout déplacement virtuel ∂Z_i du corps C_i compatible avec les liaisons.

Cette équation peut-être transformée en utilisant les relations (II-5) et (II-6) entre les corps C_n et C_{n-1} :

$$\begin{aligned}&\sum_{i=1}^{n-1} \partial Z_i^T (M_i \dot{Y}_i - Q_i) + \partial Z_{n-1}^T (B_{(n-1)n}^T (M_n B_{(n-1)n1} \dot{Y}_{n-1} + M_n B_{(n-1)n2} \ddot{q}_{(n-1)n} \\ &+ M_n D_{(n-1)n} - Q_n)) + \partial q_{(n-1)n}^T (B_{(n-1)n2}^T (M_n B_{(n-1)n1} \dot{Y}_{n-1} + M_n B_{(n-1)n2} \ddot{q}_{(n-1)n} \\ &+ M_n D_{(n-1)n} - Q_n)) = 0\end{aligned} \qquad (\text{II.8})$$

Cette équation est vraie quel que soit $\partial q_{(n-1)n}$, indépendamment des déplacements virtuels des corps qui précèdent dans la chaîne cinématique. Ceci conduit donc à l'égalité :

$$B_{(n-1)n2}^T (M_n B_{(n-1)n1} \dot{Y}_{n-1} + M_n B_{(n-1)n2} \ddot{q}_{(n-1)n} + M_n D_{(n-1)n} - Q_n) = 0 \qquad (\text{II.9})$$

$$\ddot{q}_{(n-1)n} = (B_{(n-1)n2}^T M_n B_{(n-1)n2})^{-1} (Q - B_{(n-1)n2}^T (M_n B_{(n-1)n1} \dot{Y}_{n-1} + M_n D_{(n-1)n}) \qquad (\text{II.10})$$

Tenant compte de cette égalité et des relations cinématiques entre le corps C_{n-2} et C_{n-1}, l'équation (II-8) devient :

$$\sum_{i=1}^{n-2} \partial Z_i^T (M_i \dot{Y}_i - Q_i) + \partial Z_{n-1}^T ((M_{n-1} + K_{n-1}) \dot{Y}_{n-1} - (Q_{n-1} + L_{n-1})) = 0 \qquad (II.11)$$

avec :

- $K_{n-1} = B_{(n-1)n1}^T M_n B_{(n-1)n1} - (B_{(n-1)n1}^T M_n B_{(n-1)n2})(B_{(n-1)n2}^T M_n B_{(n-1)n2})^{-1}(B_{(n-1)n2}^T M_n B_{(n-1)n2})$

- $L_{n-1} = B_{(n-1)n1}^T (Q_n - M_n D_{(n-1)n}) + (B_{(n-1)n1}^T M_n B_{(n-1)n2})(B_{(n-1)n2}^T M_n B_{(n-1)n2})^{-1}(B_{(n-1)n2}^T (M_n D_{(n-1)n} - Q_n)$

Il est ainsi possible d'obtenir de manière récursive autant d'équations de la forme (II.10) que d'articulations dans la chaîne cinématique. L'équation obtenue pour le corps C_1 de base permet alors de calculer la première accélération articulaire $\ddot{q}_{01}$, utilisée ensuite dans la deuxième équation pour obtenir $\ddot{q}_{12}$ et ainsi de suite jusqu'au calcul de $\ddot{q}_{(n-1)n}$.

Des algorithmes récursifs particuliers ont été mis en œuvre pour les systèmes arborescents [Arm79], [Bae87a], [Wal82] ainsi que pour les systèmes présentant des boucles cinématiques [Bae87b].

L'avantage de cette approche repose sur l'inversion de système des petites tailles, propriété particulièrement intéressante lorsque le nombre de corps est important.

II.3.2.2 Equations de Lagrange

Les équations de Lagrange peuvent être obtenues à partir du principe des travaux virtuels énoncé comme suit : le travail des quantités d'accélérations absolues est égal au travail virtuel des efforts appliqués au système. Il est noté sous la forme:

$$\delta W_{acc} = \delta W_f \qquad (II.12)$$

où

- δW_{acc} est le travail virtuel des quantités d'accélération absolues.
- δW_f est le travail virtuel des forces appliquées au système.

δW_{acc} s'écrit en fonction de l'énergie cinétique du système :

$$\delta W_{acc} = \left[\frac{d}{dt}\left(\frac{\partial E_c}{\partial \dot{q}} \right) - \frac{\partial E_c}{\partial q} \right]^T \cdot \delta q \qquad \text{(II.13)}$$

et δW_f se compose des trois termes :

$$\delta W_f = \delta W_1 + \delta W_2 + \delta W_3 \qquad \text{(II.14)}$$

où

- $\delta W_1 = F_c^T \cdot \delta q$ est le travail virtuel des forces conservatives F_c. Ces forces conservatives, dérivant d'un potentiel E_p, peuvent se mettre sous la forme $F_c = -\frac{\partial E_p}{\partial q}$. Elles concernent essentiellement les forces dues à la gravité et les forces développées par les contraintes internes dérivant d'un potentiel élastique dans le cas des solides déformables.
- $\delta W_2 = F_{nc}^T \cdot \delta q$ est le travail virtuel des forces non-conservatives F_{nc} appliquées au système.
- $\delta W_3 = F_{in}^T \cdot \delta q$ est le travail virtuel des efforts internes F_{in} lié aux relations algébriques de contraintes internes.
- δq est le déplacement virtuel associé aux coordonnées généralisées q.

Le vecteur q des coordonnées généralisées représente l'ensemble des degrés de liberté du système. Quant aux efforts internes, ils traduisent :

- les liaisons entre les corps qu'elles soient holonomes ou non-holonomes. Dans le cas d'un paramétrage minimal, des efforts de ce type interviennent si le système présente des bouclages cinématiques ou dans le cas de liaisons non-permanentes (type contact).
- les efforts internes aux corps issus d'un paramétrage surabondant.

Ces efforts traduisent en fait une réduction des degrés de liberté du système, exprimés par des équations algébriques de contraintes respectivement holonomes et non-holonomes :

$$\begin{cases} \phi_h(q,t) = 0 & (a) \\ \phi_{nh}(q,\dot{q},t) = 0 & (b) \end{cases} \qquad \text{(II.15)}$$

ou encore par une écriture unique par :

$$\phi(q,\dot{q},t) = 0 \qquad \text{(II.16)}$$

En pratique, les contraintes non-holonomes sont des combinaisons linéaires des vitesses et peuvent donc se mettre sous la forme :

$$\phi_{nh}(q,\dot{q},t) = \psi_{nh}\dot{q} + f(t) = 0 \qquad \text{(II.17)}$$

Le choix d'un champ de déplacement virtuel compatible avec ces contraintes conduit alors à :

$$\begin{cases} \dfrac{\partial \phi_h}{\partial q} \cdot \delta q = 0 \\ \psi_{nh} \cdot \delta q = 0 \end{cases} \qquad \text{(II.18)}$$

L'équation (II.5) conduit donc à :

$$\frac{d}{dt}\left(\frac{\partial E_c}{\partial \dot{q}}\right) - \frac{\partial E_c}{\partial q} = Q - C^T \lambda \qquad \text{(II.19)}$$

où $Q = F_c + F_{nc}$ représente l'ensemble des efforts généralisés.

Ainsi, avec une expression de l'énergie cinétique donnée sous la forme :

$$E_c = \frac{1}{2}\dot{q}^T M(q)\dot{q} \qquad \text{(II.20)}$$

l'équation de la dynamique du système total s'écrit :

$$\begin{cases} M(q)\ddot{q} + R(q,\dot{q},t) = Q - C^T\lambda & (a) \\ \phi_h(q,t) = 0 & (b) \\ \psi_{nh}\dot{q} + f(t) = 0 & (c) \end{cases} \qquad \text{(II.21)}$$

ou encore:

$$\begin{cases} M(q)\ddot{q} + R(q,\dot{q},t) = Q - C^T\lambda \\ \phi(q,\dot{q},t) = 0 \end{cases} \qquad \text{(II.22)}$$

où

- $M(q)$ est la matrice d'inertie du système de dimension $[n \times n]$.
- $R(q,\dot{q},t) = \dot{M}(q)\dot{q} - \frac{1}{2}\dot{q}^T \frac{\partial M}{\partial q}\dot{q}$ est le vecteur des termes quadratiques en vitesse aussi appelés vecteurs des forces centrifuges et de Coriolis.
- $C = \begin{bmatrix} \frac{\partial \phi_h}{\partial q} \\ \psi_{nh} \end{bmatrix}$ est la matrice jacobienne des contraintes holonomes et non-holonomes.
- λ est le vecteur des multiplicateurs de Lagrange.
- $\phi(q,\dot{q},t) = \begin{Bmatrix} \phi_h(q,t) \\ \psi_{nh}\dot{q} + f(t) \end{Bmatrix}$

II.4 Différentes formulation en vue d'une résolution numérique

La dynamique des systèmes multi-corps est donc représentée par un système d'équations différentielles du second ordre, auxquelles sont ajoutées les équations de contraintes. Le système à résoudre est donc de type algébro-différentiel. Les problèmes liés à la résolution de ces systèmes sont multiples et proviennent:

- de la difficulté à déterminer de manière précise les vitesses et positions initiales.
- des contraintes redondantes et des positions singulières.
- des perturbations liées à la résolution (erreurs dues au schéma d'intégration, erreurs d'arrondis).
- de la présence de contacts intermittents ou unilatéraux.

La littérature propose deux types de méthodes pour résoudre ces systèmes. La première classe de méthodes vise à réduire à un nombre minimal de coordonnées la description du système en trouvant un jeu de coordonnées indépendantes. Ce sont les méthodes de **projection** : le système est décrit en *state-space form* ([Füh90], [Cua00]). Dans une deuxième classe de méthodes, des inconnues supplémentaires sont introduites. Le système est alors décrit par une *descriptor form*, et résolue grâce à des méthodes dites d'**augmentation**.

Ces diverses méthodes ont été mises au point pour:

- diminuer les temps de résolution afin de tendre vers des résolutions temps-réel.
- pallier aux instabilités numériques dans le cas de durées de simulations longues.

Le choix de paramétrages minimaux associés à des méthodes récursives conduit à une obtention directe des équations du mouvement en *state-space form* [Arm79], [Wal82], [Fea83], [Bae87a]. Dans le cas de bouclages cinématiques, les coordonnées ne sont plus indépendantes. Il faut utiliser une méthode de projection [Bae87b] ou bien utiliser une méthode de coupure afin de libérer les degrés de liberté dépendants [Wan02]. Dans ce cas, la matrice masse obtenue est dense mais de petite dimension. La méthode proposée par [Hol80] est un exemple de méthode récursive avec formulation lagrangienne.

En *descriptor form* le système à résoudre est de plus grande dimension mais la matrice masse est à structure creuse.

II.4.1 Méthodes de projection

Le but est de réduire le nombre de coordonnées pour en extraire un jeu de coordonnées indépendantes et ainsi résoudre un système de taille minimale égale au nombre de degrés de liberté du système.

Différentes techniques permettent d'identifier de manière automatique ces coordonnées indépendantes : pivot de Gauss, décomposition en valeurs singulières SVD, décomposition LU ou QR, … [Sin85], [Weh82].

Les équations de la dynamique sont résolues pour obtenir ces coordonnées indépendantes. Les coordonnées dépendantes sont ensuite calculées algébriquement à l'aide des équations de contrainte.

II.4.2 Méthodes d'augmentation

Avec ces méthodes, le système à résoudre est formé des équations du mouvement ainsi que des équations de contraintes généralement résolues en une seule étape. Elles sont également appelées méthode d'augmentation car *m* inconnues supplémentaires sont introduites (multiplicateurs de Lagrange, efforts de liaison,…).

L'inconvénient majeur est qu'il faut résoudre un plus grand nombre d'inconnues et trouver des méthodes pour calculer ces inconnues supplémentaires.

II.4.2.1 Méthode de substitution

Le but de la méthode de substitution est d'obtenir une expression des multiplicateurs de Lagrange. Les équations de contraintes holonomes sont dérivées deux fois par rapport au temps, les contraintes non-holonomes une seule fois afin d'obtenir une expression de l'accélération du système.

$$\begin{cases} (\phi_h)_q \ddot{q} = -(\dot{\phi}_h)_q \dot{q} - \dot{\phi}_t \\ \psi_{nh} \ddot{q} = -\dot{\psi}_{nh} \dot{q} - \dot{f}(t) \end{cases} \qquad \text{(II.23)}$$

ou sous la forme matricielle :

$$C\ddot{q} = D \qquad \text{(II.24)}$$

avec

- $C = \begin{bmatrix} (\phi_h)_q \\ \psi_{nh} \end{bmatrix}$, matrice jacobienne des contraintes.
- $D = \begin{Bmatrix} -(\dot{\phi}_h)_q \dot{q} - \dot{\phi}_t \\ -\dot{\psi}_{nh} \dot{q} - \dot{f}(t) \end{Bmatrix}$

Une expression de $\ddot{q}$, obtenue par l'intermédiaire de l'équation (II.14.a), est substituée dans l'équation (II.17) et donne une expression des multiplicateurs de Lagrange :

$$\lambda = \left(CM^{-1}C^T\right)^{-1}\left(CM^{-1}(Q-R)-D\right) \qquad (\text{II.25})$$

Reportée dans l'équation de la dynamique, elles conduisent à une expression des multiplicateurs de Lagrange en fonction de q et $\dot{q}$.

$$M\ddot{q} = (Q-R)-C^T\left(CM^{-1}C^T\right)^{-1}\left(CM^{-1}(Q-R)-D\right) \qquad (\text{II.26})$$

Cette méthode assure un respect strict des équations de contraintes au niveau accélération mais pas aux niveaux position et vitesse entraînant des problèmes de stabilité numérique.

II.4.2.2 Méthode de stabilisation

Afin de corriger les problèmes de stabilité numérique introduit par la méthode de substitution, Baumgarte [Bau72] a proposé une méthode de stabilisation où l'équation de contrainte en accélération (II.16) utilisée dans la méthode de substitution et résumée par :

$$\begin{cases} \ddot{\phi}_h(q,t) = 0 \\ \dot{\psi}_{nh}(q,\dot{q},t) = 0 \end{cases} \qquad (\text{II.27})$$

est remplacée par la nouvelle équation de contraintes :

$$\begin{cases} \ddot{\phi}_h + 2\alpha\dot{\phi}_h + \beta^2\phi_h = 0 \\ \dot{\psi}_{nh} + \gamma\psi_{nh} = 0 \end{cases} \quad \text{(II.28)}$$

Le système à résoudre devient donc:

$$\begin{cases} M(q)\ddot{q} + R(q,\dot{q},t) = Q - C^T\lambda \\ C\ddot{q} = D - 2\alpha\dot{\phi} - \beta^2\phi \end{cases} \quad \text{(II.29)}$$

ou encore sous forme matricielle:

$$\begin{bmatrix} M & C^T \\ C & 0 \end{bmatrix} \begin{Bmatrix} \ddot{q} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} Q - R(q,\dot{q},t) \\ D - 2\alpha\dot{\phi} - \beta^2\phi \end{Bmatrix} \quad \text{(II.30)}$$

Cette méthode est efficace mais il n'existe cependant pas de méthode fiable permettant de déterminer les coefficients de stabilisation α, β et γ quel que soit le problème.

II.4.2.3 Méthode de pénalité

La méthode de pénalité consiste à donner une expression explicite des multiplicateurs de Lagrange comme fonction des violations de contraintes en position et vitesse. Les équations de contraintes sont ainsi éliminées de la formulation.

Les multiplicateurs de Lagrange λ_h et λ_{nh} associés respectivement aux contraintes holonomes et non-holonomes prennent la forme :

$$\begin{cases} \lambda_h = p_h\phi_h + \nu_h\dot{\phi}_h & (a) \\ \lambda_{nh} = p_{nh}\phi_{nh} & (b) \end{cases} \quad \text{(II.31)}$$

où p_h, ν_h, et p_{nh} sont les facteurs de pénalité.

Cela revient à considérer que les liaisons à l'origine des contraintes s'effectuent par l'intermédiaire d'un système ressort-amortisseur dont il faut choisir les caractéristiques. La difficulté est de choisir correctement ces facteurs de pénalité. En théorie, une valeur infinie

traduit le respect strict des équations de contrainte et donc la non-pénétration des corps en liaison. En pratique, les raideurs de liaisons doivent être grandes afin d'éliminer complètement les violations de contraintes mais ce choix conduit à des problèmes de stabilité numérique. Inversement, une raideur trop faible entraîne des violations de contraintes inacceptables.

Cette méthode de pénalité peut être améliorée en introduisant une masse artificielle α dans la direction des contraintes donnant une nouvelle expression des multiplicateurs de Lagrange :

$$\begin{cases} \lambda_h = \alpha_h \left(p_h \phi_h + \nu_h \dot{\phi}_h + \ddot{\phi}_h \right) & (a) \\ \lambda_{nh} = \alpha_{nh} \left(p_{nh} \phi_{nh} + \dot{\phi}_{nh} \right) & (b) \end{cases} \qquad \text{(II.32)}$$

Les équations de mouvement s'écrivent alors :

$$M_\alpha(q)\ddot{q} = Q - \alpha C^T \cdot \tilde{f} \qquad \text{(II.33)}$$

avec

- $\tilde{f} = \begin{Bmatrix} (\phi_h)_q^T \dot{q} + (\dot{\phi}_h)_t + p_h \phi_h + \nu_h \dot{\phi}_h \\ \dot{\psi}_{nh} \dot{q} + \dot{f}(t) + p_{nh} \phi_{nh} \end{Bmatrix}$
- $M_\alpha(q) = M(q) + C^T \alpha C$

Le facteur α est choisi grand afin de réduire les violations de contraintes qui ne peuvent toutefois pas être complètement éliminées. La méthode conduit ainsi à des instabilités numériques pour des temps de simulations longs. Les choix des différents facteurs de pénalité reste de plus hasardeux.

II.4.2.4 Méthode du Lagrangien augmenté

Ces problèmes cruciaux de détermination de facteurs de pénalité peuvent être évités en utilisant la méthode du Lagrangien augmenté [Bay88].

- ***1ère forme***

Elle complète la méthode de pénalité en introduisant des forces de contraintes supplémentaires λ_h^* et λ_{nh}^* :

$$\begin{aligned} \lambda_h &= \lambda_h^* + \left(p_h \phi_h + \nu_h \dot{\phi}_h\right) \\ \lambda_{nh} &= \lambda_{nh}^* + \left(p_{nh} \psi_{nh}\right) \end{aligned} \qquad \text{(II.34)}$$

Avec cette formulation des multiplicateurs de Lagrange, le système contraint à résoudre devient donc :

$$\begin{cases} M(q)\ddot{q} + C^T \lambda = Q - \alpha C^T \hat{f} \\ \phi_h = 0 \\ \phi_{nh} = 0 \end{cases} \qquad \text{(II.35)}$$

avec $\hat{f} = \begin{Bmatrix} p_h \phi_h + \nu_h \dot{\phi}_h \\ p_{nh} \phi_{nh} \end{Bmatrix}$

- ***2ème forme***

Une extension de la méthode consiste à introduire de la masse afin d'obtenir une meilleure stabilité.

$$\begin{aligned} \lambda_h &= \lambda_h^* + \alpha_h \left(p_h \phi_h + \nu_h \dot{\phi}_h + \ddot{\phi}_h\right) \quad (a) \\ \lambda_{nh} &= \lambda_{nh}^* + \alpha_{nh} \left(p_{nh} \psi_{nh} + \dot{\phi}_{nh}\right) \quad (b) \end{aligned} \qquad \text{(II.36)}$$

Le système différentiel devient donc :

$$\begin{cases} M_\alpha(q)\ddot{q} + C^T \lambda = Q - \alpha C^T \tilde{f} \\ \phi_h = 0 \\ \phi_{nh} = 0 \end{cases} \qquad \text{(II.37)}$$

où $\tilde{f}$ et M_α ont l'expression définie au paragraphe II.4.2.3. Cette dernière forme résout également les problèmes de stabilité mais conduit cependant à une inversion de M_α plus coûteuse.

II.4.3 Stabilisation GGL

D'autres méthodes de stabilisation peuvent être mises en œuvre comme par exemple la méthode introduite par [Gea85]. Les équations de contraintes en position et vitesse sont conservées. Les inconnues supplémentaires v et μ sont ajoutées. Ces variables introduisent une nouvelle relation de contraintes. Le système différentiel prend alors la forme :

$$\begin{cases} M(q)\dot{v} = R(q,\dot{q},t) - C^T(q)\lambda & (a) \\ \dot{q} = v + C^T(q)\mu & (b) \\ C(q)v = 0 & (c) \\ \phi(q) = 0 & (d) \end{cases} \qquad \text{(II.38)}$$

Le système (II.31) est d'index 2 par rapport aux variables (p, v, λ, μ). La solution exacte de μ est :

$$\mu = 0 \qquad \text{(II.39)}$$

Dans ce cas, le système possède la même solution que le système (II.15) en (p, v, λ). La solution respecte ainsi les équations de contrainte (II.31.c) et (II.31.d) en position et vitesse. L'inconvénient majeur est cependant le grand nombre d'inconnues supplémentaires qu'il est nécessaire d'introduire et de résoudre.

II.5 Intégration numérique

II.5.1 Généralités

Les équations du mouvement associées à la dynamique des systèmes multicorps rigides sont des systèmes différentiels ordinaires (ODE) ou des systèmes algébro-différentiels (DAE)

suivant le type de paramétrage retenu ou la topologie du mécanisme. La solution analytique de ces problèmes n'est, sauf cas simples et particuliers, pas possible et il est donc nécessaire d'avoir recours à des méthodes numériques afin de trouver l'évolution temporelle des variables donnant l'état du système. Le rôle du schéma numérique est de fournir une approximation de la solution du système différentiel à des instants discrets de l'intervalle de temps considéré.

Ce paragraphe propose un aperçu des méthodes numériques courantes en dynamique des systèmes multicorps mises en œuvre pour les exemples présentés au chapitre V.

Le système différentiel sous sa forme la plus générale peut s'écrire:

$$\begin{cases} F(y, \dot{y}, t) = 0 \\ y(t_0) = y_0 \\ \dot{y}(t_0) = \dot{y}_0 \end{cases} \qquad \text{(II.40)}$$

ou encore

$$\begin{cases} \dot{y} = f(y(t), t) \\ y(t_0) = y_0 \end{cases} \qquad \text{(II.41)}$$

Les grandeurs y_0, y_1, y_2, …, y_{n-2}, y_{n-1}, y_n sont les solutions approchées de la fonction F aux instants t_0, t_1, t_2, … t_n. Ces approximations sont calculées de manière récursive à partir des schémas numériques construits sur la base de développements de Taylor de la fonction. Les schémas obtenus peuvent ainsi être :

- **explicites** : la solution y_{n+1} de l'instant t_{n+1} dépend exclusivement des valeurs y_n, y_{n-1}, y_{n-2}, y_{n-3}, …, y_{n-p} calculées aux $p+1$ instants précédents.
- **implicites** : la solution à l'instant t_{n+1} dépend des valeurs calculées aux instants précédents ainsi que de la valeur à l'instant t_{n+1}.

Si $p = 0$, la méthode est dite à un pas. Si $p > 0$, la méthode est dite à pas multiples ou à $p+1$ pas.

II.5.2 Méthode de Runge-Kutta d'ordre 4

Pour la famille des méthodes de Runge-Kutta, la valeur de la fonction est obtenue par combinaison linéaire de valeurs intermédiaires dans l'intervalle de temps $[t_n, t_{n+1}]$. Le schéma est donc explicite. Pour la méthode de Runge-Kutta d'ordre 4, la relation est:

$$y_{n+1} = y_n + \frac{\Delta t}{6}(k_1 + k_2 + k_3 + k_4) \qquad (\text{II.42})$$

avec

$$\begin{cases} k_1 = f(t_n, y_n) \\ k_2 = f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2} k_1\right) \\ k_3 = f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2} k_2\right) \\ k_4 = f(t_n + \Delta t, y_n + \Delta t k_3) \end{cases} \qquad (\text{II.43})$$

II.5.3 Méthode de Newmark

La méthode de Newmark est définie par le schéma implicite à un pas suivant :

$$\begin{cases} \dot{q}_{n+1} = \dot{q}_n + ((1-\gamma)\ddot{q}_n + \gamma\ddot{q}_{n+1})\Delta t \\ q_{n+1} = q_n + \dot{q}_n \Delta t + \left(\left(\frac{1}{2} - \beta\right)\ddot{q}_n + \beta\ddot{q}_{n+1}\right)\Delta t^2 \end{cases} \qquad (\text{II.44})$$

Le degré 'd'implicité' est contrôlé par la donnée des coefficients γ et β. Pour différentes valeurs, elle permet donc de tester rapidement un certain nombre de schémas numériques classiques.

- Pour $\gamma = \frac{1}{6}$ et $\beta = \frac{1}{4}$, le schéma correspond à une approximation linéaire de l'accélération sur Δt.

- Pour $\gamma = \frac{1}{2}$ et $\beta = \frac{1}{4}$, le schéma est inconditionnellement stable et correspond à une approximation constante de l'accélération égale à la valeur moyenne de l'accélération sur Δt.

Ces résultats de stabilité sont exprimés pour des systèmes sans contraintes. Des problèmes de stabilité apparaissent dans le cas contraire [Ger96]. Quant à la mise en œuvre de la méthode, elle peut s'effectuer par inversion du schéma numérique ou en écrivant la forme incrémentale de l'équation de la dynamique.

II.5.3.1 Inversion du schéma

L'inversion consiste à obtenir une expression de $\ddot{q}_{n+1}$ et $\dot{q}_{n+1}$ en fonction des valeurs de la position et de la vitesse à l'instant t_n.

$$\begin{cases} \ddot{q}_{n+1} = \dfrac{(q_{n+1} - q_n)}{\beta \Delta t^2} - \dfrac{1}{\beta \Delta t}\dot{q}_n - \left(\dfrac{1}{2} - \beta\right)\ddot{q}_n \\ \dot{q}_{n+1} = \dfrac{\gamma}{\beta \Delta t}(q_{n+1} - q_n) + \dot{q}_n + (1-\gamma)\ddot{q}_n \Delta t \end{cases} \quad \text{(II.45)}$$

Ces expressions sont remplacées dans l'équation d'équilibre écrite à l'instant t_{n+1}. Le système à résoudre est donc un problème non linéaire en q_{n+1}.

II.5.3.2 Forme incrémentale

Soit le système algébro-différentiel de la dynamique d'un système multicorps soumis à des contraintes holonomes écrit sous la forme générale :

$$\begin{cases} M(q)\ddot{q} + F(q, \dot{q}, t) = -\phi_q^T \lambda \\ \phi(q, t) = 0 \end{cases} \quad \text{(II.46)}$$

où

- F représente l'ensemble des forces internes, externes et d'inerties complémentaires.
- $\phi(q,t)$ sont les équations algébriques de contraintes holonomes.

Pour la mise en œuvre de la méthode incrémentale, la quantité $r(q,\dot{q},\ddot{q})$, nommée résidu, est formée de la façon suivante :

$$r(q,\dot{q},\ddot{q},\lambda) = M(q)\ddot{q} + F(q,\dot{q},t) + \phi_q^T \lambda \qquad \text{(II.47)}$$

La forme incrémentale pour le système contraint est donc:

$$\begin{cases} M \cdot \Delta\ddot{q} + C \cdot \Delta\dot{q} + K \cdot \Delta q + \phi_q^T \Delta\lambda = -r(q,\dot{q},\ddot{q},\lambda) \\ \phi_q \cdot \Delta q = -\phi(q,t) \end{cases} \qquad \text{(II.48)}$$

avec :

- $C = \dfrac{\partial F}{\partial \dot{q}}$ $\qquad\qquad$ $K = \dfrac{\partial (M\ddot{q})}{\partial q} + \dfrac{\partial F}{\partial q} + \dfrac{\partial \left(\phi_q^T \lambda\right)}{\partial q}$

Le système se met donc sous la forme :

$$\hat{K} \cdot \Delta x = \hat{F} \qquad \text{(II.49)}$$

avec

- $\hat{K} = \begin{bmatrix} \tilde{K} & \phi_q^T \\ \phi_q & 0 \end{bmatrix}$ où $\tilde{K} = K + \dfrac{\gamma}{\beta \Delta t} C + \dfrac{1}{\beta \Delta t^2} M$
- $\Delta x = \begin{Bmatrix} \Delta q \\ \Delta \lambda \end{Bmatrix}$
- $\hat{F} = \begin{Bmatrix} -r(q,\dot{q},\ddot{q},\lambda) \\ -\phi(q,t) \end{Bmatrix}$

Des problèmes de stabilité numérique apparaissent malgré tout dans le cas de systèmes contraints [Ger96].

II.5.4 Méthode Hilbert-Hughes-Taylor

Pour corriger ces problèmes de stabilité, les équations peuvent être modifiées afin d'introduire de l'amortissement numérique ([Hil77]). L'équation d'équilibre de forme générale:

$$M(q)\ddot{q} + F(q,\dot{q},t) = -\phi_q^T \lambda \qquad (\text{II.50})$$

est remplacée par l'équation d'équilibre discrétisée à l'instant t_{n+1} :

$$M\ddot{q}_{n+1} + (1-\alpha)F(q_{n+1},\dot{q}_{n+1},t_{n+1}) + \alpha F(q_n,\dot{q}_n,t) = -(1-\alpha)\left(\phi_q^T\right)_{n+1}\lambda_{n+1} - \alpha\left(\phi_q^T\right)_n \lambda_n \qquad (\text{II.51})$$

Ainsi, en utilisant un schéma de Newmark avec les paramètres choisit comme suit :

- $\beta = \dfrac{(1+\alpha)^2}{4}$ $\gamma = \dfrac{1}{2} + \alpha$ $0 \le \alpha \le \dfrac{1}{3}$,

la méthode est précise à l'ordre 2 et inconditionnellement stable, cette modification introduisant en effet de l'amortissement numérique pour les hautes fréquences mais conservant la précision numérique pour les basses fréquences.

II.5.5 Prédiction – Correction

La méthode de Prédiction – Correction correspond à une mise en œuvre particulière des schémas de résolution implicite réalisée en trois étapes :

- Prédiction: en fonction des états du système calculés aux $p+1$ instants précédents, l'état du système à l'instant t_{n+1} suivant est estimé par $\left(\hat{q}_{n+1}, \hat{\dot{q}}_{n+1}\right)$.
- Résolution: dans un processus qui peut être itératif, l'équation de la dynamique est résolue, fournissant l'accélération $\ddot{q}_{n+1}$ à l'instant t_{n+1}.
- Correction: les positions et vitesses sont corrigées à partir de cette nouvelle accélération fournissant l'état du système $\left(q_{n+1}, \dot{q}_{n+1}\right)$ à l'instant t_{n+1}.

Conclusion

La résolution d'un problème de dynamique des systèmes multicorps correspond à un ensemble cohérent de choix sur les coordonnées, le formalisme et la méthode de résolution employée. Ces étapes sont en général liées et le choix d'un type de coordonnées se fait généralement en même temps que le formalisme. Pour la plupart des problèmes, faire le choix d'une méthode optimale, c'est faire un décompte des opérations nécessaires à la résolution afin de minimiser le temps de résolution. Cependant, ce décompte s'effectue séparément pour le modèle dynamique et pour l'intégration et ne prend pas en compte la difficulté du problème dans sa globalité [Asc97]. Il est en fait unanimement reconnu qu'aucune méthode n'est assez générale pour convenir pour l'ensemble des problèmes [Und87], certains types de modélisation ou de résolutions étant mieux adaptés à certaines classes de problèmes.

Concernant la résolution, les différentes méthodes sont équivalentes du point de vue purement analytique mais pas du point de vue numérique, c'est-à-dire dès que l'on s'intéresse à la précision ou la stabilité de la méthode. Le nombre d'opérations ou le temps de calcul, qu'ils concernent les méthodes de pénalités ou les méthodes itératives de résolution, sont liés au choix de paramètres pour lesquels il n'existe pas de méthodes universelles de détermination. Il n'est ainsi pas rare de trouver dans la littérature que ces paramètres sont choisis grâce à l'expérience du programmeur.

Le choix des méthodes dépend alors du problème ou du niveau de simulation désiré. Cette analyse doit en effet prendre en compte la précision voulue pour la simulation, la durée de la simulation, le nombre de paramètres en fonction de la capacité des machines pour un fonctionnement temps-réel par exemple. Une attention particulière doit être accordée à l'importance des différents paramètres entrant en jeu.

De même, ces choix permettent d'obtenir des propriétés particulières pour la résolution. Pour une modélisation modulaire, ces choix qui doivent donc être effectués pour répondre au critère de flexibilité recherché font l'objet du chapitre suivant.

CHAPITRE III

Modélisation modulaire

III.1 Introduction

Afin de diminuer les temps de conception et donc les coûts associés, il apparaît que le bon choix de développement doit être effectué le plus tôt possible dans le processus de conception. Les outils actuels supposent cependant d'avoir déjà une idée du système à développer. Aucun outil numérique ne propose en effet de démarche intuitive de construction mêlant simultanément les aspects *ébauche du mécanisme* et *dynamique du système* [Gom99].

Dans cette optique, un outil innovant de prototypage rapide doit posséder un certain nombre de caractéristiques qui peuvent être définies comme suit:

- l'outil permet une ébauche de la forme des pièces et de la topologie des mécanismes par construction incrémentale au travers d'une interface graphique évoluée.
- les corps sont des *boites noires* autonomes de modèles de pièces qu'il suffit d'assembler graphiquement pour construire le mécanisme.
- les modifications sont facilitées permettant ainsi de tester rapidement différentes solutions.
- des événements extérieurs peuvent modifier en ligne la simulation : changements de topologie, contraintes cinématiques, applications d'efforts, … Ces actions peuvent être produites par l'opérateur au travers d'une interface appropriée.
- son aspect modulaire le rend adapté à la simulation de mécaniques multi-physiques ou à dynamiques différentes, chaque sous-système contenant son propre mode local de résolution.

La principale contrainte est alors d'identifier les grandeurs de *communication* entre les différents sous-systèmes créés.

Le but de ce chapitre est donc de proposer des méthodes adaptées à une modélisation modulaire et facilitant la construction d'un outil de prototypage rapide. Les différentes étapes nécessaires à la construction d'un modèle doivent être choisies pour permettre une construction graphique incrémentale et autoriser des événements extérieurs à modifier la topologie au cours de la simulation. Les modèles doivent être écrits sous forme modulaire, le mécanisme n'étant que le couplage de ces sous-systèmes indépendants. Ces sous-systèmes

peuvent aussi bien être des corps rigides simples que des chaînes articulées plus complexes. Cette indépendance assure la flexibilité recherchée.

Ce type de modélisation doit permettre à terme d'utiliser des pas de temps ou des méthodes de résolution différentes pour chacun des corps, le but étant de diminuer les temps de résolution afin de tendre vers des résolutions temps-réels, conditions nécessaires à l'élaboration d'un logiciel interactif. Cette indépendance permet également de modéliser des liaisons flexibles.

Dans le chapitre précédent, les différentes étapes nécessaires à la simulation de systèmes multi-corps rigides ont été présentées. Ce chapitre expose les choix effectués répondant à la problématique de la modélisation modulaire. Sont donc présentés, les différents types de coordonnées compatibles avec une représentation modulaire du système ainsi que le modèle dynamique obtenu à partir de ces différents paramétrages. Les propriétés associées, concernant en particulier l'expression des contraintes entre corps rigides, sont exposées ainsi que la forme modulaire du modèle dynamique.

III.2 Coordonnées

III.2.1 Caractéristiques

Différents types de paramétrage sont possibles pour décrire une chaîne poly-articulée de corps rigides. Pour un outil de prototypage virtuel, ces coordonnées doivent répondre à la notion d'universalité. Elles doivent en effet décrire l'état du système qu'elle que soit sa topologie mais aussi prendre en compte d'éventuels changements de celle-ci. Le choix de coordonnées relatives n'est donc pas adapté:

- l'utilisation d'un paramètre exprimant une position relative entre deux corps ne garantit pas leur indépendance. Ils sont liés par ce paramétrage et ne tolère donc pas de changement de topologie du système, comme par exemple la coupure d'une liaison. Un corps libre, c'est-à-dire soumis à aucune contrainte de liaison ne peut pas être positionné puisqu'il ne possède pas de référence. De nouvelles coordonnées doivent dans ce cas être introduites à chaque changement de topologie.

- l'écriture des équations de la dynamique nécessite de connaître le mouvement des différents corps par rapport à un référentiel galiléen. Le calcul des vitesses et accélérations, désignées comme absolues, nécessitent donc l'utilisation de méthodes récursives obligeant à parcourir l'ensemble de la chaîne cinématique à chaque pas de simulation. Ce type de paramétrage contraint donc à utiliser des méthodes fortement liées à la topologie courante.

Les coordonnées absolues, donnant la position d'un corps par rapport à un référentiel galiléen, sont ainsi mieux adaptées à une définition modulaire du modèle.

Le nombre de coordonnées doit de plus permettre de positionner le corps quel que soit le nombre de liaisons qui le lient à ses voisins. Le cas le plus exigeant, autrement dit nécessitant le plus de paramètres est celui où un corps est libre de toute contrainte. Il faut donc choisir au moins six paramètres pour un solide dans R^3 et au moins trois dans R^2.

III.2.2 Coordonnées de type translation - rotation

III.2.2.1 Représentation classique

Cette représentation classique de la position absolue d'un solide rigide consiste à exprimer la position et la rotation d'un repère mobile $\Re_0$, lié au solide rigide, par rapport au repère galiléen $\Re$. Les rotations s'expriment au moyen des angles d'Euler, des quaternions, des angles de roulis-tangage-lacet,…..

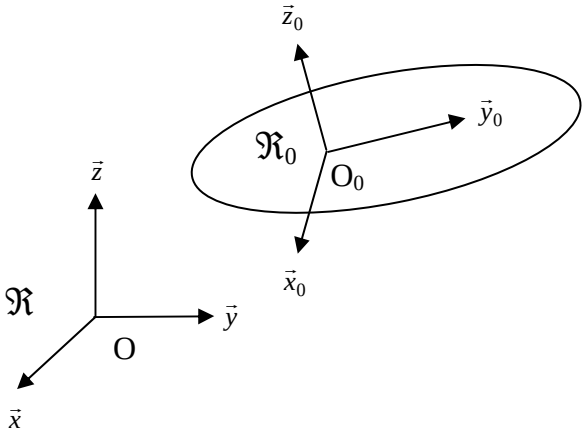


Figure III-1 – Coordonnées translation – rotation

III.2.2.2 Coordonnées canoniques

A partir de la représentation position - rotation, il est possible de construire le vecteur des coordonnées généralisées formé des composantes de ces éléments de transformations solides [Isn97] et exprimés de la façon suivante:

$$\vec{T}(t) = \begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix} \qquad \text{(III.1)}$$

et

$$R(t) = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \qquad \text{(III.2)}$$

$\vec{T}$ et R sont respectivement la translation et la rotation exprimant donc le passage de la position initiale du solide à sa position courante à l'instant t.

Le vecteur des coordonnées généralisées associées au corps C prend donc la forme :

$$q = \begin{pmatrix} T_1 & T_2 & T_3 & R_{11} & R_{12} & R_{13} & R_{21} & R_{22} & R_{23} & R_{31} & R_{32} & R_{33} \end{pmatrix}^T \qquad \text{(III.3)}$$

Les paramètres R_{ij} sont les éléments d'une matrice de rotation qui vérifie la propriété :

$$R^T \cdot R = I \qquad \text{(III.4)}$$

La matrice $R^T \cdot R$ est une matrice symétrique où six seulement des neufs équations algébriques exprimées par la relation matricielle (III.4) suffisent. En pratique, il est possible d'identifier uniquement les éléments de la matrice triangulaire supérieure.

III.2.3 Coordonnées de type points - vecteurs

III.2.3.1 Modèle 4 masses de Plexus

Ce modèle de représentation géométrique des solides rigides a été initialement introduit dans le code de calcul éléments finis Plexus. Le but initial était de pouvoir coupler des systèmes composés de corps rigides avec des structures flexibles. Le paramétrage est donc issu des méthodes de maillage éléments finis et respecte donc les contraintes suivantes [Jol93] :

- un corps rigide est discrétisé en plusieurs masses ponctuelles dont la réunion conserve les caractéristiques du corps rigide (masse, inertie, position du centre de gravité).
- des relations de contraintes sont écrites entre ces masses afin de garantir l'indéformabilité des corps.

Plexus utilise 4 masses ponctuelles positionnées dans les directions du repère principal d'inertie. En pratique, la masse totale est uniformément répartie entre les quatre masses.

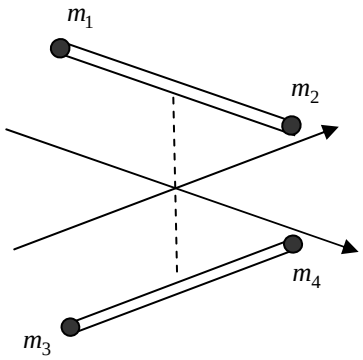


Figure III-2 – Représentation 4 masses de Plexus

Les positions de ces points sont déduites de la répartition des inerties et n'ont donc pas de signification physique. Cette remarque a une conséquence sur la formation des liaisons, nécessitant l'ajout de nœuds supplémentaires possédant six degrés de liberté (translation et rotation).

III.2.3.2 Description des *point coordinates*

Un solide peut également être représenté par une collection de points. Dans le cas le plus général, quatre points sont nécessaires (Figure III.3). Les paramètres sont les

coordonnées absolues de ces points dans le référentiel galiléen. [Nik94] propose une vue d'ensemble de la définition et de l'utilisation de ces coordonnées.

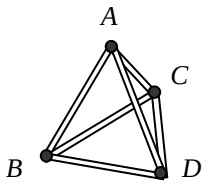


Figure III-3 – Solide rigide en représentation *point coordinates*

Le vecteur des coordonnées généralisées s'exprime alors par:

$$q = \begin{pmatrix} x_A & y_A & z_A & x_B & y_B & z_B & x_C & y_C & z_C & x_D & y_D & z_D \end{pmatrix}^T \qquad \text{(III.5)}$$

Ces douze coordonnées sont liées par six équations de corps rigides exprimant une longueur constante entre les points :

$$\begin{cases} \left\|\overrightarrow{AB}\right\| = d_1 & \left\|\overrightarrow{AC}\right\| = d_2 & \left\|\overrightarrow{AD}\right\| = d_3 \\ \left\|\overrightarrow{BC}\right\| = d_4 & \left\|\overrightarrow{BD}\right\| = d_5 & \left\|\overrightarrow{CD}\right\| = d_6 \end{cases} \qquad \text{(III.6)}$$

où d_1, d_2, d_3, d_4, d_5 et d_6 sont donc constantes issues des caractéristiques géométriques du solide.

La masse est ensuite distribuée entre ces points afin de reconstituer les caractéristiques inertielles du solide.

III.2.3.3 Coordonnées naturelles

Les coordonnées naturelles présentées dans [Bay88], et largement étudiées dans [Gar93] peuvent être vues comme un cas particulier des *point coordinates*. Egalement nommées *fully cartesian coordinates*, elle définissent la position absolue d'un solide dans l'espace par la donnée des coordonnées absolues d'au moins deux points et d'au moins un

vecteur unitaire attachés à ce corps (Figure III-4). Comme il sera vu dans la suite, afin de faciliter l'écriture des équations de liaisons, il est en fait préférable de choisir un vecteur supplémentaire.

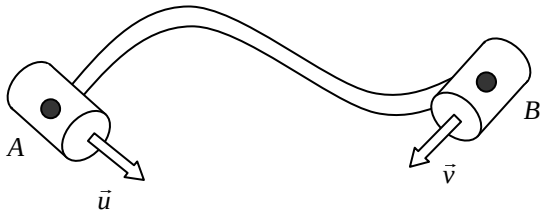


Figure III-4 – Solide rigide défini par les coordonnées naturelles

Soient A, B, $\vec{u}$ et $\vec{v}$ les deux points et les deux vecteurs rigidement liés au solide, et désignés dans la suite par le terme *élément*. A et B seront disposés de manière à définir les centres d'éventuelles liaisons. Quant à $\vec{u}$ et $\vec{v}$, ils correspondent aux axes d'éventuelles liaisons passant respectivement par A, B. Les coordonnées naturelles sont les coordonnées absolues des ces éléments dans le repère universel. Un corps est donc paramétré par $n = 12$ coordonnées généralisées représentées par le vecteur :

$$q = \begin{pmatrix} x_A & y_A & z_A & x_u & y_u & z_u & x_B & y_B & z_B & x_v & y_v & z_v \end{pmatrix}^T \qquad \text{(III.7)}$$

Un corps solide, libre dans l'espace à trois dimensions, possède $d = 6$ degrés de liberté. Il est donc nécessaire de définir $m = n - d$, soit $m = 6$ contraintes. Ces six relations traduisent l'appartenance de ces éléments au solide rigide, soit :

$$\begin{cases} \left\| \overrightarrow{AB} \right\| = c_1 & \left\| \vec{u} \right\| = 1 & \left\| \vec{v} \right\| = 1 \\ \vec{u} \cdot \vec{v} = c_2 & \vec{u} \cdot \overrightarrow{AB} = c_3 & \vec{v} \cdot \overrightarrow{AB} = c_4 \end{cases} \qquad \text{(III.8)}$$

où c_1, c_2, c_3 et c_4 sont des constantes issues des caractéristiques physiques du corps considéré et définies comme suit :

- c_1 est la distance entre les points A et B.

- c_2, c_3 et c_4 traduisent que les vecteurs $\vec{u}$, $\vec{v}$ et $\overrightarrow{AB}$ rigidement liés au solides, conservent des directions relatives constantes.

Malgré la surabondance du paramétrage ($n=12$ coordonnées pour $d=6$ degrés de liberté au maximum), ce type de représentation assure non seulement la flexibilité recherchée mais donne des expressions de contraintes simples facilitant la création et la destruction de liaisons. L'utilisation d'autres coordonnées telles que les paramètres d'Euler n'assure pas en effet cette facilité d'écriture pour les contraintes de liaisons.

III.2.4 Bilan

Les paramétrages présentés répondent à la problématique de la modélisation modulaire: les coordonnées sont absolues et assure une représentation des corps indépendamment les uns des autres.

Ils possèdent cependant quelques points de différences. Pour les coordonnées naturelles ou les *point coordinates*, les points représentent les centres des liaisons. Dans la représentation quatre masses de Plexus, c'est la répartition de la masse qui est imposée et donc la position des points qui en est déduite. Les positions des points sur le corps n'ont donc pas de signification physique et obligent à introduire des points supplémentaires possédant six degrés de liberté, munis d'une faible masse, pour exprimer les liaisons. Une expression des liaisons par des méthodes de pénalités faisant intervenir de fortes raideurs est alors à proscrire.

III.3 Expressions des contraintes de liaisons

La création de la topologie de la chaîne poly-articulée résulte de la mise en place de liaisons entre les corps. Elles correspondent à la restriction du mouvement relatif d'un corps par rapport à son voisin et s'expriment par des équations algébriques de contraintes. Dans l'espace 3D, le nombre de degrés de liberté relatif entre deux corps libres est de six. Il est donc nécessaire d'introduire *m* équations de contraintes pour le faire passer à *d*, nombre de degrés de liberté de la liaison ($d = 6 - m$).

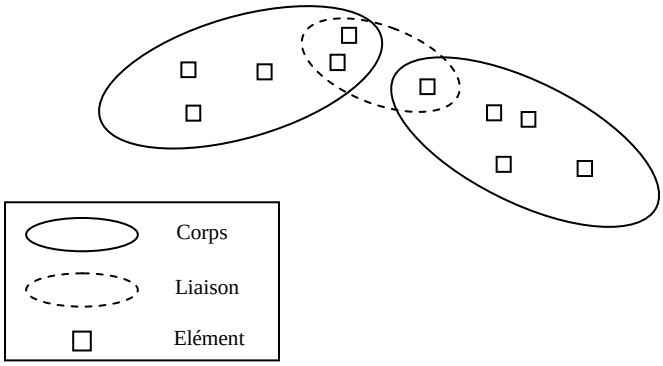


Figure III-5 – Forme générale d'une liaison

La modélisation proposée doit permettre de lier des corps dont les représentations sont différentes. Ainsi, il est nécessaire d'établir des relations unifiées des équations de contraintes définissant les liaisons. Ces équations algébriques proviennent de relations entre les coordonnées absolues de points ou de vecteurs.

Les paragraphes suivants constituent un catalogue des principales liaisons intervenant dans les mécanismes.

III.3.1 Liaison sphérique

La liaison sphérique, ou rotule, possède trois degrés de liberté de rotation ($d = 3$).

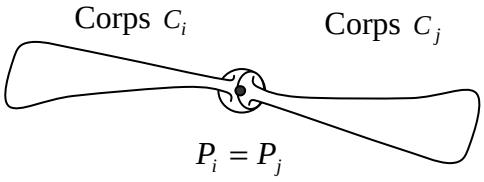


Figure III-6 – Liaison sphérique

Les point P_i et P_j appartenant respectivement aux corps C_i et C_j doivent rester confondus. Ils définissent ainsi le centre de la liaison rotule. L'expression vectorielle de contrainte s'écrit donc:

$$\overrightarrow{OP_i} = \overrightarrow{OP_j} \qquad \text{(III.9)}$$

Les trois équations scalaires de contraintes traduisent le blocage des trois translations relatives ($m = 3$).

III.3.2 Liaison pivot-glissant

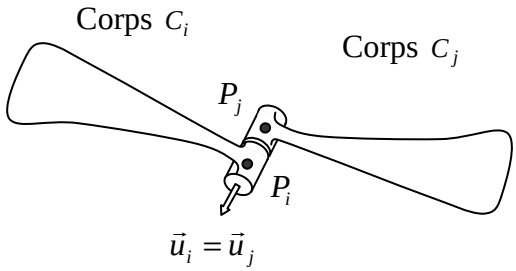


Figure III-7 – Liaison pivot - glissant

Une liaison pivot-glissant ou cylindrique possède deux degrés de liberté, un en translation et l'autre en rotation (*d = 2*). Elle se caractérise par les points suivants:

- les vecteurs $\vec{u}_i$ et $\vec{u}_j$, appartenant respectivement au corps C_i et C_j , sont colinéaires et définissent l'axe de la liaison.
- le vecteur $\overrightarrow{P_iP_j}$ formé à partir d'un point de chaque, est colinéaire à l'axe de la liaison.

Ces contraintes se traduisent par les équations vectorielles:

$$\begin{cases} \vec{u}_i \times \vec{u}_j = 0 & (a) \\ \overrightarrow{P_iP_j} \times \vec{u}_i = 0 & (b) \end{cases} \qquad \text{(III.10)}$$

Un produit vectoriel conduit à trois équations scalaires dont deux seulement sont indépendantes. Les équations III.10.a et III.10.b produisent donc quatre équations algébriques indépendantes ($m = 4$).

III.3.3 Liaison pivot

La liaison pivot est une liaison à un seul degré de liberté de rotation (*d = 1*). Elle s'obtient à partir d'une liaison cylindrique en supprimant la translation suivant l'axe de la liaison. Aux équations vectorielles III.10.a et III.10.b, il convient donc d'ajouter l'équation scalaire suivante:

$$\left\| \overrightarrow{P_i P_j} \right\| = Cte \qquad \text{(III.11)}$$

En plus des quatre équations scalaires traduisant la liaison pivot, l'équation algébrique III.11 est ajoutée portant ainsi le nombre total de contraintes à $m = 5$.

III.3.4 Liaison prismatique

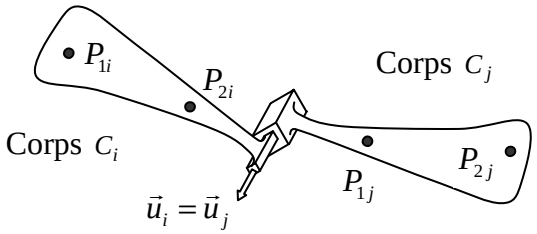


Figure III-8 – Liaison prismatique

La liaison prismatique est une liaison à un degré de liberté de translation ($d = 1$). Elle est obtenue à partir d'une liaison cylindrique à laquelle est ajoutée la contrainte supplémentaire exprimant le blocage de la rotation autour de l'axe de la liaison. Cette équation scalaire s'écrit donc:

$$\overrightarrow{P_{1i}P_{2i}} \cdot \overrightarrow{P_{1j}P_{2j}} = Cte \qquad \text{(III.12)}$$

Cette relation traduit que la direction relative entre les deux corps reste constante. Dans ce cas, le nombre d'équations de contraintes est $m = 5$.

III.3.5 Liaisons et représentations

Les expressions générales des liaisons doivent donc ensuite être exprimées pour chaque type de représentation. Cela consiste à construire la fonction permettant de calculer les coordonnées absolues des points et des vecteurs intervenants dans les liaisons en fonction de la représentation utilisée. Voici, à titres d'exemples, quelques relations :

- Pour les coordonnées naturelles

Par définition, les coordonnées naturelles sont les coordonnées absolues des points et des vecteurs intervenant dans les expressions générales des liaisons. Aucune transformation n'est donc nécessaire. Ces coordonnées conduisent alors à des expressions algébriques de contraintes de liaisons simples de forme linéaire ou quadratique (Annexe 2).

- Pour les coordonnées canoniques

Les coordonnées absolues des éléments intervenant dans les liaisons doivent être recalculées en fonction des composantes du vecteur de translation $\vec{T}$ et de la matrice de rotation R. La position absolue d'un point quelconque P d'un solide à l'instant t est donnée en fonction de la position P_0 de ce même point à l'instant initial par la relation:

$$\overrightarrow{OP} = \vec{T} + R.\overrightarrow{OP_0} \qquad \text{(III.13)}$$

Pour la liaison rotule par exemple, l'expression générale donnée par la relation III.9 devient donc:

$$\overrightarrow{T_i} + R_i \overrightarrow{OP}_{oi} = \overrightarrow{T_j} + R_j \overrightarrow{OP}_{oj} \qquad \text{(III.14)}$$

Il est à noter, à partir de cet exemple de la liaison rotule, que les expressions des contraintes pour les coordonnées canoniques sont moins simples que pour les coordonnées naturelles. Elles sont dans les deux cas linéaires ou quadratiques mais l'expression à l'aide

des coordonnées canoniques fait intervenir l'ensemble des paramètres de la translation et de la rotation. Les jacobiennes de contraintes associées sont dans ce cas moins creuses que celles issues des coordonnées naturelles.

III.4 Equations du mouvement

Les équations de Newton-Euler qui proviennent directement des théorèmes généraux de la mécanique, sont mal adaptées à une modélisation modulaire comme à la parallélisation. Celle-ci implique en effet de pouvoir effectuer des calculs simultanés sur différentes parties du modèle alors que l'obtention des équations du mouvement par le formalisme de Newton-Euler est basée sur des méthodes récursives et donc séquentielles. C'est de plus un formalisme adapté à l'utilisation de coordonnées relatives.

Quant aux équations de Hamilton, leur utilisation double le nombre d'équations. Pour des représentations déjà surabondantes telles que les coordonnées naturelles ou canoniques qui utilisent 12 paramètres, la taille du système à résoudre devient vite excessive. Ainsi, un système mécanique constitué de n corps conduit dans ce cas à un système différentiel de $2\times(12\times n)$ équations auxquelles doivent être ajoutées les équations de contraintes.

L'utilisation des équations de Lagrange est plus adaptée au problème de la modélisation modulaire pour lequel l'obtention et la résolution des équations doit se faire de la manière la plus découplée possible tout en restant de taille modérée.

III.4.1 Forme globale

L'équation de la dynamique du système global peut s'écrire sous la forme :

$$\begin{cases} M(q)\ddot{q} + R(q,\dot{q},t) = Q - C^T\lambda & (a) \\ \phi(q,\dot{q},t) = 0 & (b) \end{cases} \qquad \text{(III.15)}$$

où

- $M(q)$ est la matrice d'inertie du système de dimension $[n\times n]$.

- $R(q,\dot{q},t) = \dot{M}(q)\dot{q} - \frac{1}{2}\dot{q}^T \frac{\partial M}{\partial q}\dot{q}$ est le vecteur des termes quadratiques en vitesse aussi appelés vecteurs des forces centrifuges et de Coriolis.
- $C = \begin{bmatrix} \frac{\partial \phi_h}{\partial q} \\ \psi_{nh} \end{bmatrix}$ est la matrice jacobienne des contraintes holonomes et non-holonomes.
- λ est le vecteur des multiplicateurs de Lagrange.
- $Q = F_c + F_{nc}$ résultante des efforts extérieurs appliqués au système.
- $\phi(q,\dot{q},t) = \begin{Bmatrix} \phi_h(q,t) \\ \psi_{nh}\dot{q} + f(t) \end{Bmatrix}$

Remarque 1:

Les coordonnées naturelles ont été plus particulièrement utilisées dans les développements. Pour cette représentation, les détails de l'obtention de la matrice masse et donc des propriétés énoncées ci-dessous sont présentés en Annexe 1. Pour les autres types de coordonnées, on pourra se reporter aux références associées: [Isn97] pour les cordonnées canoniques, [Nik94] pour les *point coordinates*, [Jol93] pour la représentation quatre masses,…

Remarque 2:

Pour les coordonnées naturelles, le nombre de coordonnées utilisées peut être inférieur à douze en choisissant de représenter les corps par une forme dégénérée utilisant seulement deux points, deux points et un vecteur,….Dans le cas de la modélisation modulaire, la forme à deux points et deux vecteurs est retenue car, elle assure une forme générique pour la représentation et l'expression de la matrice masse qui dans ce cas est constante. Les positions des points et les directions des vecteurs sont donc figées mais dans le cas où elles ne correspondraient effectivement pas aux centres ou aux directions des liaisons, il faut construire la transformation exprimant les coordonnées de ces nouveaux éléments par rapport aux coordonnées naturelles comme pour les autres types de représentation (paragraphe III.3.5).

Ainsi, compte tenu des remarques, les propriétés suivantes sont mises en évidence:

- la matrice masse M est constante. Elle ne dépend que des coordonnées locales des points et des vecteurs définissant la représentation.
- les termes quadratiques des forces centrifuges et de Coriolis sont nuls.

Suite à ces remarques, le système global prend donc la forme :

$$\begin{cases} M\ddot{q} = Q - C^T \lambda & (a) \\ \phi(q, \dot{q}, t) = 0 & (b) \end{cases} \qquad \text{(III.16)}$$

III.4.2 Séparation des contraintes

Les équations de contraintes sont de deux types et traduisent :

- l'appartenance des éléments à un même corps rigide.
- les liaisons entre les corps.

Ainsi, $\phi(q, \dot{q}, t)$ peut se mettre sous la forme:

$$\phi(q, \dot{q}, t) = \begin{Bmatrix} \phi_\ell \\ \phi_{cs} \end{Bmatrix} \begin{matrix} (a) \\ (b) \end{matrix} \qquad \text{(III.17)}$$

où

- ϕ_ℓ est l'équation des contraintes externes des liaisons holonomes et non-holonomes.
- ϕ_{cs} est l'équation des contraintes internes de corps solides.

La matrice jacobienne des contraintes notée C, peut donc se mettre sous la forme:

$$C = \begin{bmatrix} H \\ J \end{bmatrix} \qquad \text{(III.18)}$$

où

- H est la matrice jacobienne des contraintes de corps solide associées aux multiplicateurs de Lagrange μ.
- J est la matrice Jacobienne des contraintes de liaison associées aux multiplicateurs de Lagrange σ.

L'équation de la dynamique s'écrit donc :

$$M\ddot{q} = Q - H^T \mu - J^T \sigma \qquad \text{(III.19)}$$

où les matrices jacobiennes sont de la forme:

$$H = \begin{bmatrix} H_1 & & 0 \\ & H_2 & \\ 0 & & H_{n_c} \end{bmatrix} \qquad \text{(III.20)}$$

et

$$J = \begin{bmatrix} J_1 \\ J_2 \\ \cdots \\ J_{n_\ell} \end{bmatrix} \qquad \text{(III.21)}$$

où

- H_j est la matrice jacobienne des contraintes internes associées au corps C_j, de dimension $[n_{cs_j} \times n_j]$.
- J_j est la matrice jacobienne de l'ensemble des contraintes de liaisons s'appliquant sur le corps C_j de dimension $[n_{c\ell_j} \times n]$.

III.4.3 Expressions des matrices jacobiennes

III.4.3.1 Contraintes internes

Des contraintes dites *internes* doivent être introduites dans le cas où la représentation utilisées pour un corps fait intervenir plus de degrés de liberté que le nombre strictement nécessaire, soit six dans l'espace 3D. Ces contraintes traduisent l'appartenance de ces degrés de liberté à un corps rigide s'exprimant par des relations de longueurs ou de directions constantes. Les équations introduites sont donc quadratiques.

Parmi les représentations proposées au paragraphe précédent, les coordonnées de type translation – rotation ne font pas intervenir de contraintes internes.

Les coordonnées canoniques, les *point coordinates* et les coordonnées naturelles expriment leurs liaisons par les relations de contraintes respectivement (III.4), (III.6) et (III.8). Elles expriment toutes des relations de distance ou d'angles constants: elles sont donc linéaires ou quadratiques.

III.4.3.2 Contraintes externes

Les contraintes de liaisons s'écrivent simplement et contribuent donc à rendre la modélisation plus flexible. Elles conduisent aussi à des équations linéaires ou quadratiques par rapport aux coordonnées. Les matrices jacobiennes associées sont donc linéaires et peu denses. En effet, seuls interviennent les degrés de liberté des éléments intervenant dans la liaison.

La forme générale de la matrice jacobienne est :

$$J = \begin{bmatrix} J_{11} & J_{12} & \dots & J_{1n_c} \\ J_{21} & J_{22} & \dots & J_{2n_c} \\ \dots & \dots & \dots & \dots \\ J_{n_\ell 1} & J_{n_\ell 2} & \dots & J_{n_\ell n_c} \end{bmatrix} \qquad \text{(III.22)}$$

où J_{ij} est la matrice jacobienne de la liaison i s'exerçant sur le corps C_j avec $i \in \{1...n_\ell\}$ et $j \in \{1...n_c\}$.

Remarque : Il est possible d'isoler les éléments (points ou vecteurs) intervenant effectivement dans une liaison afin de déterminer les termes non nuls des jacobiennes afin d'éliminer les calculs inutiles lors de la programmation. Ainsi, la jacobienne J_{ij} peut se décomposer en :

$$J_{ij} = [J_{ij,1} \quad J_{ij,2} \quad ..... \quad J_{ij,n_{ej}}] \qquad (\text{III.23})$$

où

- n_{ej} est le nombre d'éléments (point ou vecteur) servant à décrire le corps C_j.
- $J_{ij,k}$ est la partie de la jacobienne de la liaison i relativement à l'élément k du corps C_j.

La matrice J est peu dense. De nombreux termes $J_{ij,k}$ sont nuls. Pour les coordonnées naturelles, les termes $J_{ij,k}$ prennent en effet des formes particulières, comme par exemple :

- pour la mise en commun d'éléments :

$$J_{ij,k} = \varepsilon \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (\text{III.24})$$

où $\varepsilon = \pm 1$.

- pour la colinéarité, l'expression est donnée par un produit vectoriel. Les jacobiennes intervenant dans ces liaisons sont donc de la forme :

$$J_{ij,k} = \varepsilon \begin{bmatrix} 0 & z_i & -y_i \\ -z_i & 0 & x_i \\ y_i & -x_i & 0 \end{bmatrix} \qquad (\text{III.25})$$

où x_i, y_i et z_i sont les coordonnées de l'élément intervenant dans la liaison.

III.4.4 Forme locale

En choisissant un des paramétrages présentés au paragraphe III.2, la matrice masse ainsi que la matrice H, jacobienne des contraintes de corps solides sont bloc-diagonales.

Il est ainsi possible de parler d'équations locales de la dynamique où le couplage ne s'effectue que par l'intermédiaire des efforts de liaisons, traduits par les contraintes ϕ_ℓ. L'équation locale de la dynamique associée au corps C_j prend donc la forme découplée suivante :

$$\begin{cases} M_j \ddot{q}_j = Q_j - H_j^T \mu_j - \sum_{\ell=1}^{n_{\ell j}} J_{\ell j}^T \cdot \sigma_\ell \\ \phi_{cs\,j}(q_j, t) = 0 \end{cases} \qquad \text{(III.26)}$$

où $J_{\ell j}$ est de dimension $[n_{c\ell_j} \times n_j]$.

Le système d'équations d'un assemblage de corps rigides est donc la mise en commun des équations de la dynamique des éléments constituant ce corps et contraints par :

- les équations de contrainte de corps solide dites internes, assurant la cohésion de ces éléments pour former le corps rigide
- les équations de contraintes de liaisons entre les corps dites externes afin de former la topologie du système global.

III.5 Assemblage de sous-systèmes

III.5.1 Paramétrages mixtes

Le paragraphe précédent a présenté la méthode dans le cas où chacun des sous-systèmes est un corps unique. Cette méthode reste valable pour des sous-systèmes étant eux mêmes des chaînes poly-articulées. Il est alors nécessaire d'identifier les interfaces permettant de connecter les différents sous-systèmes entre eux. Cet assemblage provient de même de la mise en commun d'éléments de liaison.

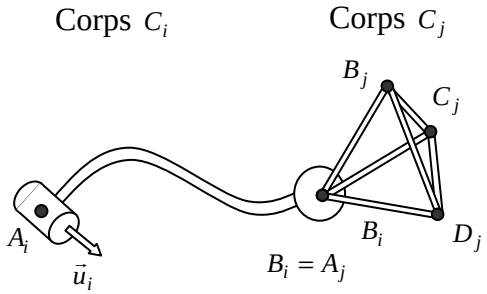


Figure III-9 – Assemblage de corps à représentations différentes

III.5.2 Assemblage de sous-systèmes

Cette encapsulation d'un assemblage dans un sous-système peut éventuellement permettre d'utiliser un paramétrage différent, par exemple des paramètres relatifs en nombre minimal, afin de réduire le nombre total de paramètres.

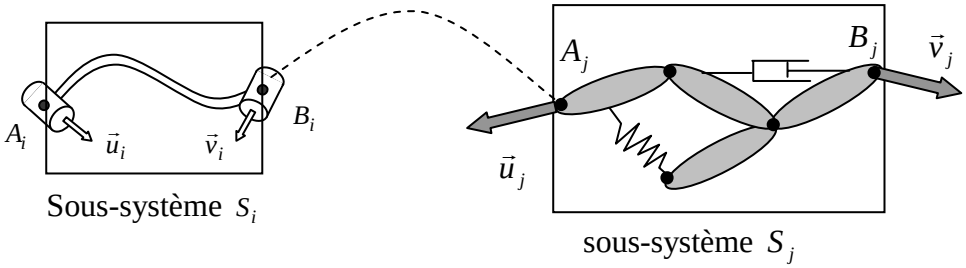


Figure III-10 – Assemblage de sous-systèmes

Différentes formes de modèles peuvent également être ainsi connectées entre elles (corps rigides - éléments finis) telles que l'assemblage de structures à corps rigides et d'autres à corps flexibles. Dans tous les cas, l'important reste l'identification dans le sous-système des points d'entrées de liaisons.

III.6 Conclusion

Les paramétrages proposés dans ce paragraphe conviennent à une modélisation modulaire. Ils conduisent à une expression locale des équations de la dynamique qui possèdent les propriétés suivantes:

- la matrice d'inertie est constante.
- le formalisme ne fait pas intervenir de termes centrifuges ou de Coriolis.
- les matrices jacobiennes sont peu denses car ne faisant intervenir que les degrés de liberté des éléments intervenant dans la liaison.

Ces paramétrages ne sont pas minimaux : le nombre de coordonnées utilisées pour la représentation d'un corps est supérieur au nombre de paramètres juste nécessaires (six dans l'espace 3D).

Dans le cas particulier des coordonnées naturelles, lors de la mise en commun de points ou de vecteurs, il est possible d'éliminer directement au moment de la construction de la liaison, les degrés de liberté redondants. Dans le cas de la modélisation modulaire, l'ensemble des degrés de liberté de chacun des corps est conservé afin de garantir l'indépendance entre les corps et donc les propriétés de modularité.

Les équations de contrainte sont quadratiques ou linéaires et conduisent à des matrices jacobiennes linéaires et peu denses facilitant ainsi l'écriture des équations de contraintes de liaisons. Le formalisme utilisé permet une écriture locale des équations de la dynamique par séparation des contraintes de corps solide et de liaison qui seront désignés par la suite par les termes *contraintes internes* et *contraintes externes*. Le modèle global possède donc une forme modulaire assurant la flexibilité recherchée.

CHAPITRE IV

Résolution modulaire

IV.1 Introduction

Le but est de s'intéresser aux méthodes de résolution conservant les propriétés de modularité apportées par la représentation et la modélisation exposées au chapitre précédent. Cette propriété est conservée si le système à résoudre, qui est de type algébro-différentiel, peut être résolu localement. Le terme *local* signifie ici que les informations transmises entre les différentes entités doivent se limiter à un échange entre voisins, c'est-à-dire entre entités connectées, et non à une information récursive qui nécessiterait une transmission tout au long de la chaîne cinématique.

Une première approche peut consister à considérer les corps libres de toutes contraintes puis à traiter des problèmes de contact et de chocs comme dans la méthode proposée par Mirtich [Mir96]. Elle est cependant adaptée aux traitements de contacts unilatéraux. Cette étude s'intéresse à d'autres types de résolution, variantes des méthodes de pénalités.

La modélisation proposée fait apparaître deux types de contraintes: les contraintes internes associées à chaque corps rigide ainsi que les contraintes externes à chacun des corps traduisant l'existence de liaisons. Ainsi le modèle dynamique local d'un système C_j écrit sous forme modulaire peut s'écrire conformément aux notations définies dans le chapitre III :

$$\begin{cases} M_j \ddot{q}_j = Q_j - H_j^T \mu_j - \sum_{\ell=1}^{n_{\ell_j}} J_{\ell j}^T \sigma_\ell \\ \phi_{cs\,j}(q_j, t) = 0 \end{cases} \qquad \text{(IV.1)}$$

La question principale liée à ce problème est donc essentiellement liée au traitement des contraintes de liaisons, c'est-à-dire à la façon de transmettre l'existence d'une liaison entre deux corps. La résolution de la dynamique d'un seul corps avec prise en compte des contraintes internes est elle, par définition, un problème local qui peut être résolu par tout type de méthode.

La solution proposée est de calculer les efforts résultant des contraintes de liaisons, contraintes *externes*, de manière explicite par une méthode classique de pénalité. Il reste alors à résoudre pour chaque corps, par la méthode de son choix, le système algébro-différentiel (IV.1) des équations de la dynamique *locales* aux corps, chacun d'eux étant soumis à un ensemble d'efforts extérieurs.

La résolution d'un problème de dynamique des systèmes multicorps avec contraintes peut être résolu en utilisant les méthodes de pénalités présentées au chapitre II. Le but de ces méthodes, directes ou itératives étant de calculer les efforts de contraintes comme fonctions des violations de contraintes en position, vitesse et accélération. La signification physique de cette modélisation est de remplacer la liaison par un système ressort-amortisseur dont il est nécessaire de donner les caractéristiques.

En choisissant une expression explicite de ces efforts de liaison, le découplage entre sous-systèmes est total. Chaque corps est considéré libre, soumis uniquement à un ensemble de forces extérieures. Ce mode de résolution des contraintes externes s'apparente aux méthodes des coupures utilisées pour la résolution de chaînes cinématiques fermées. La libération de degrés de liberté supplémentaires pour ces systèmes sur-contraints est compensée par l'ajout d'efforts de liaisons. La détermination du nombre de degrés de liberté du mécanisme n'est de plus pas nécessaire.

Les méthodes de résolution avec expression explicite des efforts liés aux contraintes ont démontrés depuis longtemps leurs limites. La stabilité des méthodes numériques d'intégration est fortement liée au choix des facteurs de pénalités. Choisis trop faibles, ils ne traduisent pas la liaison: les contraintes associées ne sont pas respectées. Ils doivent en théorie être infinis, mais des valeurs trop grandes entraînent des instabilités numériques.

Ce chapitre présente les algorithmes de résolution utilisés respectant les principes de modularité ainsi que les critères numériques de stabilité.

IV.2 Calcul explicite des efforts de liaisons

Afin de garantir l'indépendance entre les différents corps, les efforts de liaisons sont calculés de manière explicite par une méthode de pénalité. Ainsi, l'expression des multiplicateurs de Lagrange est une combinaison linéaire des contraintes en position et vitesse. Pour les contraintes holonomes et non-holonomes, ces expressions prennent la forme générale:

$$\sigma = \begin{Bmatrix} \sigma_h \\ \sigma_{nh} \end{Bmatrix} = \begin{Bmatrix} \alpha_h (p_h \phi_h + \nu_h \dot{\phi}_h) & (a) \\ \alpha_{nh} p_{nh} \phi_{nh} & (b) \end{Bmatrix} \qquad \text{(IV.2)}$$

Ainsi, l'effort total, de toutes les liaisons, appliqué au corps C_j prend la forme :

$$F_{\ell_j} = -\sum_{\ell=1}^{n_{\ell j}} J_{\ell j}^T \begin{Bmatrix} \alpha_h (p_h \phi_h^j + \nu_h \dot{\phi}_h^j) \\ \alpha_{nh} p_{nh} \phi_{nh}^{\ell} \end{Bmatrix} = -\alpha J_j^T \tilde{f}_j \qquad \text{(IV.3)}$$

α est un facteur d'échelle qui est le même ici pour toutes les liaisons afin de simplifier le formalisme.

Le système à résoudre pour un corps C_j est donc

$$\begin{cases} M_j \ddot{q}_j = \hat{Q}_j - \alpha J_j^T \tilde{f}_j \\ \phi_{cs\,j}(q_j, t) = 0 \end{cases} \qquad \text{(IV.4)}$$

où $\hat{Q}_j = Q_j - H_j^T \mu_j$ est la quantité des efforts internes et externes autres que les efforts des liaisons externes.

IV.3 Résolution des équations locales

IV.3.1 Méthode itérative

A partir d'une formulation des équations du mouvement avec Lagrangien augmenté, [Bay94] construit une méthode itérative de stabilisation de la résolution de la dynamique des systèmes multicorps contraints. Cette méthode basée sur une technique d'Uzawa s'avère particulièrement efficace : elle converge en quelques itérations. Pourtant, il a été établi en calcul des structures dans des problèmes de contact que ce type de méthode converge moins vite que des méthodes basées sur une technique type Newton-Raphson [Cur93]. L'efficacité constatée de cette méthode provient peut-être du cas particulier des coordonnées naturelles qui conduisent à une matrice de masse constante. Le calcul itératif s'effectuant au niveau des accélérations avec un modèle dynamique qui est linéaire relativement aux accélérations.

Le but de cette méthode est de calculer, sur un pas de temps, une accélération du système corrigeant les violations de contraintes en position et vitesse. Dans ce paragraphe, les termes employés concernent le corps C_j. Pour plus de clarté, l'indice j est omis.

L'expression des multiplicateurs de Lagrange en formulation augmentée s'exprime par :

$$\begin{aligned} \mu_h &= \mu_h^* + \alpha_h \left(p_h \phi_h + \nu_h \dot{\phi}_h + \ddot{\phi}_h \right) \\ \mu_{nh} &= \mu_{nh}^* + \alpha_{nh} \left(p_{nh} \psi_{nh} + \dot{\phi}_{hn} \right) \end{aligned} \qquad (\text{IV.5})$$

Les termes de pénalité viennent dans ce cas augmenter l'expression des multiplicateurs de Lagrange et non le remplacer comme dans le cas de la méthode de pénalité simple. Ainsi, la dynamique du système est décrite par un système algébro-différentiel de la forme suivante :

$$\begin{cases} M_\alpha(q)\ddot{q} + C^T \mu^* = Q - \alpha C^T \tilde{f} \\ \phi_h = 0 \\ \phi_{nh} = 0 \end{cases} \qquad (\text{IV.6})$$

avec

- $C = \begin{bmatrix} \dfrac{\partial \phi_h}{\partial q} \\ \psi^{nh} \end{bmatrix}$, matrice jacobienne des contraintes.
- $\tilde{f} = \begin{Bmatrix} \left(\dfrac{\partial \phi_h}{\partial q} \right)^T \dot{q} + (\dot{\phi}_h)_t + p_h \phi_h + \nu_h \dot{\phi}_h \\ \dot{\psi}_{nh} \dot{q} + \dot{f}(t) + p_{nh} \phi_{nh} \end{Bmatrix}$
- $M_\alpha(q) = M(q) + C^T \alpha C$

Ces multiplicateurs de Lagrange μ_h^* et μ_{nh}^* sont alors calculés de manière itérative jusqu'à convergence de l'accélération $\ddot{q}$ par :

$$\begin{aligned} \mu_h^{(i+1)} &= \mu_h^{(i)} + \alpha_h \ddot{\phi}_h^{(i+1)} + \alpha_h (p_h \phi_h + \nu_h \dot{\phi}_h) \quad (a) \\ \mu_{nh}^{(i+1)} &= \mu_{nh}^{(i)} + \alpha_{nh} \dot{\phi}_{nh}^{(i+1)} + \alpha_{nh} p_{nh} \phi_{nh} \quad (b) \end{aligned} \qquad (\text{IV.7})$$

avec :

- $\mu_h^{(0)} = 0$
- $\mu_{nh}^{(0)} = 0$

A chaque itération, la position et la vitesse, respectivement q et $\dot{q}$, restent constantes. Elles ne sont en effet pas modifiées par le processus itératif. Le calcul de l'accélération s'effectue alors par la relation récurrente suivante :

$$M_\alpha(q)\ddot{q}^{(i+1)} = M(q)\ddot{q}^{(i)} - \alpha\phi_q^T \tilde{f} \qquad (\text{IV.8})$$

avec comme conditions initiales :

$$\ddot{q}^{(0)} = M^{-1}(q) \cdot Q \qquad (\text{IV.9})$$

Les avantages de cette méthode sont multiples :

- en choisissant une masse artificielle α de l'ordre de 10^5 à 10^7, quelques itérations suffisent à respecter le critère de convergence généralement $\left(\ddot{q}^{(i+1)} - \ddot{q}^{(i)}\right) \leq \varepsilon \cdot F$ où:
  - F est une force représentative du système.
  - ε est la précision numérique souhaitée
- la matrice $M_\alpha(q)$ reste inversible quelle que soit la topologie du système et même dans le cas où les contraintes sont redondantes contrairement à la matrice masse $M(q)$. La méthode stabilise en effet la résolution même dans le cas de configurations singulières [Bay94].

Cette méthode assure également une correction des erreurs dues au schéma numérique d'intégration utilisé ainsi que des erreurs d'arrondis pour un coût numérique très minime. La masse artificielle α a été dans définie dans [Bla02] comme une masse ajoutée suivant la direction des contraintes, celle-ci doit être très importante afin d'empêcher toute violation de contraintes au niveau des accélérations.

IV.3.2 Organigramme

La figure IV-1 présente l'organigramme de mise en œuvre itérative basée sur une formulation Lagrangien augmenté. Cette méthode itérative, utilisée dans le cadre de la résolution modulaire, permet de satisfaire les contraintes internes au cours de la résolution qui expriment que les coordonnées généralisées associées au corps en question appartiennent à un solide rigide.

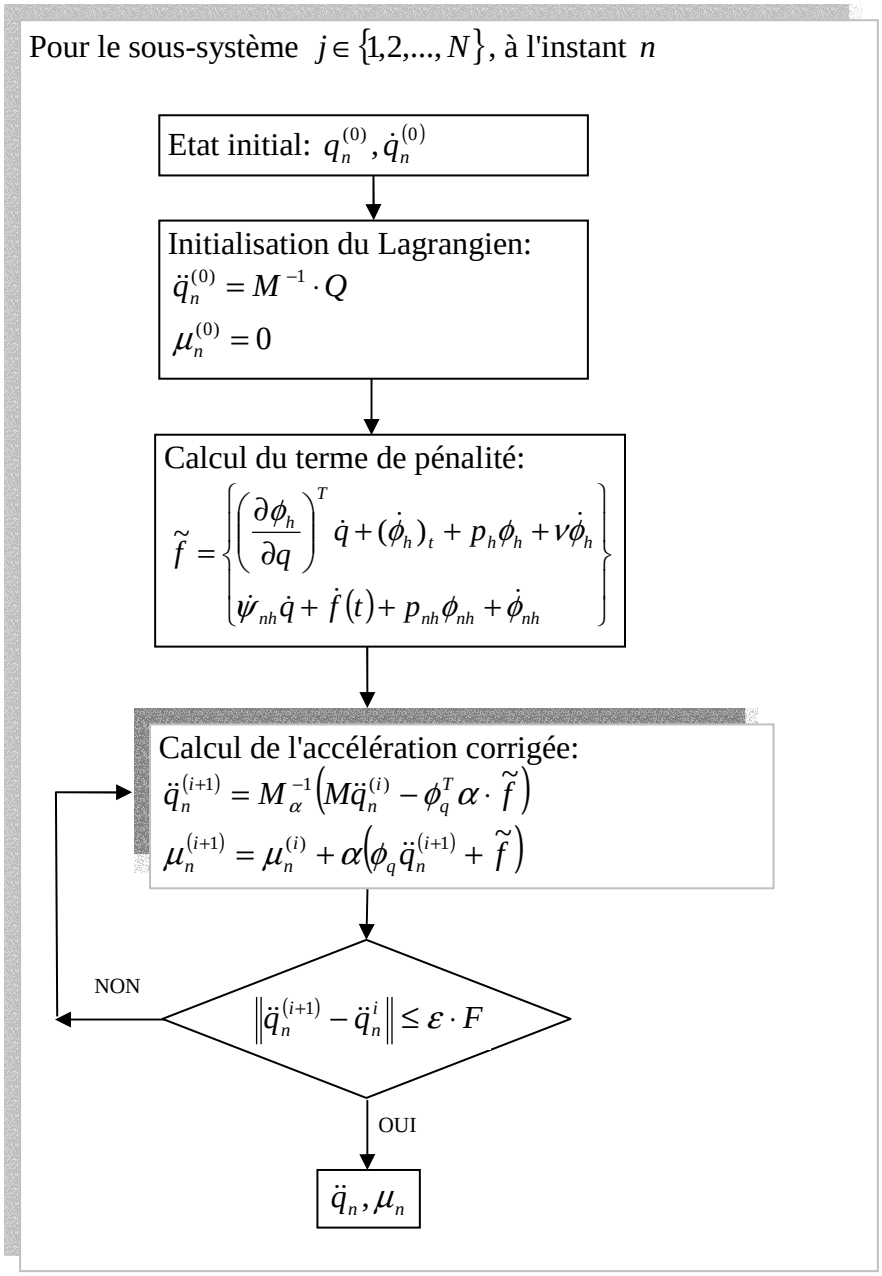


Figure IV-1 – Organigramme de la méthode itérative [Bay94]

L'état initial donné par $q_n^{(0)}$ et $\dot{q}_n^{(0)}$ provient d'une prédiction ou du résultat de l'intégration au pas de temps précédent et ne vérifie donc pas les contraintes en position et vitesse. Le terme de pénalité $\tilde{f}$ ne dépend pas de l'accélération du système et reste donc constant tout au long du processus itératif.

Cette résolution locale est ainsi appliquée à chacun des sous-systèmes de la chaîne cinématique.

IV.4 Schéma Prédiction – Correction

La résolution pour l'ensemble du système multicorps est décrite Figure IV-2.

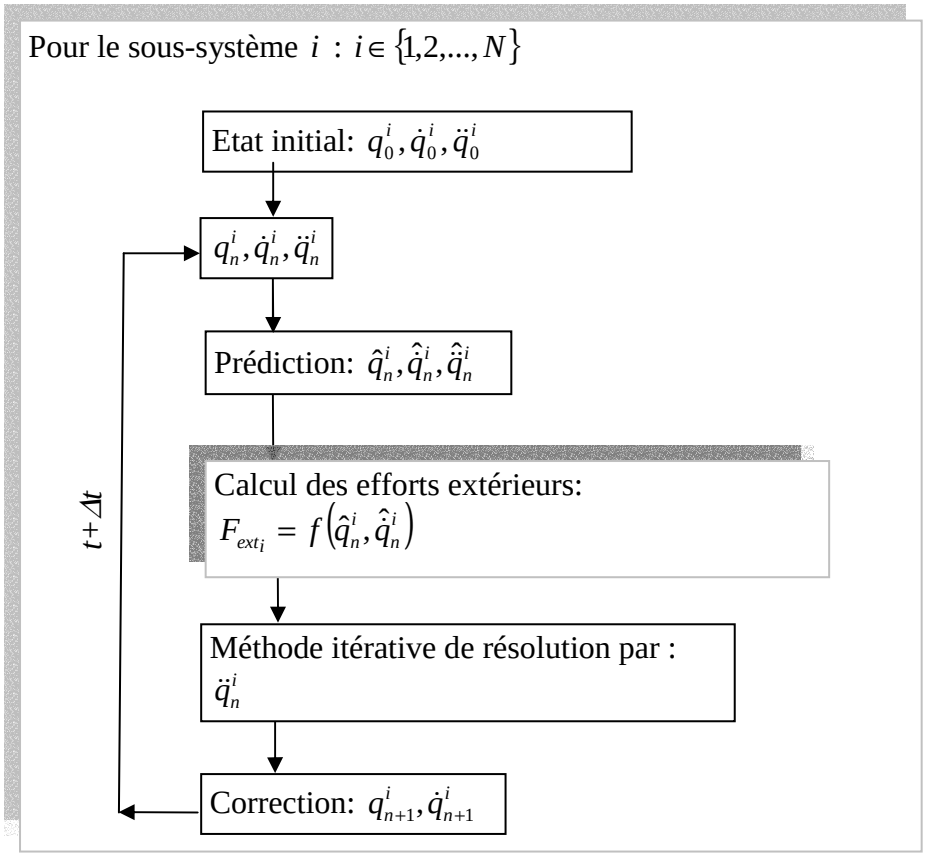


Figure IV-2 – Algorithme de résolution modulaire en Prédiction – Correction

L'intégration numérique de chaque sous-système de la chaîne cinématique est réalisée de manière indépendante.

La résolution pour l'ensemble du système nécessite le calcul des efforts extérieurs, dus aux liaisons, qui s'effectue au travers des composants *liaison*. Ces composants collectent les informations sur l'état (position et vitesse) des sous-systèmes qu'ils relient afin de déterminer explicitement les efforts extérieurs à chacun des corps. Le processus est donc local et pourrait par conséquent être différent pour chacun des sous-systèmes.

IV.5 Critère numérique de stabilité

En choisissant de traiter les contraintes extérieures par des méthodes explicites de pénalités, il apparaît des problèmes de stabilité numérique liés à la difficulté de choisir les facteurs de pénalité. En considérant chaque contrainte algébrique indépendamment des autres, il est possible d'établir une équation différentielle associée à la violation de contrainte. L'équation différentielle est du premier ordre pour une liaison non-holonome et du second ordre pour une liaison holonome. On peut définir pour chacune d'elle une constante de temps caractéristique. La résolution numérique étant un processus discret, une condition nécessaire pour avoir une simulation cohérente des violations des contraintes est que le pas de temps de simulation soit suffisamment petit devant ces constantes de temps caractéristiques, respectant ainsi le principe de Shannon. Les deux critères numériques de stabilité définis suivant ce principe, l'un pour les contraintes holonomes, l'autre pour les contraintes non-holonomes, sont indépendants des schémas numériques des composants. Ils sont à considérer en plus des critères de stabilité propres aux schémas numériques utilisés.

IV.5.1 Liaisons holonomes

Le critère repose sur l'estimation de la pulsation de l'oscillateur introduit par la méthode de pénalité. Soient deux corps C_j et C_k liés par une contrainte holonome i. Les équations du mouvement des deux corps, à partir des notations (IV.4), sont :

$$\begin{cases} M_j \ddot{q}_j = \hat{Q}_j - \alpha J_{ij}^T \tilde{f}_i \\ M_k \ddot{q}_k = \hat{Q}_k - \alpha J_{ik}^T \tilde{f}_i \end{cases} \Leftrightarrow M\ddot{q} = \hat{Q} - \alpha J_i^T \tilde{f}_i \qquad (\text{IV.10})$$

avec

- $\tilde{f}_i = \alpha(p_h^i \phi_h^i + \nu_h^i \dot{\phi}_h^i)$
- $\hat{Q}_i = Q_i - H_i^T \mu_i$ est la quantité des efforts internes et externes autres que les efforts des liaisons externes.

α est un facteur d'échelle qui n'a ici aucune signification physique et peut être quelconque. Les quantités que l'on cherche à estimer sont $\alpha_h p_h$ et $\alpha_h \nu_h$. La dérivée seconde de ϕ_h^i donne :

$$\ddot{\phi}_h^i = J_i \ddot{q} + \dot{J}_i \dot{q} \qquad \text{(IV.11)}$$

En substituant l'expression de l'accélération dans l'équation IV.11, on obtient :

$$\ddot{\phi}_h^i + J_i M^{-1} J_i^T \alpha(p_h^i \phi_h^i + \nu_h^i \dot{\phi}_h^i) = -\dot{J}_i \dot{q} + J_i M^{-1} \hat{Q} \qquad \text{(IV.12)}$$

avec

- $M = \begin{bmatrix} M_j & 0 \\ 0 & M_k \end{bmatrix}$
- $J_i = \begin{bmatrix} \frac{\partial \phi_h^i}{\partial q_j} & \frac{\partial \phi_h^i}{\partial q_k} \end{bmatrix}$

IV.5.1.1 Masses vues par l'oscillateur

De cette dernière équation, la notion de masse vue par la contrainte est introduite. En effet, le terme $J_i M^{-1} J_i^T$ peut se mettre sous la forme :

$$J_i M^{-1} J_i^T = \begin{bmatrix} \frac{\partial \phi_h^i}{\partial q_j} & \frac{\partial \phi_h^i}{\partial q_k} \end{bmatrix} \begin{bmatrix} M_j^{-1} & 0 \\ 0 & M_k^{-1} \end{bmatrix} \begin{bmatrix} \frac{\partial \phi_h^i}{\partial q_j} & \frac{\partial \phi_h^i}{\partial q_k} \end{bmatrix}^T \qquad \text{(IV.13)}$$

ou encore

$$m_{v_i} = (m_{v_i\,j}^{-1} + m_{v_i\,k}^{-1})^{-1} \qquad (\text{IV.14})$$

avec

- $m_{v_i} = (J_i M^{-1} J_i^T)^{-1}$ est la masse vue par la contrainte algébrique i.
- $m_{v_i\,j}^{-1} = \dfrac{\partial \phi_h^i}{\partial q_j} M_j^{-1} \left(\dfrac{\partial \phi_h^i}{\partial q_j} \right)^T$
- $m_{v_i\,k}^{-1} = \dfrac{\partial \phi_h^i}{\partial q_k} M_k^{-1} \left(\dfrac{\partial \phi_h^i}{\partial q_k} \right)^T$

où les trois quantités m_{v_i}, $m_{v_i\,j}$ et $m_{v_i\,k}$ sont des scalaires.

IV.5.1.2 Identification des facteurs de pénalité

Cet oscillateur est alors à identifier avec la forme générale :

$$\ddot{\phi}_h^i + 2\varepsilon_i \omega_{0i}^h \dot{\phi}_h^i + {\omega_{0i}^h}^2 \phi_h^i = -\dot{J}_i \dot{q} + J_i M^{-1} \hat{Q} \qquad (\text{IV.15})$$

où ω_{0i}^h est le vecteur des pulsations de l'oscillateur équivalent à la contrainte scalaire de la liaison i. Par identification avec l'expression de pénalité :

$${\omega_{0_i}^h}^2 = \frac{\alpha p_h^i}{m_{vi}} \qquad (\text{IV.16})$$

α est un facteur d'échelle qui est donc connu. Ce coefficient pourrait être différent pour chaque contrainte algébrique mais est en pratique égal pour toutes les contraintes et est choisi pour être de l'ordre de 10^5 à 10^7.

La stabilité du système est possible si au cours du processus d'intégration numérique l'oscillateur associé est effectivement 'vu' par le reste du système. D'après le théorème de Shannon, c'est le cas si la fréquence de l'oscillateur est petite devant la fréquence

d'échantillonnage du schéma numérique d'intégration dont le pas de temps est Δt. Ce critère se traduit par:

$$\omega_{0_i}^h << \frac{2\pi}{\Delta t} \qquad \text{(IV.17)}$$

ou encore

$$\omega_{0_i}^h = \delta \frac{2\pi}{\Delta t} \qquad \text{(IV.18)}$$

avec $\delta < 0.5$. En pratique, il sera choisi $\delta = 0.3$ pour obtenir $\delta^2 \approx 10^{-1}$.

$$p_h^i = \delta^2 \frac{m_{v_i} 4\pi^2}{\alpha(\Delta t)^2} \qquad \text{(IV.19)}$$

Après identification, il vient donc :

$$\nu_h^i = 2m_{v_i}\varepsilon_i\omega_{0_i}^h / \alpha = 2\varepsilon_i\sqrt{\frac{m_{v_i} p_h^i}{\alpha}} \qquad \text{(IV.20)}$$

IV.5.2 Liaisons non-holonomes

Un raisonnement analogue peut être effectué pour la définition d'un critère de stabilité relatif à une contrainte non-holonome *i* entre deux corps C_j et C_k.

En partant de la dérivée par rapport au temps de la contrainte non-holonome :

$$\dot{\phi}_{nh}^i = J_i\ddot{q} + \dot{J}_i\dot{q} \qquad \text{(IV.21)}$$

En substituant l'expression de l'accélération comme pour le cas de l'équation (IV.12) :

$$\dot{\phi}_{nh}^i + \alpha J_i M^{-1} J_i^T p_{nh}\phi_{nh}^i = -\dot{J}_i\dot{q} + J_i M^{-1}\hat{Q} \qquad \text{(IV.22)}$$

Cette dernière équation peut être identifiée avec :

$$\dot{\phi}_{nh}^{i} + \frac{1}{\tau_i}\phi_{nh}^{i} = -\dot{J}_i\dot{q} + J_i M^{-1}\hat{Q} \qquad (IV.23)$$

Le paramètre τ_i représente la constante de temps de ce système dynamique. Il est ainsi possible de déduire la relation suivante:

$$\tau_i = \frac{m_{v_i}}{\alpha p_{nh}^{i}} \qquad (IV.24)$$

où m_{v_i} représente cette fois-ci la masse du système vue dans la direction de la contrainte non-holonome i, soit:

$$J_i M^{-1} J_i^T = \begin{bmatrix} \frac{\partial \phi_{nh}^{i}}{\partial \dot{q}_j} & \frac{\partial \phi_{nh}^{i}}{\partial \dot{q}_k} \end{bmatrix} \begin{bmatrix} M_j^{-1} & 0 \\ 0 & M_k^{-1} \end{bmatrix} \begin{bmatrix} \frac{\partial \phi_{nh}^{i}}{\partial \dot{q}_j} & \frac{\partial \phi_{nh}^{i}}{\partial \dot{q}_k} \end{bmatrix}^T \qquad (IV.25)$$

Le critère $\Delta t << \tau_k$ doit être respecté. Il s'exprime de la façon suivante :

$$p_{nh}^{i} << \frac{m_{v_i}}{\alpha \Delta t} \qquad (IV.26)$$

Cette relation s'exprime en pratique par :

$$p_{nh}^{i} = \delta \frac{m_{v_i}}{\alpha \Delta t} \qquad (IV.27)$$

avec $\delta = 0.3$.

IV.6 Conclusion

La méthode proposée permet de résoudre de manière locale les équations de la dynamique. Les contraintes internes sont calculées par une méthode basée sur une formulation

du Lagrangien augmenté qui stabilise la résolution et les contraintes externes sont quant à elles obtenues à partir d'une expression explicite des violations de contraintes.

Ces expressions explicites conduisent à de l'instabilité. Deux critères numériques, pour liaisons holonomes et non-holonomes, ont donc été définis afin de calculer, pour un pas de temps donné, les facteurs de pénalités. Ces deux critères sont à considérer en plus de critères de stabilité liés au schéma numérique.

Ces algorithmes montrent déjà sous cette forme la structure objet à mettre en place pour respecter le critère de modularité: affectation de la résolution locale aux composants *corps* et calcul explicite des efforts externes à chacun des corps pour les composants *liaison*.

CHAPITRE V

Réalisations informatiques

V.1 INTRODUCTION

Dans cette étude, les programmes de simulation et de visualisation ont été développés à l'aide du langage C++ qui utilise la technique de programmation orientée objets.

De nos jours, cette technique de programmation est devenue une tendance majeure de plus en plus acceptée comme premier choix pour le développement de logiciels dans l'industrie. Il existe de nombreux langages tels que SMALLTALK, CLOST (Common Lisp Object System) qui soutiennent la P.O.O. (Programmation Orientée Objets), mais le C++ se positionne cependant comme le langage P.O.O. prédominant. Il possède en effet tous les avantages de la P.O.O. ainsi que toutes les fonctionnalités d'un langage procédural comme le FORTRAN. Il a également l'avantage d'être une extension de ANSI C déjà populaire et standardisé [Wan94].

Dans ce chapitre, les concepts de base et les avantages de la technique de programmation orientée objets en C++ sont exposés. Puis, l'organisation des différentes structures des programmes de simulation est décrite. Enfin, le logiciel FER/Mech est présenté ainsi que les nouveaux concepts développés dans le cadre de ce travail de thèse.

V.2 Avantages et concepts de base de la programmation orientée objets

V.2.1 Avantages de la programmation orientée objets

De nos jours, les programmes de simulation doivent permettre la réalisation de simulations de plus en plus diversifiées et complexes. Les programmes en FORTRAN, qui est le langage le plus souvent utilisé pour écrire les programmes de calculs scientifiques, atteignent leurs limites. En effet, dans ce type de programmation procédurale, la représentation des données ouvertes est limitée car le tableau est la seule structure de données disponible. Le langage n'autorise de plus pas les opérations sur des variables de types différents. L'utilisation des sous-programmes est le seul mécanisme permettant d'isoler des opérations sur des variables. Ceci nécessite alors de passer un large nombre d'arguments dans les sous-programmes ou de déclarer de manière globale les principales variables, accessibles

ainsi à de nombreux sous-programmes. Il est par conséquent difficile de les gérer. De plus, le type d'allocation mémoire en FORTRAN est statique. Des tableaux de gestion spécialisés doivent donc être créés pour simuler l'allocation dynamique de la mémoire. Cependant, la clarté des programmes est diminuée et cette subtilité complique l'accès aux données.

La quantité des procédures se multiplie en même temps que la spécificité des programmes. Et, le nombre conséquent de procédures écrites, parfois afin de réaliser des actions similaires, ne facilite pas la lisibilité alors que ces codes, souvent développés par de nombreuses personnes au cours du temps, doivent avoir comme priorités des facilités de maintenance et de ré-adaptabilité. Le FORTRAN est un langage très accessible qui donne des exécutables rapides. Cependant, les programmes de ce langage procédural ne sont en général pas facilement reformulables et il n'est pas facile d'introduire de nouveaux éléments sans engendrer des perturbations globales aux programmes déjà constitués. De ce fait, les extensions et la maintenance sont coûteuses. Ceci s'accompagne souvent d'un grand nombre d'instructions conditionnelles, incontournables pour intégrer les modifications [Dub93].

De façon à combler les difficultés présentées ci-dessus, l'utilisation du C++ avec la technique de la P.O.O. offre beaucoup d'avantages :

- Simplicité : La complexité du programme est réduite et la structure des programmes devient claire et simple car les objets logiciels simulent facilement les objets réels du domaine d'application.
- Modularité : Chaque objet forme une entité séparée dont les fonctions internes sont indépendantes d'autres parties du système.
- Modification : Il est facile d'effectuer des modifications dans la représentation des données ou des procédures employées dans les programmes orientés objets. Les changements dans un objet ne concernent aucune autre partie du programme.
- Extensibilité : De nouvelles caractéristiques peuvent être ajoutées ou des fonctionnalités modifiées en introduisant de nouveaux objets ou en changeant quelques objets existants.
- Flexibilité : Un programme orienté objets peut être très flexible et s'adapter à des situations différentes parce que les modèles d'interaction entre objets peuvent être changés sans modification des objets.

- Maintenance : Les objets peuvent être entretenus séparément, car il est très facile de localiser les problèmes et d'apporter des modifications.

- Réutilisation : Les objets peuvent être réutilisés dans des programmes différents. Un objet de matrice, par exemple, peut être employé dans tout programme qui nécessite une matrice de même genre.

Un travail de synthèse bibliographique a été effectué [Fos90] et montre que le domaine est relativement peu exploré. Néanmoins, des codes en C++ existent, et des comparaisons avec les codes FORTRAN montrent un gain de temps évident au niveau du développement et de la maintenance tandis que les temps de calcul restent raisonnables.

V.2.2 Quelques concepts de la programmation orientée objets

La programmation orientée objets est un type de programmation où le programmeur ne se préoccupe plus en premier lieu de l'enchaînement des instructions comme en programmation dite procédurale mais seulement des différents concepts qui interagissent entre eux. Elle possède, en plus de toutes les caractéristiques de la programmation procédurale telle que FORTRAN, C, etc…

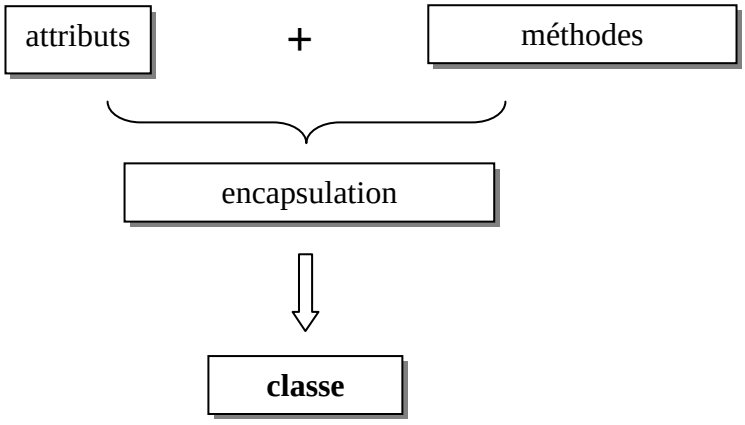


Figure V-1 : Le concept de la POO

La base fondamentale de la programmation orientée objets est le concept de classe qui combine des attributs (données) avec les fonctions employées pour manipuler ces données, les méthodes [Cap94]. Ce concept change la ségrégation traditionnelle entre les données et le programme. Cette combinaison de données et de fonctions est appelée *encapsulation*, une des

caractéristiques essentielles des classes. Les *objets* sont appelés instances d'une classe. Le cœur de la P.O.O. peut être illustré comme sur la Figure V-1.

Dans le programme, seule est considérée la manière d'employer un objet et il n'est pas nécessaire de connaître les données et les détails des opérations internes. En cachant ces détails internes, l'objet devient abstrait, cette technique est appelée *l'abstraction de données*. Ainsi qu'il est illustré sur la Figure V-2, les objets encapsulent leurs données (attributs) qui peuvent seulement être manipulées par l'objet lui-même. Pour accéder aux données d'un objet, un message doit être envoyé à l'objet. Dans ce cas, il active une de ses méthodes pour opérer sur ses propres données.

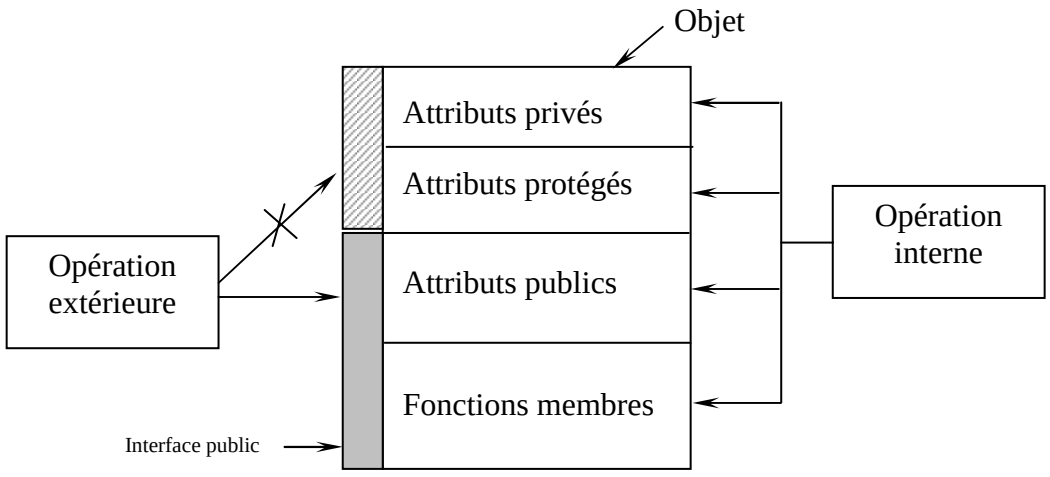


Figure V-2 : Le concept de l'encapsulation

Un objet est un outil capable de réaliser des actions prédéfinies. Il s'agit en fait d'un nouveau type de variables. Les types classiques sont : entier, réel etc., le nouveau type de variable est l'*objet*. Pour la programmation des systèmes multicorps, les objets sont par exemple "MATRIX", "VECTOR", "BODY", "JOINT", "COORDINATE", "SOLUTION" etc…. Un objet est caractérisé par ses attributs et ses méthodes qui dirigent son comportement et qui peuvent répondre à des messages en effectuant des actions spécifiques. Pour un corps, les attributs peuvent par exemple être le nombre de liaisons et les numéros des liaisons, la position du centre de gravité, la matrice d'inertie, etc…. Les attributs peuvent bien sûr être eux-mêmes des objets. Les messages, quant à eux, sont appelés *méthodes*. La Figure V-3 montre une partie de la définition de la classe BODY.

Notons que des objets existants (VECTOR et MATRIX) sont utilisés comme composants pour définir la classe BODY. Cette technique est appelée *composition*.

```
class BODY {

private :
        VECTOR vfe ; // vecteur de force
        MATRIX vke ; // matrice de masse

public :
        BODY ();        // constructeur
        ~ BODY ();      // destructeur
        virtual void integre(MATRIX&, VECTOR&){}
        ……..……
};
```

Figure V-3 - Exemple de classe : la classe BODY

Une autre notion importante de la P.O.O. est celle de l'héritage. Elle permet la dérivation d'un objet (l'objet dérivé) connexe ou similaire à un autre objet (l'objet de base). Un objet dérivé peut ainsi hériter des propriétés de sa classe de base tout en s'enrichissant de ses propres fonctions et données. L'héritage est très important pour garantir l'expansibilité et la rapidité du développement. Par exemple, dans les programmes de simulation de systèmes multicorps, il existe des liaisons différentes telles que BALL, REVOLUTE, etc. Elles partagent certains attributs qui sont définis dans la classe de base JOINT, mais possèdent leurs propres méthodes, qui sont différentes, pour calculer par exemple les efforts de liaison. Il est donc possible d'utiliser la classe JOINT comme une classe générique (classe de base ou classe mère) qui contient les attributs communs, dérivée pour définir les classes de liaisons plus spécifiques (Figure V-4).

```
class BALL : public JOINT {

public:
        BALL (TYPE, int, int,VECTOR&);
        virtual void compute_force(SOLVER_MANAGER*, MATRIX&,VECTOR&);
        ……..
};

class REVOLUTE : public JOINT {

public:
        REVOLUTE (TYPE, int, int, VECTOR&);
        virtual void compute_force(SOLVER_MANAGER*, MATRIX&, VECTOR&);
        ……..
};
```

Figure V-4 : Exemples de classes : la classe JOINT dérivée

Pour des objets correspondants aux classes dérivées différentes, une exigence commune doit répondre différemment au même appel de fonction. Cette propriété est connue sous le nom de *polymorphisme* qui est souvent associé à la notion d'héritage. Le polymorphisme permet à des méthodes, ayant des implémentations complètement différentes, d'avoir le même nom. Cette possibilité est tout particulièrement utilisée pour garder une interface cohérente entre toutes les classes dérivées d'une même classe de base. Par exemple, le calcul des forces de liaison est différent suivant les types de liaison de la Figure V-4, et pourtant, il s'agit du même type d'opération. Grâce au polymorphisme et à l'héritage, un message appelant ce calcul n'aura pas à se soucier du type de liaison auquel il s'applique, d'une part parce que, pour chaque liaison, ce calcul a le même nom, et d'autre part parce que le lien avec la bonne opération se fait de façon dynamique au moment de l'exécution (Figure V-5).

```
VECTOR BODY::Flink(double pas)
{
        VECTOR G(n);
        for (int i=0;i<nlink_tot;i++)
                if (TabL[i]!=NULL) G=G+(TabL[i]->Force(tab[i],pas));

        return G;
}
```

Figure V-5 - Boucle sur les liaisons pour le calcul des forces

Ainsi, la programmation orientée objets est particulièrement bien adaptée pour gérer le développement d'applications évolutives. Le terme *application évolutive* désigne ici tout d'abord les facilités d'évolution du logiciel ou de maintenance au cours de son existence, issues directement des propriétés de la P.O.O. énumérées dans les paragraphes précédents.

Dans un deuxième temps, il exprime que certains processus permettent de faire évoluer dynamiquement les allocations de la mémoire. Les systèmes modélisés peuvent donc ainsi être modifiés au cours de la simulation. De nouveaux objets peuvent en effet être créés ou certains détruits au travers des deux méthodes appelées *constructeur* et *destructeur*. Le constructeur assure l'allocation de la mémoire pour les attributs de l'objet dont les valeurs initiales sont ses paramètres d'entrée (Figure V-6). Le destructeur libère les zones allouées soit par appel direct, soit automatiquement en fin de programme (Figure V-7).

```
BODY::BODY (double l , double m , int dim , int num , int type_coord)
{
……..…
}
```

Figure V-6 – Exemple de constructeur : classe *BODY*

```
JOINT::~JOINT()
{
        ……;
}
```

Figure V-7 – Exemple de destructeur : classe *JOINT*

V.2.3 Exemples de développement en POO

Fort de ces caractéristiques, les exemples de développement en programmation orientée objets se multiplient. Cependant, les logiciels actuels n'utilisent pas pleinement ses propriétés. Ils sont souvent des emballages de codes déjà existants [Dix99] et ceci au détriment de l'interactivité. Les propriétés de la P.O.O. servent à améliorer l'interface ou l'organisation générale du développement informatique mais sont rarement utilisées pour apporter des fonctionnalités nouvelles dans les processus modélisés.

[Kun98], [Keil99] ou [Dix99] avec CLODION entre autres, proposent des hiérarchies de classes adaptées à la description et à la construction de système multicorps, souvent à partir de librairies définissant les classes mathématiques de bases telles que LAPACK++, version orientée objets des librairies FORTRAN. Le trait commun de ces logiciels est une modélisation des éléments du système multicorps indépendamment de leur formalisme conférant ainsi au code une certaine généricité.

Le concept FER/System [Www5] est un ensemble complet de calculs en mécanique basés sur des méthodes éléments finis avec pre et post-processing. La conception du logiciel sur une programmation orientée objets en fait un outil qui avec peu de modifications permet de s'adapter à différentes applications: FER/Drop pour la simulation en dynamique des fluides, FER/Thermal pour l'analyse de transfert thermiques, FER/Contact pour l'étude des problèmes de contacts 2D et 3D entre solides soumis à de grandes déformations, FER/Mech étant l'application dédiée à la simulation des mécanismes servant de base aux développements liés à notre étude.

V.3 FER/MECH : un logiciel interactif

Ce paragraphe présente les développements informatiques réalisés pour la construction d'un outil de prototypage rapide avec interaction en dynamique mettant ainsi en œuvre le concept de modélisation modulaire. Les implémentations ont été réalisées sur la base du logiciel FER/Mech développé par Feng [Fen03], [Www5], [Seg03].

Cet outil a tout d'abord été utilisé en tant que post-processeur pour afficher les résultats de simulation et réaliser l'animation de la scène virtuelle à partir de fichiers de résultats calculés hors-ligne par la résolution modulaire.

Dans un deuxième temps, le *solver modulaire* a été incorporé à FER/Mech. L'affichage et la résolution sont alors simultanés et il est possible de prendre en compte des événements extérieurs provenant de certains des boutons-poussoirs de la barre d'outils de l'interface afin de supprimer certaines des liaisons au cours de la simulation. L'architecture de la résolution correspond donc au diagramme de la figure I-2 du chapitre I.

Après description du logiciel FER/Mech dans les deux premiers paragraphes, les modifications apportées et les nouvelles classes d'objets développées sont présentées au troisième paragraphe.

V.3.1 Description

Le logiciel FER/Mech a été développé en C++ et utilise les librairies MFC (Microsoft Foundation Class) et OpenGL pour le développement d'applications Windows et d'effets graphiques évolués [Bra97], [Wri96]. L'interface graphique est d'une utilisation conviviale car la plupart des manipulations sont effectuées uniquement avec la souris.

Dans cet environnement, l'utilisateur construit un système multi-corps par assemblage graphique de corps de différentes formes (cylindre, sphère,…). L'interface permet également de définir les caractéristiques des différents éléments à l'aide de boîtes de dialogue, comme montré Figure V-8. Les différents corps du système sont ensuite liés toujours à l'aide de la souris par l'intermédiaire de ressorts, d'amortisseurs ou de liens rigides.

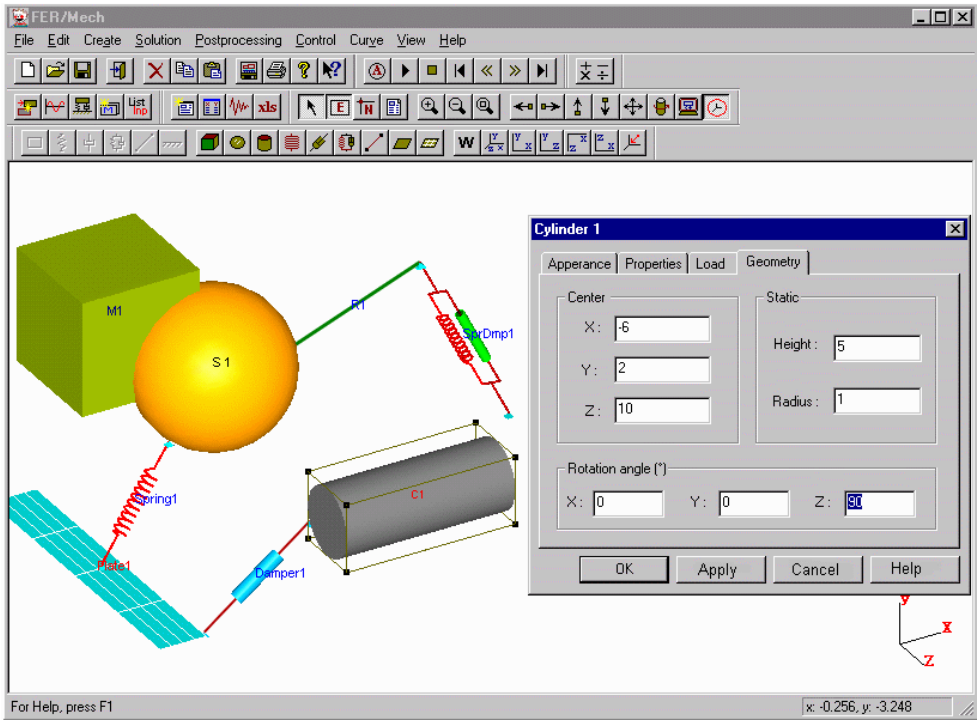


Figure V-8 – Interface utilisateur FER/Mech

V.3.2 Diagramme de classes

La Figure V-9 décrit l'organisation des principales classes définies pour le développement de FER/Mech. Les classes VECTOR et MATRIX concernent la définition des outils mathématiques nécessaires à la résolution.

La construction des pièces s'effectue au travers de la classe GEOMETRY qui contient les classes ELEMENT pour une résolution éléments finis et RIGIDBODY pour les systèmes multicorps.

La gestion des fichiers s'effectue au travers de la classe FILE_IO et le reste des fonctions d'affichage et de l'animation par COLOR, XYPLOT, PLOT2D et PLOT3D.

L'implémentation dans FER/Mech de l'approche modulaire est donc à réaliser en incorporant les nouvelles classes de corps rigides à la classe déjà existante RIGIDBODY

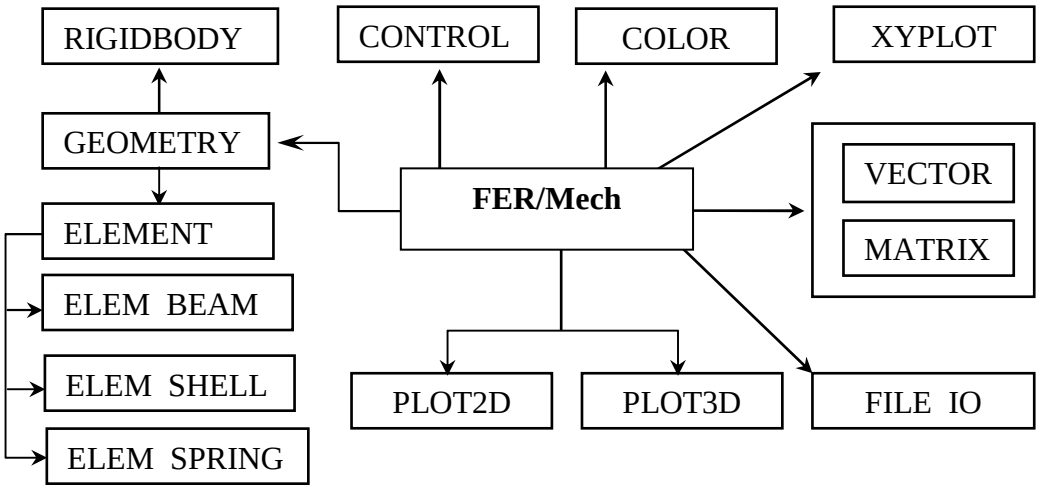


Figure V-9 – Diagramme de classes de FER/Mech

.

V.3.3 Composants mécaniques

Ces nouvelles classes sont élaborées à partir de la définition des composants *corps* et *liaison* énoncée au chapitre III et résumés sur la Figure V-10.

Composant *corps*:

- caractéristiques physiques
- type de coordonnées
- schéma numérique

Composant *liaison*:

- type de liaison
- expression de la contrainte

Figure V-10 – Définition des composants mécaniques

Le composant *corps* se définit indépendamment de sa représentation. Il contient les caractéristiques physiques du sous-système ainsi que les méthodes de résolution des équations locales de la dynamique.

Le composant *liaison* existe quant à lui indépendamment des corps. Il dépend du type de liaison et contient les méthodes nécessaires aux calculs des efforts explicites à appliquer aux corps qu'il relie.

Ces composants définissent donc directement les objets informatiques à développer pour réaliser une modélisation modulaire. Les deux classes principales sont la classe BODY et la classe JOINT.

V.3.4 Classe *BODY*

Cette classe décrit les attributs et méthodes nécessaires à la description d'un corps rigide en modélisation modulaire.

```
BODY::BODY (double l , double m , int dim , int num , int type_coord)

VECTOR BODY::Integre(double t , double pas)

VECTOR phi_cs();

VECTOR dphi_cs();

VECTOR BODY::Flink(double pas)
{
        VECTOR G(n);
        for (int i=0;i<nlink_tot;i++)
                if (TabL[i]!=NULL) G=G+(TabL[i]->Force(tab[i],pas));
        return G;
}
```

Figure V-11 – Principales méthodes de la classe *BODY*

A partir des données fournies par l'interface graphique, le constructeur instancie les corps de la chaîne cinématique en leur attribuant leurs caractéristiques physiques (dimensions, masse, position du centre de gravité, matrice d'inertie). Au moment de la création, il est également nécessaire de connaître le type de coordonnées qui seront utilisées pour représenter ce corps particulier.

En plus des données classiques sur les caractéristiques physiques du corps, chaque corps répertorie l'ensemble des liaisons qui agissent sur lui par l'intermédiaire du tableau *TabL* (Figure V-12).

```
JOINT * TabL[5];
```

Figure V-12 – Exemple d'attribut de la classe *BODY*

Pour la résolution, chaque corps détient son propre schéma d'intégration au travers de la méthode *Integre* qui sollicite les objets de JOINT pour le calcul des efforts de liaison au travers de la méthode *Flink()* (Figure V-11). Cette méthode utilise alors le tableau des pointeurs de liaison *TabL*.

Chaque corps possède de plus, pour la mise en œuvre du Lagrangien augmenté, les méthodes nécessaires au calcul des contraintes internes, (*phi_cs(), dphi_cs(),…*).

V.3.5 Classe *JOINT*

La classe JOINT définit les différentes liaisons intervenant dans la chaîne cinématique (Figure V-13).

```
JOINT::JOINT(BODY *c1 , BODY *c2 , int a , int b , int c)

JOINT::~JOINT()
{
        TabB[0]->nlink-=1;
        TabB[1]->nlink-=1;

        TabB[0]->TabL[num1]=NULL;
        TabB[1]->TabL[num2]=NULL;
}

MATRIX Jl1();

MATRIX Jl2();

VECTOR phi();

VECTOR dphi();
```

Figure V-13 – Principales méthodes de la classe *JOINT*

Les liaisons sont instanciées à partir d'un fichier de données renseigné par l'intermédiaire de l'interface graphique. Elles sont considérées comme faisant intervenir deux corps et nécessitent donc un pointeur pour chaque corps entrant en jeu dans la liaison, répertoriés à l'aide de l'attribut *TabB* (Figure V-14). Le type de la liaison, ainsi que la position sur le corps (numéro d'élément dans le cas des coordonnées naturelles) sont également renseignés.

```
BODY * TabB[2];
```

Figure V-14 – Attribut de la classe *JOINT*

Cette classe contient également les méthodes nécessaires au calcul des contraintes externes calculées de manière explicite. Le destructeur *~JOINT( )* est quant à lui appelé, par exemple, sur un événement déclenché par l'opérateur et met à jour les tableaux des pointeurs de liaisons qui de ce fait, n'interviennent plus dans le calcul des efforts extérieurs dus aux liaisons.

V.3.6 Résolution

Les éléments informatiques de la chaîne cinématique étant définis, la résolution est réalisée dans le programme principal par appel des résolutions locales de tous les composants. La Figure V-15 présente un extrait de ce programme pour la résolution modulaire d'un système mécanique constitué de deux corps rigides représentés par les objets *corps1* et *corps2*, de type *BODY*.

La méthode générique de résolution *Integre* est appelée pour tous les corps du système. L'algorithme nécessite ensuite la mise à jour de l'état de chaque composant (position et vitesse) réalisée à l'aide de la méthode *Re_init()*. Il est fait appel à ces méthodes uniquement que lorsque toutes les résolution locales ont été effectuées.

Remarque : Le code présenté est propre à l'exemple. L'appel aux différentes méthodes a en effet été codé manuellement pour des raisons de lisibilité. Les objets composant la chaîne cinématique sont en fait contenus dans un tableau, initialisé par le fichier de données construit et mis à jour par l'interface graphique. L'appel aux méthodes *Integre* et *Re_init()* est réalisée pour tous les éléments de ce tableau.

```
void main (void)
{       ……..
    // --------- Boucle temporelle----------
    for (n_pas=0;n_pas<n_tot;n_pas++)
    {  ….
        corps1.Integre(t,pas);          // Résolution locale 1
        corps2.Integre(t,pas);          // Résolution locale 2
        ….
        // Réinitialisation état corps
        corps1.Re_init();
        corps2.Re_init();
        ….
     t+=pas;
    } //FOR……
} //main
```

Figure V-15 – Implémentation de la résolution modulaire

Cette forme montre l'indépendance dans la résolution. Les détails d'implémentation de la résolution, censée pouvoir être différente pour chaque corps, sont masqués. Une modification des schémas d'intégration ne remet alors pas en cause la structure du programme principal.

V.4 Conclusion

Le choix de la programmation orientée objets se justifie donc pour ses diverses propriétés structurantes et d'adaptabilité. Ainsi, la déclinaison de FER/System en plusieurs logiciels adaptés à différentes problématiques s'effectue de manière plus aisée qu'en programmation procédurale classique. L'évolution de FER/Mech vers FER/Mech interactif est donc également réalisée sur le principe de l'assemblage de classes déjà testées et validées.

Plus encore, les structures de programmation sont adaptées à la problématique de la modélisation et de la résolution modulaire. Les corps ou les portions de systèmes multicorps à assembler pour constituer le système total peuvent ainsi être encapsulés dans des entités indépendantes munies de leurs propres méthodes pour lesquels les interfaces sont de plus clairement identifiées.

CHAPITRE VI

Résultats

VI.1 Introduction

Ce chapitre présente les résultats de simulation obtenus pour divers systèmes 2D et 3D. Pour le premier exemple, la simulation est réalisée pour différents paramétrages et différentes méthodes de résolution afin de valider la méthode de résolution modulaire proposée. Les animations réalisées emploient FER/Mech comme un simple post-processeur. Dans la dernière partie, la séquence présentée illustre l'implémentation du solver modulaire dans FER/Mech et montre les possibilités d'interaction.

VI.2 Pendule double

Le pendule double a été choisi à titre de référence afin de valider les propriétés de la modélisation modulaire et des algorithmes de résolution associés. Les modèles et les résultats de simulation obtenus par le paramétrage minimal à l'aide de deux angles ainsi que par le paramétrage modulaire à l'aide des coordonnées naturelles sont présentés.

VI.2.1 Modélisation

Le système étudié est constitué de masses ponctuelles m_1 et m_2 liées par deux tiges rigides sans masse de longueurs respectives ℓ_1 et ℓ_2. Le système est soumis à la seule force de gravité. Il possède $d = 2$ degrés de liberté.

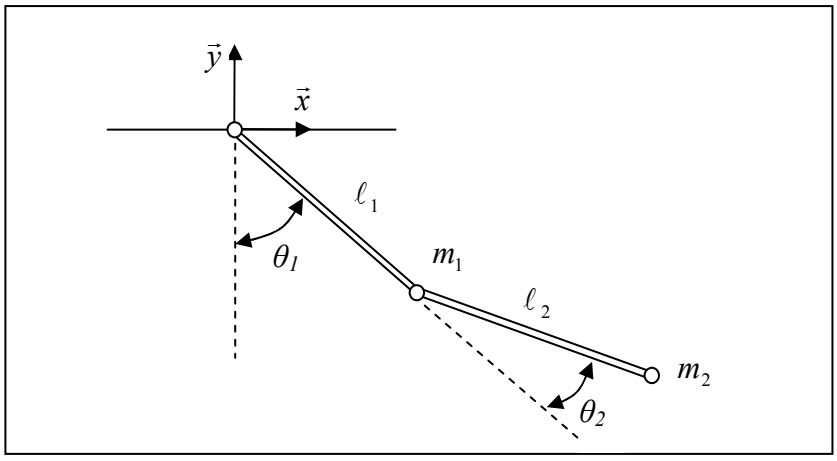


Figure VI-1 - Pendule double: paramétrage minimal

La Figure VI-1 et la Figure VI-2 décrivent les paramétrages retenus pour les simulations et nommées dans la suite paramétrage 1, paramétrage 2.

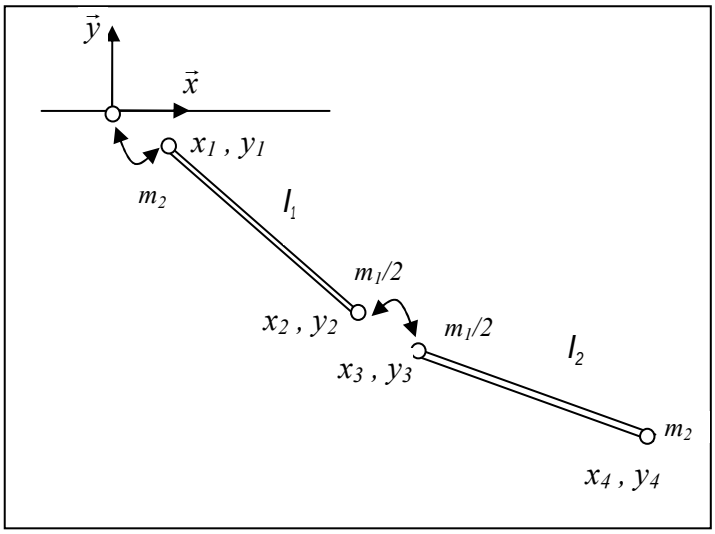


Figure VI-2 - Pendule double : paramétrage modulaire

Les équations sont données pour un système ayant les caractéristiques $m = m_1 = m_2$ et $\ell = \ell_1 = \ell_2$.

- **Paramétrage 1** : représentation minimale par les angles θ_1 et θ_2

Le système possède deux degrés de liberté, ce paramétrage est donc minimal. L'équation de la dynamique se réduit à une équation différentielle ordinaire :

$$\begin{pmatrix} m\ell^2(3+2\cos\theta_2) & m\ell^2(1+\cos\theta_2) \\ m\ell^2(1+\cos\theta_2) & m\ell^2 \end{pmatrix}\begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{pmatrix}+\begin{pmatrix} mg\ell\left((2\sin\theta_1+\sin(\theta_1+\theta_2)\right) \\ m\ell^2\sin\theta_2\left(\dot{\theta}_1^2+\dot{\theta}_1\dot{\theta}_2\right)+mg\ell\sin(\theta_1+\theta_2) \end{pmatrix}=\begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad (\text{VI.1})$$

- **Paramétrage 2** : coordonnées modulaires x_1, y_1, x_2, y_2, x_3, y_3, x_4 et y_4.

Pour ce paramétrage, le nombre total de contraintes doit être de 6. Pour ce paramétrage modulaire, on distingue les équations de contraintes de corps solides des équations de liaisons entre les corps. Le système s'écrit alors, comme vu au chapitre III:

$$\begin{cases} \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{pmatrix} = \begin{pmatrix} F_{ext1} \\ F_{ext2} \end{pmatrix} + \begin{bmatrix} H_1^T & 0 \\ 0 & H_2^T \end{bmatrix} \cdot \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} + \begin{pmatrix} \sum_{\ell=1}^{2} J_{\ell,1}^T \sigma_\ell \\ \sum_{\ell=2}^{2} J_{\ell,2}^T \sigma_\ell \end{pmatrix} \\ \phi^{cs}(q,t) = \begin{cases} \frac{1}{2}\left((x_1 - x_2)^2 + (y_1 - y_2)^2\right) - \ell_1^2 = 0 \\ \frac{1}{2}\left((x_3 - x_4)^2 + (y_3 - y_4)^2\right) - \ell_2^2 = 0 \end{cases} \\ \phi^{\ell}(q,\dot{q},t) = \begin{cases} x_1 = 0 \\ y_1 = 0 \\ x_2 - x_3 = 0 \\ y_2 - y_3 = 0 \end{cases} \end{cases} \qquad \text{(VI.2)}$$

Les matrices d'inertie sont, dans ce cas, très simples et ont pour expression:

$$M_1 = \begin{bmatrix} m & 0 & 0 & 0 \\ 0 & m & 0 & 0 \\ 0 & 0 & \frac{m}{2} & 0 \\ 0 & 0 & 0 & \frac{m}{2} \end{bmatrix} \qquad M_2 = \begin{bmatrix} \frac{m}{2} & 0 & 0 & 0 \\ 0 & \frac{m}{2} & 0 & 0 \\ 0 & 0 & m & 0 \\ 0 & 0 & 0 & m \end{bmatrix} \qquad \text{(VI.3)}$$

Les jacobiennes de contraintes s'écrivent alors :

$$H_1 = \begin{pmatrix} x_1 - x_2 & y_1 - y_2 & -(x_1 - x_2) & -(y_1 - y_2) \end{pmatrix} \qquad \text{(VI.4)}$$

$$H_2 = \begin{pmatrix} x_3 - x_4 & y_3 - y_4 & -(x_3 - x_4) & -(y_3 - y_4) \end{pmatrix} \qquad \text{(VI.5)}$$

$$J_{\ell,1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(VI.6)}$$

$$J_{\ell,2} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(VI.7)}$$

L'exposé de ces deux modèles pour un même système illustre les propriétés énoncées au chapitre III. Avec le paramétrage 1, le nombre d'équations différentielles est minimal mais elles sont fortement non-linéaires. De plus, la matrice masse dépend de la position du

système. Dans le cas de la modélisation modulaire à l'aide des coordonnées naturelles, le nombre d'équations à résoudre est plus important mais:

- la matrice masse est constante.
- les jacobiennes de contraintes internes (VI.4) et (VI.5) sont linéaires par rapport aux coordonnées et les jacobiennes de contraintes externes (VI.6) et (VI.7) sont constantes.

Le paramétrage 2 présente donc les propriétés de flexibilité recherchée comme le montre les résultats du paragraphe suivant.

VI.2.2 Résultats de simulation

Les caractéristiques du système sont $m = 1\,\text{kg}$ et $\ell = 1\,\text{m}$. L'état initial correspond à $\theta_1 = \frac{\pi}{2}$ et $\theta_2 = -\frac{\pi}{2}$. Le système est lâché à vitesse nulle.

Les méthodes d'intégration utilisées pour ces deux paramétrages sont:

- l'équation différentielle ordinaire (VI.1) résolue par une méthode explicite de Runge-Kutta d'ordre 4 avec un pas $\Delta t = 10^{-4}\,\text{s}$.
- la forme modulaire des équations (VI.2) résolue par une méthode centralisée H.H.T.
- la forme modulaire représentée par les équations (VI.2) résolue avec l'algorithme de Newmark en prédiction – correction avec Lagrangien augmenté conformément à l'algorithme de la Figure IV.2.

Les trajectoires de l'extrémité du double-pendule obtenues pour ces trois méthodes sont représentées sur la Figure VI-3. La résolution modulaire pour un pas de temps $\Delta t = 10^{-4}\,\text{s}$ conduit à des trajectoires quasi-identiques à celles obtenues par des méthodes centralisées classiques de type Runge-Kutta d'ordre 4 pour le paramétrage 1 ou H.H.T. pour le paramétrage 2. Pour un pas de temps plus grand de $\Delta t = 10^{-3}\,\text{s}$, la trajectoire en pointillés se différencie nettement des autres.

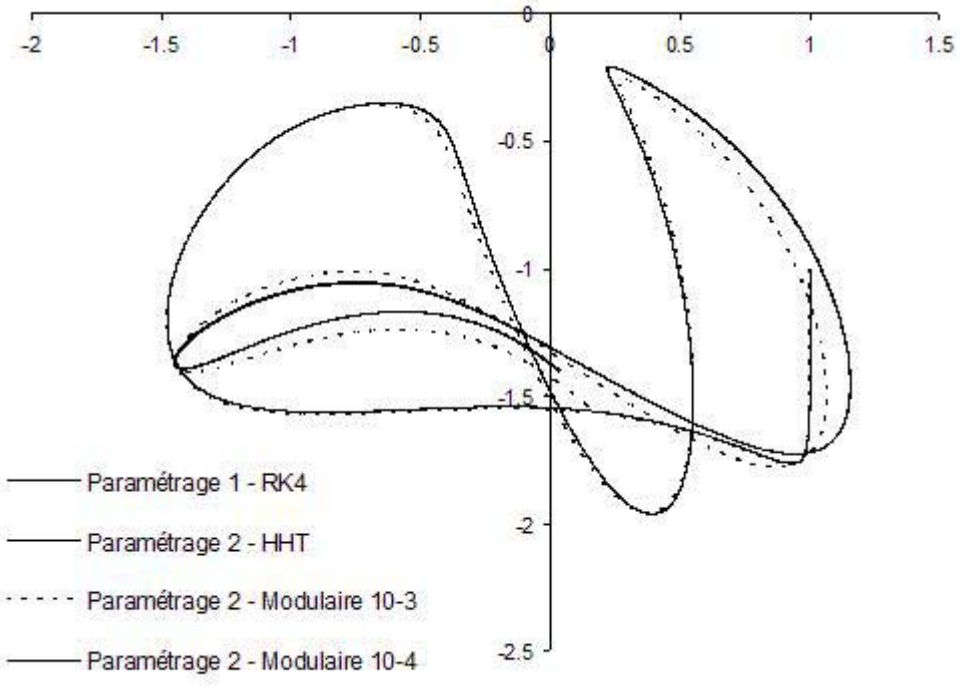


Figure VI-3 - Trajectoires du double pendule, durée 5s

- Violations de contraintes en position et vitesse pour les corps 1 et 2.

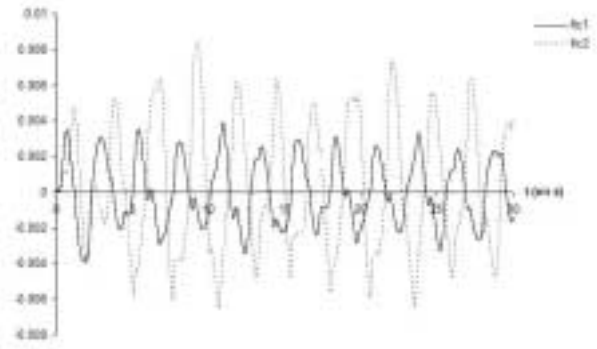


Figure VI-4 – Violations de contraintes internes en position fic1 et fic2 pour les corps 1 et 2.

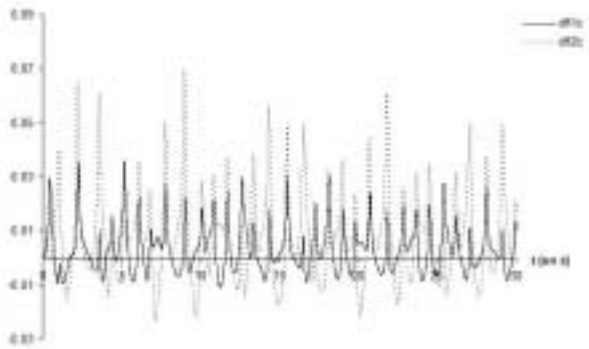


Figure VI-5 – Violations de contraintes internes en vitesse dfic1 et dfic2 pour les corps 1 et 2.

Les violations de contraintes en position (Figure VI-4) correspondent physiquement aux allongements des corps au cours de la simulation qui reste très faibles (entre -0.6 et 0.8 pourcents de la longueur initiale).

- Violations de contraintes en position fi1 et fi2 pour les liaisons 1 et 2.

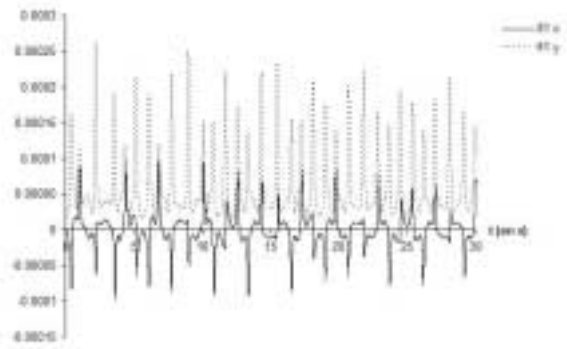


Figure VI-6 – Violations des contraintes en position fi1_x et fi1_y associées à la liaison 1.

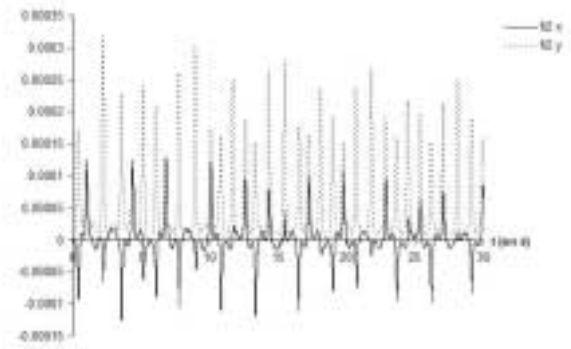


Figure VI-7 – Violations des contraintes en position fi2_x et fi2_y associées à la liaison 2.

- Violations de contraintes en vitesse dfi1 et dfi2 pour les liaisons 1 et 2.

Figure VI-8 – Violations des contraintes en vitesse dfi1_x et dfi1_y associées à la liaison 1.

Figure VI-9 – Violations de contraintes en vitesse dfi2_x et dfi2_y associées à la liaison 2.

L'énergie de chacun des corps ainsi que l'énergie mécanique totale du système sont données par la Figure VI-10.

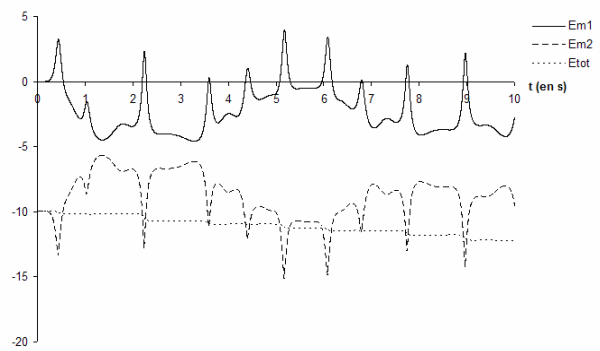


Figure VI-10 – Energie mécanique Em1 et Em2 de chacun des corps
et énergie totale Etot du système.

L'énergie totale E_{tot} du système diminue au cours du temps. La forme explicite des contraintes externes introduit en effet de l'amortissement. Les plus fortes de ces variations surviennent lors des variations importantes d'énergie et correspondent à un transfert d'énergie mécanique entre les corps 1 et 2. La différence est perdue dans la liaison qui est fortement sollicitée à ces instants.

Pour confirmer cela, la même simulation a été réalisée en modifiant simplement les conditions initiales. Les masses m_1 et m_2 sont positionnées pour correspondre au paramétrage 1 équivalent: $\theta_1 = \frac{\pi}{10}$ et $\theta_2 = 0$. L'énergie totale du système est mieux conservée car les liaisons entre les corps sont moins sollicitées. Les violations de contraintes sont en effet moins importantes.

Classiquement, l'augmentation du pas de temps conduit à des erreurs dus au schéma numérique plus importantes. Les violations de contraintes se trouvent donc également augmentées et peuvent conduire, pour des facteurs de pénalité constants, à des efforts de contraintes trop importants conduisant à de l'instabilité numérique. Le critère proposé dans le chapitre 4 remplit donc ici son rôle de stabilisation en diminuant la raideur effective dans les liaisons. Cet ajustement s'effectue alors au détriment de la précision comme il est montré Figure VI-11. Ce critère permet également de réaliser des simulations relativement longues (Figure VI-12) sans explosion numérique.

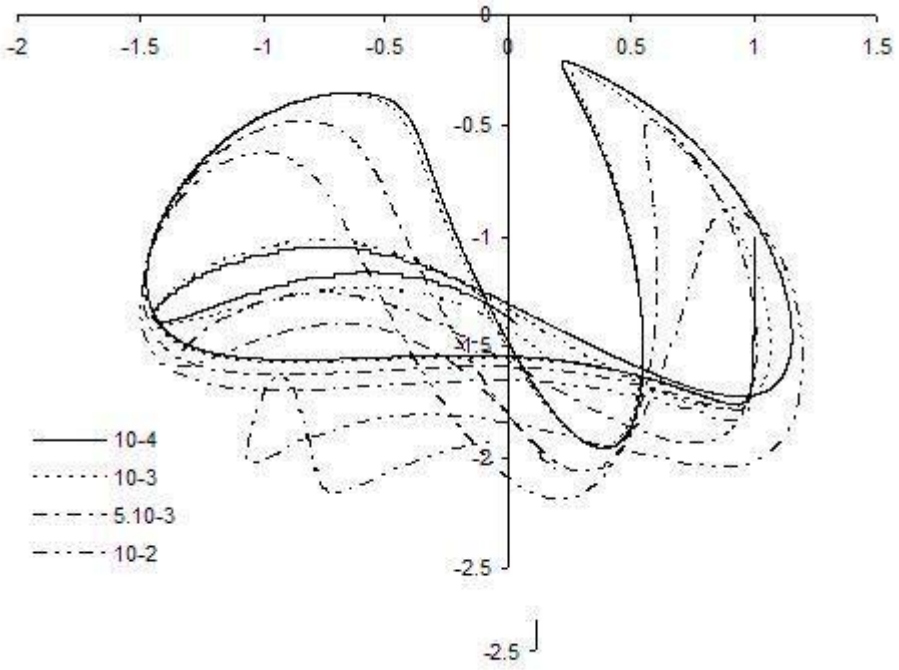


Figure VI-11 – Pendule double, pas de temps $\Delta t = 10^{-2}$

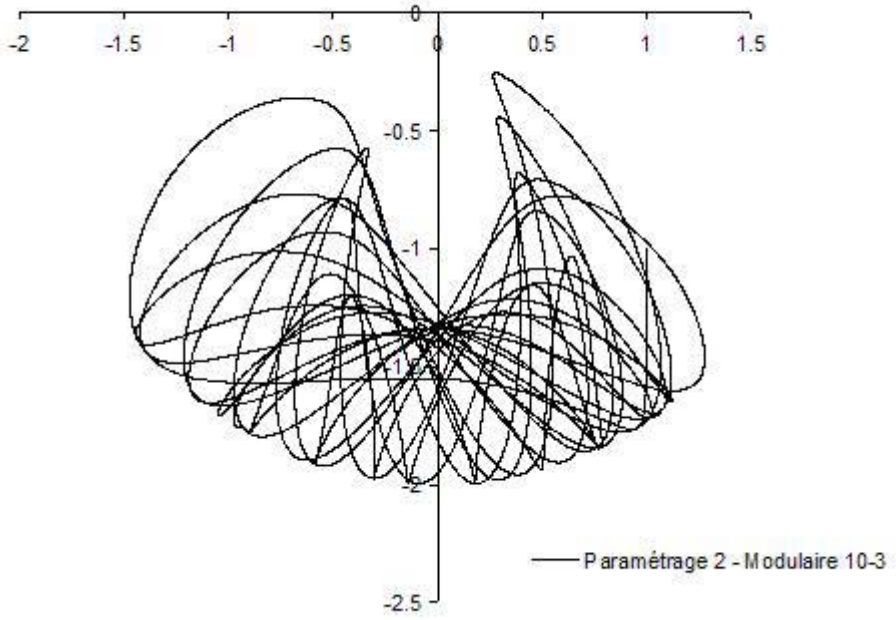


Figure VI-12 – Pendule double, pas de temps $\Delta t = 10^{-3}$, durée : 30 s

VI.3 Autres systèmes 2D

VI.3.1 Système bielle-manivelle

Ce système permet à la fois de tester une chaîne cinématique fermée, une liaison glissière ainsi que la robustesse des algorithmes en présence de système présentant des configurations singulières.

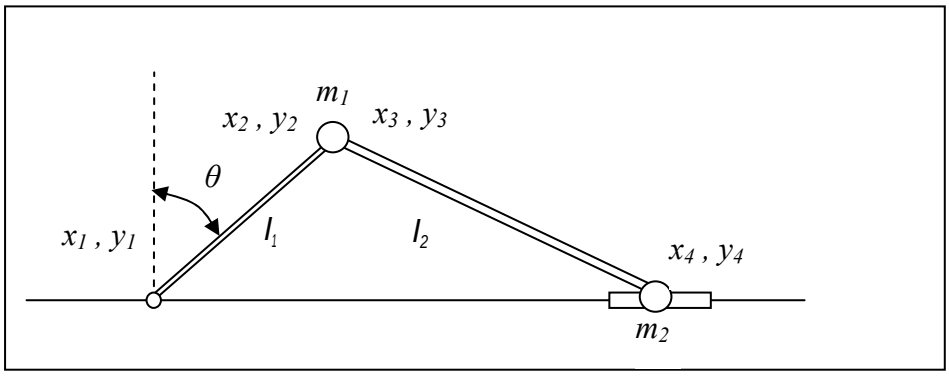


Figure VI-13 : Système bielle-manivelle

Les équations de la dynamique de ce système sont simples à obtenir puisqu'elles sont identiques à celles du pendule double. Seule reste à ajouter une équation de contrainte traduisant la liaison glissière et exprimée par l'appartenance de la masse m_2 à l'axe $(0,\vec{x})$, soit:

$$\phi(x_3, y_3, x_4, y_4) = y_4 \qquad \text{(VI.8)}$$

d'où la jacobienne des contraintes associée :

$$\phi_q(x_3, y_3, x_4, y_4) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(VI.9)}$$

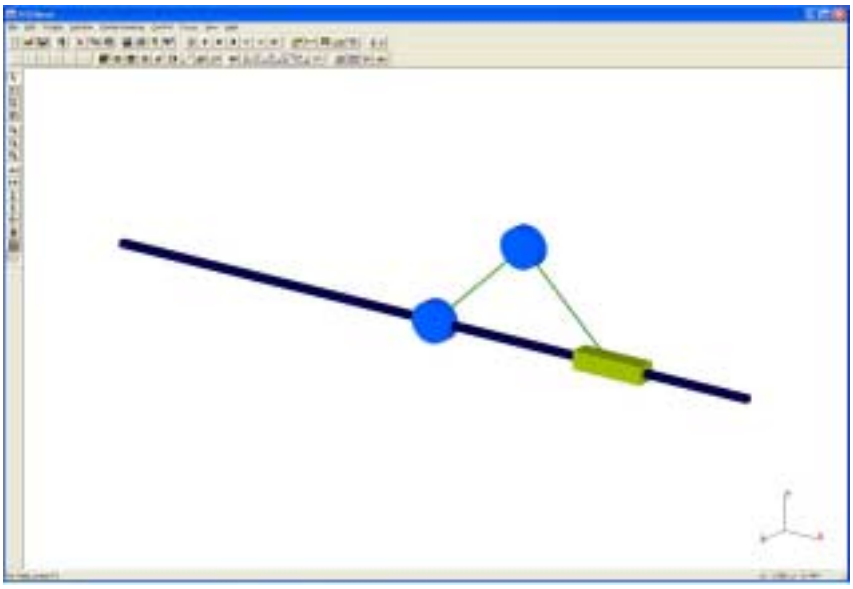


Figure VI-14 – Représentation FER/Mech du système bielle-manivelle

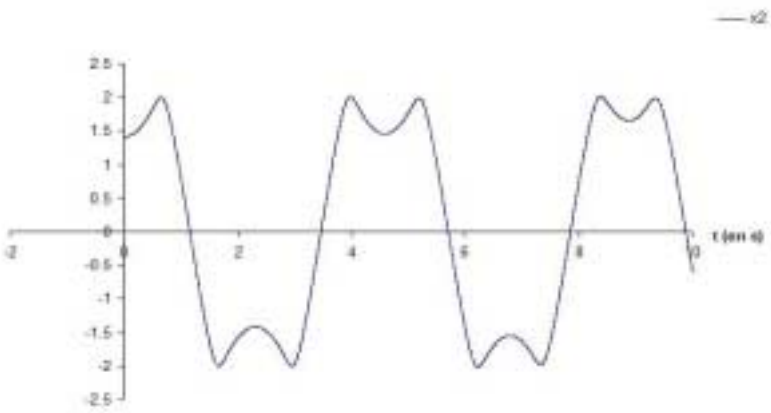


Figure VI-15 – Trajectoire de l'extrémité de la manivelle

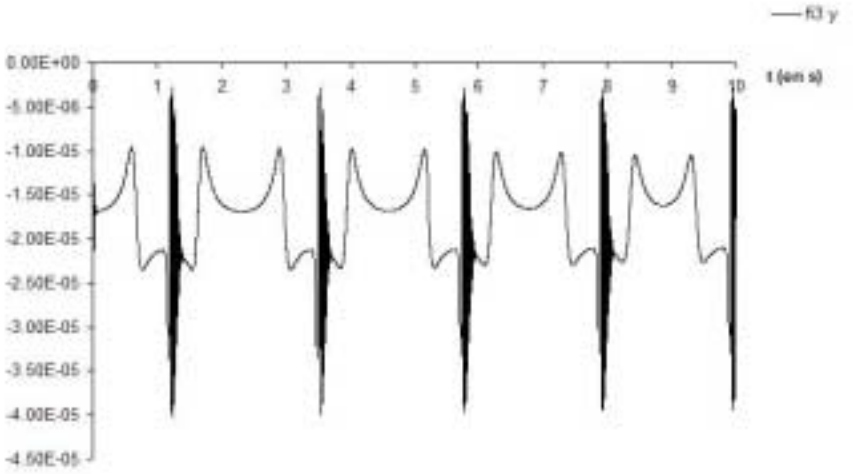


Figure VI-16 – Violations de contraintes fi3 y

Les violations de contraintes sur la position de la masse m_2 sont présentées Figure VI-16. Cette erreur sur la position est de l'ordre de 10^{-5}.

VI.3.2 Pendule simple avec changements de contraintes

Cette section présente un exemple 2D de mise en œuvre de la modélisation modulaire pour un système soumis à des changements de topologie inspiré de l'exemple proposé par [Dje99].

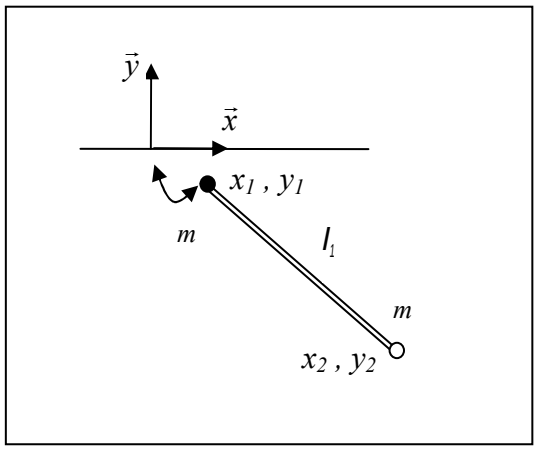


Figure VI-17 – Modèle de pendule simple

Le système est constitué d'un pendule simple représenté par deux masses m liées par une tige rigide sans masse de longueur ℓ. Le paramétrage retenu est (x_1, y_1, x_2, y_2), coordonnées cartésiennes des extrémités du pendule, représentées sur la Figure VI-17 par les points 1 et 2, respectivement en noir et blanc. Les conditions initiales sont les suivantes:

- le point 1 est à l'origine du repère de coordonnées $(0,0)$.
- le point 2 a pour coordonnées $(0,l)$.
- le système est lâché à vitesse nulle.
- le point 1 est lié au bâti par une liaison pivot d'axe $\vec{z}$.

A chaque fois que l'extrémité libre atteint l'axe $(O,\vec{x})$, les contraintes sont modifiées: l'extrémité libre devient contrainte et inversement.

Ce système représenté par les quatre paramètres x_1, y_1, x_2 et y_2 possède un seul degré de liberté. Il faut donc introduire trois équations de contraintes. Conformément à la définition de la modélisation modulaire, elles sont de deux types :

- deux équations algébriques pour traduire l'appartenance du point lié à l'axe $(O,\vec{x})$.
- une équation de contrainte de corps rigides exprimant la distance constante entre les points 1 et 2.

La Figure VI-18 représente les trajectoires des deux extrémités du pendule.

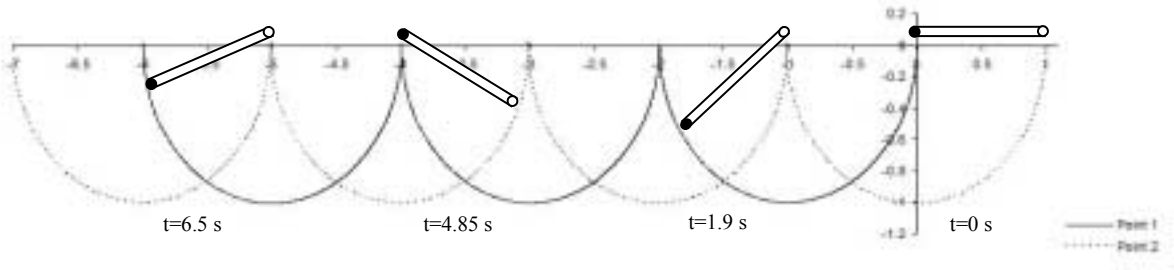


Figure VI-18 – Trajectoires des extrémités du pendule simple à changements de contraintes

La méthode proposée par [Dje99] consiste également en l'ajout ou la suppression de contraintes pour le système. La technique utilisée nécessite cependant une modification du paramétrage lors du changement de topologie. Ce modèle est donc plus figé car construit pour répondre à ce problème précis de modification des contraintes. Avec la modélisation modulaire, aucun calcul n'est réalisé lors du changement de topologie. Une suppression simple de la liaison est par exemple possible, sans modification du modèle, libérant ainsi la tige soumise alors à la seule pesanteur.

VI.4 Exemple 3D

La validation 3D de la méthode modulaire est réalisée au travers de l'étude du double pendule présenté sur la Figure VI-19 (représentation FER/Mech associée Figure VI-20). Le système est constitué de corps rigides de masses réparties m sur toute leur longueur ℓ.

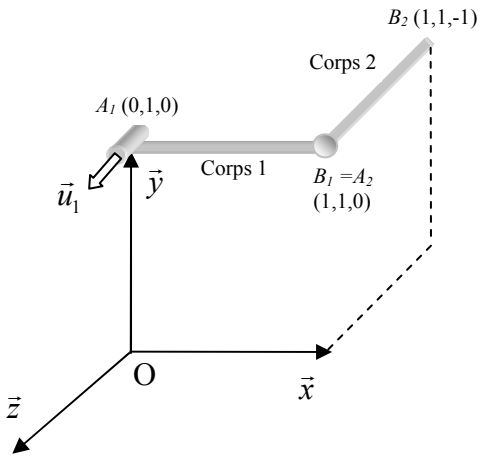


Figure VI-19 – Position initiale du double pendule 3D

Le corps 1 est lié au bâti par une liaison pivot d'axe $\vec{z}$. Les corps 1 et 2 sont liés par une liaison rotule. L'ensemble est soumis à la seule force de gravité $\vec{F}_g = -mg\vec{y}$. La position initiale est la suivante:

- l'extrémité B_1 du corps C_1 a pour coordonnées $(1,1,0)$.
- l'extrémité B_2 du corps C_2 a pour coordonnées $(1,1,-1)$

Le système est lâché à vitesse nulle à l'instant initial.

Le système est décrit à l'aide des coordonnées naturelles par l'intermédiaire de deux points et deux vecteurs conformément à la représentation définie au chapitre III. Chaque corps est donc représenté par douze coordonnées. Seul est représenté sur la Figure VI-19 le vecteur $\vec{u}_1$ nécessaire à la définition de la liaison pivot d'axe $\vec{z}$. Le double pendule est donc décrit par 24 coordonnées naturelles. Le système possède quatre degrés de liberté et nécessite donc l'introduction de 20 équations de contraintes qui, suivant la définition de la modélisation modulaire, se décomposent en:

- six contraintes internes pour chaque corps traduisant l'appartenance des coordonnées naturelles à un corps rigide.
- cinq équations de contraintes de liaison pour la liaison pivot.
- trois équations de contraintes externes pour la liaison rotule.

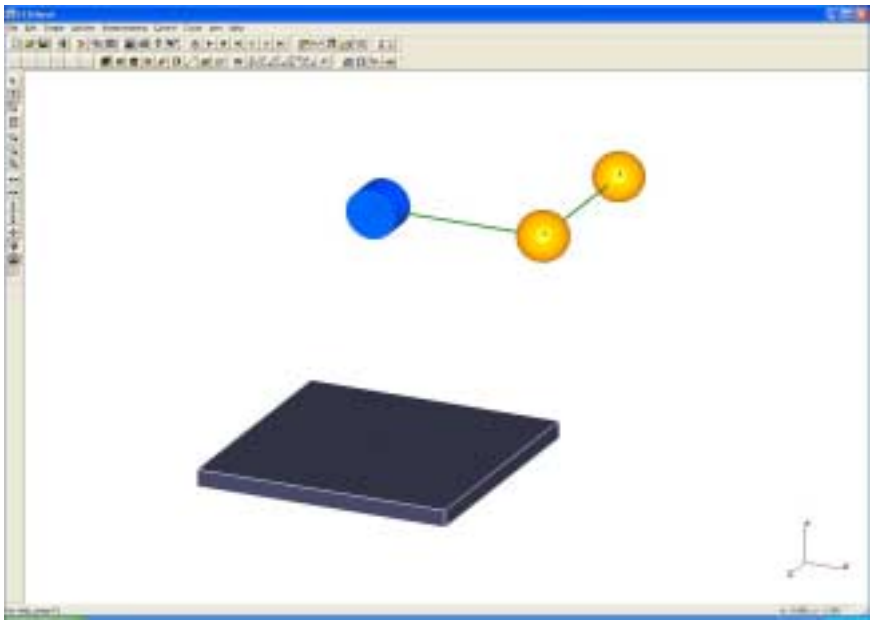


Figure VI-20 – Représentation FER/Mech du pendule double 3D

- coordonnées de l'extrémité du premier pendule

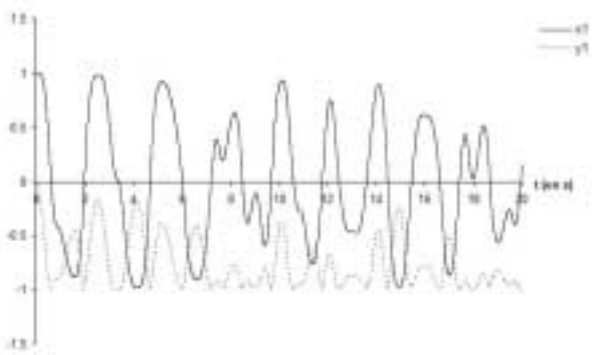


Figure VI-21 – Coordonnées x1 et y1 de l'extrémité du premier pendule

Figure VI-22 – Coordonnées z1 de l'extrémité du premier pendule

Les variations de position dans la direction z restent très faibles malgré les sollicitations du deuxième pendule dans ces directions (de l'ordre de 1/1000).

- coordonnées de l'extrémité du deuxième pendule

Figure VI-23 – Coordonnées x2 et y2 de l'extrémité du deuxième pendule

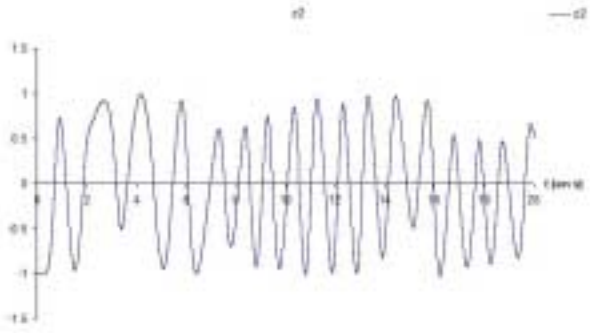


Figure VI-24 – Coordonnées z2 de l'extrémité du deuxième pendule

- Violations des contraintes de liaison

Pour le cas de la liaison pivot, seuls sont représentées les violations de contraintes fi1_x, fi1_y et fi1-z.

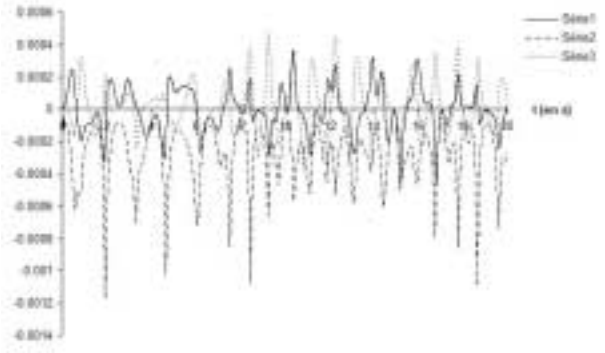


Figure VI-25 – Violations de contraintes en position pour la liaison 1

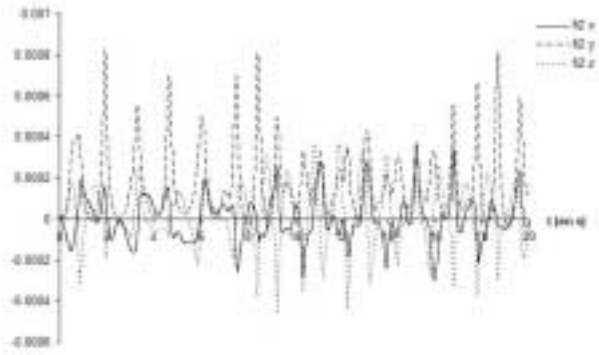


Figure VI-26 – Violations de contraintes en position pour la liaison 2

VI.5 FER/Mech interactif

Dans une phase plus avancée du développement de l'outil de conception, le solver modulaire a été incorporé au post-processeur de FER/Mech pour une résolution et un affichage simultané de la position des systèmes. Cette modification logicielle autorise alors de pouvoir agir en ligne pour modifier la topologie du système. A chaque pas de temps de résolution, le solver se renseigne auprès de l'interface graphique sur les modifications éventuelles apportées au système. Cette incorporation dans la boucle de résolution peut être schématisée comme sur la Figure VI-27, déjà présentée au chapitre I.

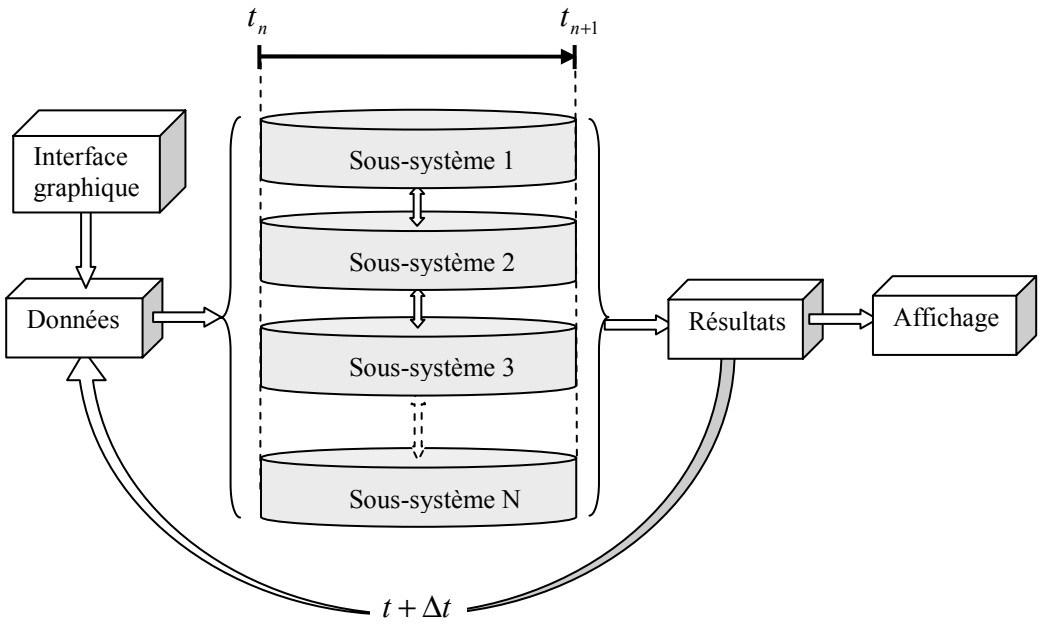


Figure VI-27 – Approche modulaire

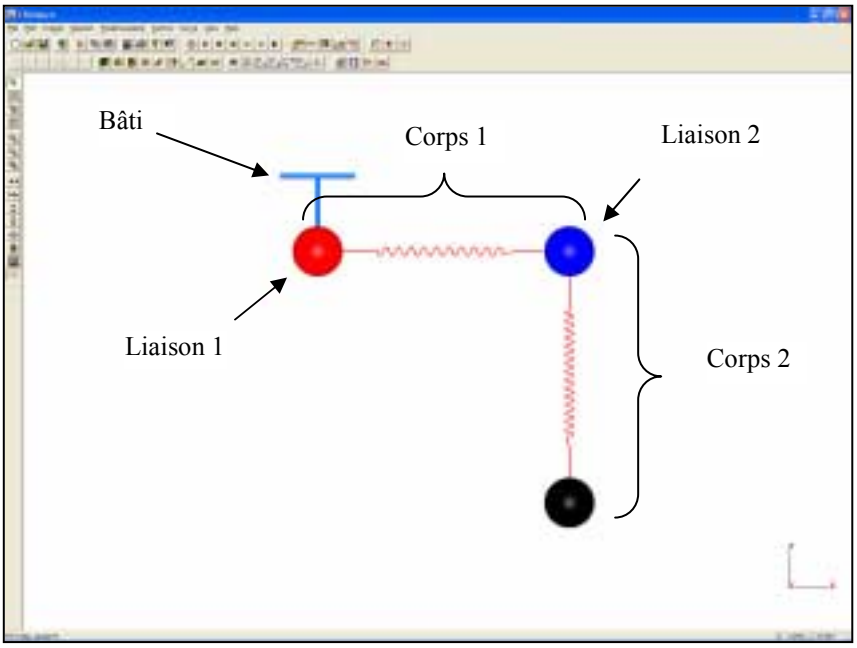


Figure VI-28 – Pendule double pour FER/Mech interactif

Présenté Figure VI-28, le système est composé de deux tiges rigides sans masse de longueur $\ell = 1\text{m}$ affectées à leurs extrémités de masse ponctuelles de valeur $m = 1\text{kg}$. Les deux liaisons Bâti – corps1 et corps1 – corps2 sont des liaisons pivots.

Deux icônes ont été directement rattachés aux destructeurs des liaisons 1 et 2 (Figure VI-29) afin de simuler un événement capable de modifier la topologie, dans ce cas, la coupure d'une liaison.

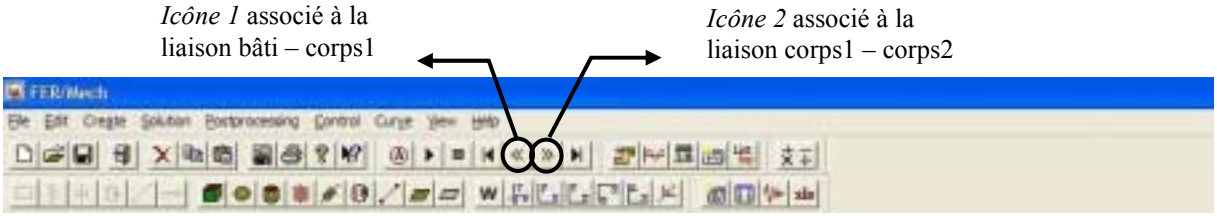


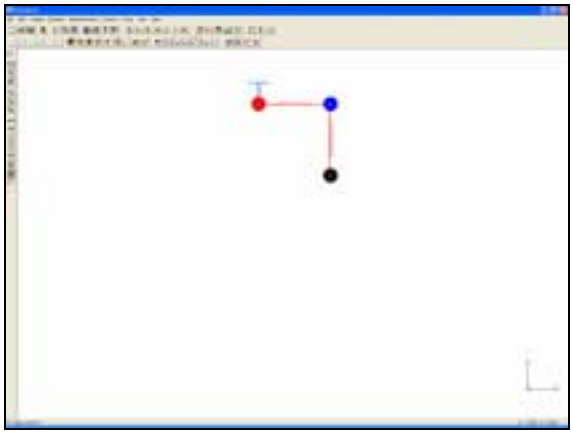
Figure VI-29 – Barre d'outil FER/Mech : Icônes de liaisons

Remarque : Dans l'état actuel, le développement est très spécifique à l'exemple présenté, le but étant uniquement de montrer la simplicité de la mise en oeuvre. Les modifications ne concernent en effet que la programmation de l'interface mais aucunement le modèle lui-même. Les versions ultérieures de FER/Mech prévoient la création d'une fenêtre figurant l'architecture du système multicorps dans un schéma arborescent représentant les éléments de la structure (corps et liaison). Il est également envisagé, à l'aide de menus contextuels, d'accéder aux propriétés de ces éléments (taille, masse, nature des laisons…) et de permettre la création ou la suppression d'éléments.

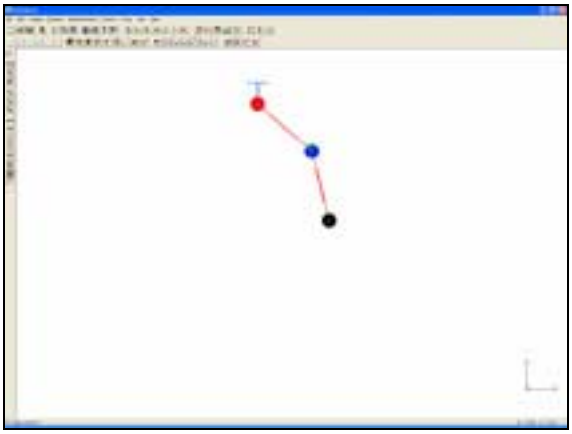
Les deux figures suivantes (Figure VI-30 et Figure VI-31) montrent la succession des positions respectivement aux instants t_0 à t_5, et t_6 à t_{11}.

A l'instant t_5, une coupure de la liaison 1 est réalisée par activation de l'icône 1. Le corps 2 est donc soumis à la seule gravité. Puis, à l'instant t_8, l'icône est actionné afin de libérer la liaison 1. Le corps 1 est à son tour libre de toute contrainte.

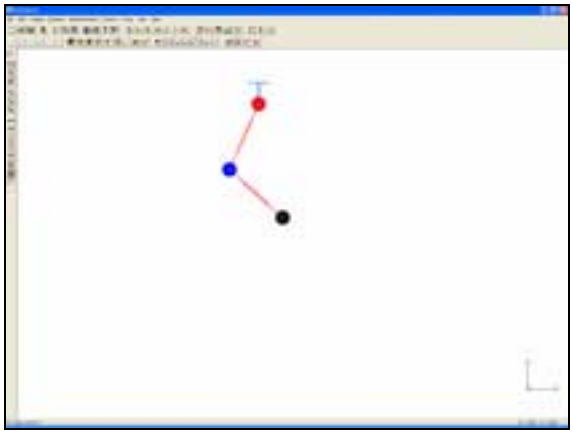
t_0 : position initiale



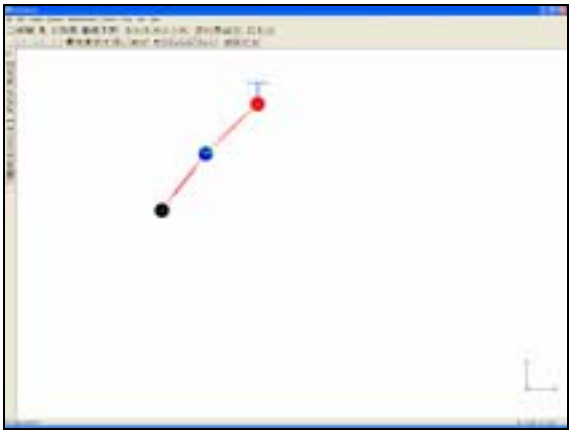
t_1



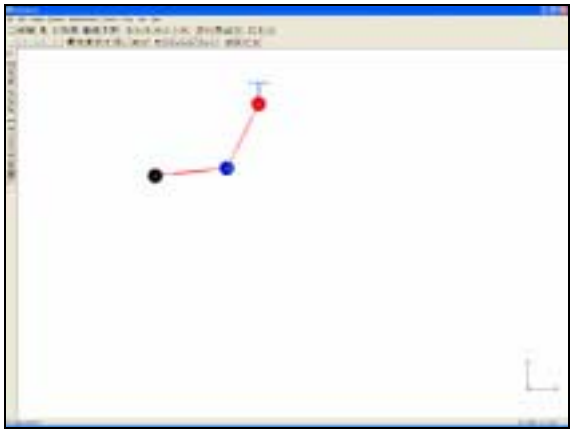
t_2



t_3



t_4



t_5 : Suppression de la liaison corps 1 – corps 2

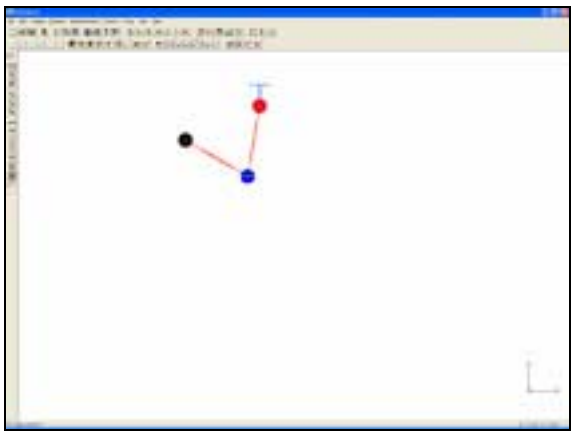
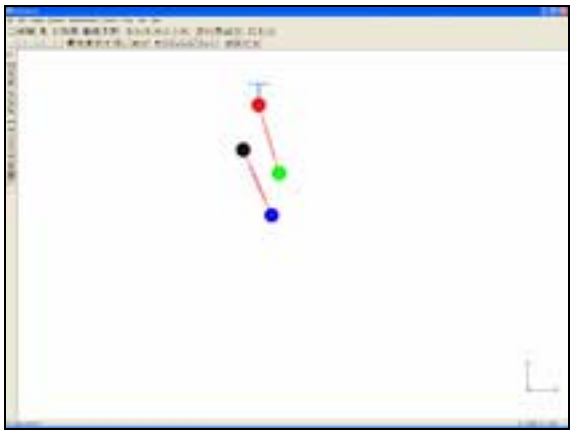
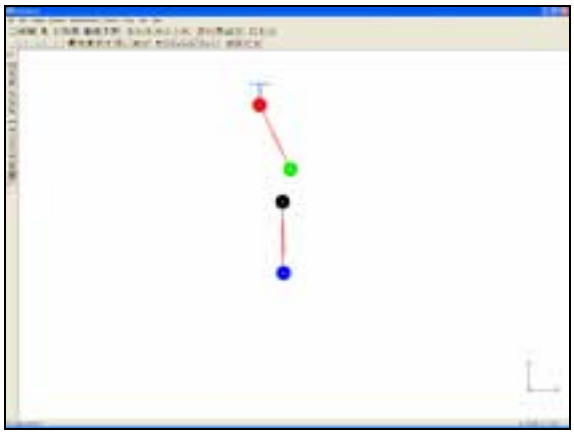


Figure VI-30 – Simulation pendule double : instants t_0 à t_5

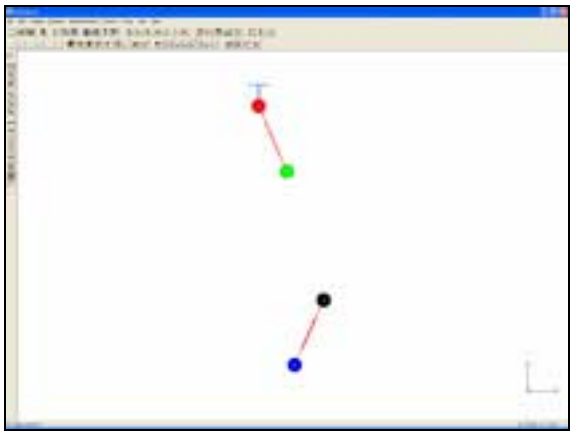
t_6 : début de chute libre du corps 2



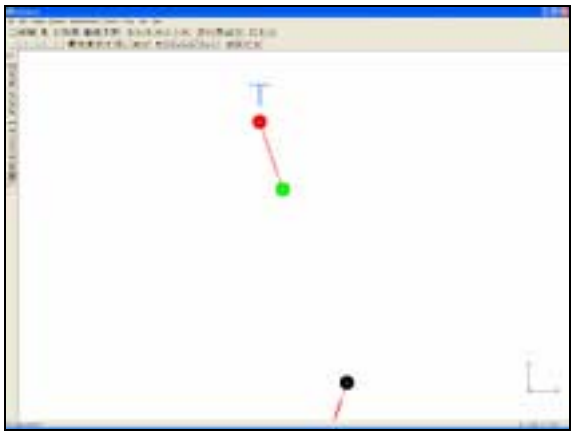
t_7



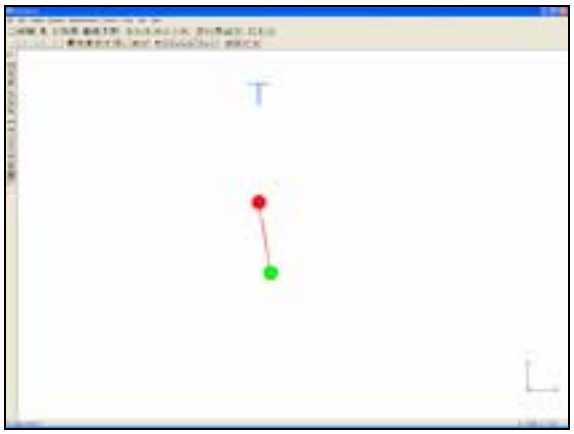
t_8 : suppression de la liaison corps 1 - bâti



t_9 : début de chute libre du corps 1



t_{10}



t_{11}

Figure VI-31 – Simulation pendule double : instants t_6 à t_{11}

VI.6 Conclusion

Les exemples présentés permettent de valider la modélisation et la résolution modulaire. Le pendule simple soumis à des changements de topologie, ainsi que l'exemple développé pour FER/Mech interactif, montrent l'apport de la méthode pour l'ajout ou la suppression de liaisons.

La méthode du Lagrangien augmenté pour les contraintes internes garantit la cohésion des corps. En pratique, le critère d'arrêt de la méthode est plutôt basé sur un nombre d'itérations. Quelques unes (deux à trois) suffisent à assurer la convergence. Des itérations supplémentaires n'améliorent en effet pas les résultats conformément aux résultats présentés par [Bay94].

A travers ces exemples, la mise en pratique du critère de stabilité, défini au chapitre IV est validée. Pour des pas de temps de simulation différents (de 10^{-4} à 10^{-2}) et pour des durées de simulation relativement important (≈30s), il n'apparaît pas d'instabilité numérique. Par contre, plus le pas de temps augmente, plus les liaisons externes sont flexibles. Les violations des contraintes sont dans ce cas importantes conformément à nos attentes.

Conclusion générale

1. Conclusion

La méthode retenue pour construire un outil de prototypage rapide avec possibilité d'une interaction en dynamique avec le mécanisme virtuel est basée sur une décomposition en composants mécaniques (*composants corps* et *composants liaisons*) de la chaîne cinématique. Cette décomposition permet de construire un modèle modulaire où chaque composant possède ses propres caractéristiques en terme de représentation ou de résolution.

Ce formalisme autorise donc divers paramétrages (translation-rotation, coordonnées naturelles, canoniques,….) et conduit à des expressions locales des équations de la dynamique possédant alors diverses propriétés (matrice masse constante, absence de termes centrifuges et de Coriolis, expressions des liaisons simplifiées).

Les paramétrages conduisant à ce type de propriétés ne sont pas minimaux et font donc intervenir des équations de contraintes nommées *internes*, traduisant la rigidité des corps. La topologie du système est quant à elle construite par ajout de contraintes entre ces corps appelées contraintes *externes*. Contrairement aux méthodes classiques, ces deux types de contraintes sont traités de manières différentes:

- les contraintes ex*ternes* sont calculées de manière explicite à l'aide d'une méthode de pénalité.
- les contraintes *internes* sont résolues de manière itérative lors de la résolution des équations de la dynamique par une méthode de lagrangien augmenté.

Les expressions explicites des contraintes internes conduisent à de l'instabilité et nécessitent l'introduction de critères numériques afin de calculer, pour un pas de temps donné, les facteurs de pénalités à attribuer à chacune des liaisons.

Sous cette forme, la modélisation est dite *modulaire*. Ainsi, les composants définis pour modéliser un système articulé sont proches des éléments physiques (corps et liaisons)

constituant la structure du système polyarticulé étudié. Ils conduisent ainsi directement à une définition des objets informatiques.

Les résultats présentés illustrent la mise en oeuvre de la méthode pour divers exemples. La simulation du double pendule permet de comparer les résultats obtenus par résolution modulaire avec les résultats obtenus à l'aide d'une méthode centralisée classique, et le système bielle-manivelle pour le cas d'une chaîne cinématique fermée. Les deux exemples de pendule simple et double à changements de topologie mettent en évidence les propriétés de flexibilité et la possibilité de modifier le scénario de simulation par un simple événement produit par l'utilisateur. Enfin, le dernier exemple du double pendule est un cas d'usage dans l'espace 3D.

Le traitement des efforts des liaisons externes de manière explicite est intrinsèquement efficace du point de vue du temps de calcul. De plus, nous avons constaté qu'à travers un critère simple, il est possible de contrôler la stabilité numérique en fonction uniquement du pas de temps de simulation et de la masse vue par la liaison à chaque instant. Cependant, lorsque le pas de temps augmente, la souplesse des liaisons augmente entraînant des violations de contrainte qui peuvent être importantes. Ce phénomène est d'autant plus vrai si, en même temps, la masse vue par la liaison externe diminue. Cela peut se produire en présence de composants de faible masse ou en présence de transmission d'efforts de liaisons importants dus à un couplage cinématique.

2. Perspectives

Afin de remédier à ce problème de violation de contraintes, une première solution possible est d'augmenter la masse vue en fonction de la violation de contraintes (constatée au niveau des accélérations) par un processus local et itératif. Il se posera alors le problème du contrôle de la stabilité numérique.

Une deuxième solution serait de considérer des composants évolutifs par assemblage de composants élémentaires en cours de calcul. Les liaisons externes entre ces composants deviendraient des liaisons internes au composant évolutif. Dans ce cas, le traitement de ces liaisons pourrait se faire de manière implicite avec des masses vues augmentées.

Une dernière solution serait, comme vu au chapitre 1 (algorithme de *'gluing'*), d'utiliser des techniques de sous-structuration mais avec, dans ce cas, une résolution au niveau des liaisons externes à l'aide d'un processus centralisé s'éloignant ainsi d'une approche véritablement modulaire. Dans le cas où l'interactivité en cours de simulation n'est pas essentielle (cas de la conception mécanique de systèmes complexes), la dernière approche est suffisante. Dans le cas contraire (cas de la réalité virtuelle), la première approche serait la meilleure en recherchant un compromis entre précision et temps de calcul. La deuxième approche constituerait une solution intermédiaire à considérer.

Afin d'étendre les possibilités de la simulation, les perspectives plus générales suivantes sont proposées.

Pour les exemples présentés, les systèmes sont uniquement composés de corps rigides simples. Cependant, les principes de modularité doivent permettre d'étendre ce mode de résolution à tout type de représentation afin de coupler non seulement des corps simples mais aussi des assemblages de corps déjà constitués possédant leurs propres paramétrages. De manière plus générale, tout paramétrage pour lequel il est possible d'identifier des éléments d'interface, de type points ou vecteurs et assurant l'écriture simple de liaisons conviennent. Ainsi des corps flexibles en représentation éléments finis, possédant leurs propres méthodes de résolution adaptées, peuvent ainsi être assemblés à des assemblages de corps rigides.

L'interfaçage doit également permettre à terme de saisir un objet de la scène et de le manipuler. L'action de saisie est susceptible de provoquer des discontinuités de vitesses nécessitant alors le traitement de chocs. De manière plus générale, afin de rendre la scène virtuelle plus réaliste, les développements nécessitent l'incorporation de deux éléments:

- un dispositif de détection de collision.
- le traitement des chocs mis en évidence par le détecteur.

Afin de respecter les critères de modularité, ces traitement doivent être réalisés localement au niveau des composant *corps* ou *liaison*.

Concernant les développements informatiques, deux points peuvent être développés:

- les propriétés de la programmation orientée objets ont été utilisées de manière basique avec une architecture à deux classes principales, une classe *Body* et une classe *Joint,*

redéfinies pour chaque mécanisme et chaque exemple présentés dans ce mémoire. Afin de construire de manière automatique la topologie du système, de nouvelles classes d'objets, dérivées de la classe *Joint*, peuvent être définies pour une prise en compte automatique de chaque type de liaison.

- une attention doit être apportée à l'optimisation des calculs, parfois redondants dans les développements actuels, afin de limiter le nombre d'échange d'informations entre entités et donc de diminuer les temps de résolution.

La modularité pour l'ajout de nouvelles représentations et l'optimisation des calculs doivent donc également guider la construction de l'architecture du logiciel.

L'incorporation de ces développements au logiciel déjà existant peut faire de Fer/Mech un outil complet de prototypage rapide adapté à tout type de représentation où l'utilisateur construit et agit de manière interactive sur la scène virtuelle.

Annexe 1: Coordonnées naturelles

Cette annexe présente les développements pour l'obtention de la matrice masse pour un corps rigide représenté à l'aide des coordonnées naturelles [Gar86]. L'exemple retenu est le plus général et concerne un corps rigide dans l'espace 3D représenté par deux points et deux vecteurs unitaires non-coplanaires.

Coordonnées naturelles : calcul de la matrice masse

Soit un corps rigide S sur lequel sont défini deux points A et B ainsi que deux vecteurs unitaires $\vec{u}$ et $\vec{v}$. Soient $\Re$ un repère galiléen et $\Re_0$ un repère lié au solide.

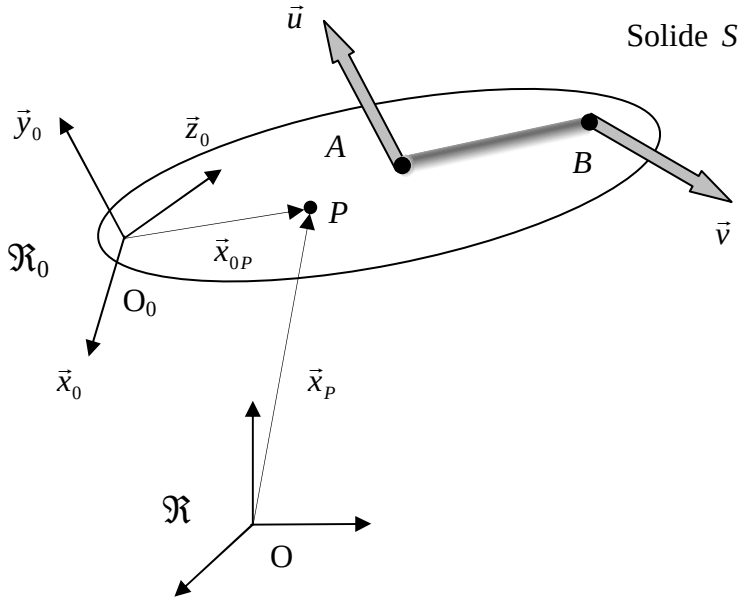


Figure A1-1 – Solide S en coordonnées naturelles

Pour ce calcul de la matrice masse, les caractéristiques suivantes sont supposées connues:

- coordonnées locales des points *A* et *B* ainsi que des vecteurs $\vec{u}$ et $\vec{v}$ dans le repère $\Re_0$.

- position du centre d'inertie du corps dans le repère $\Re_0$.

- masse *m* et matrice d'inertie J_0 exprimée dans le repère $\Re_0$.

Pour tout point P du solide S, $\vec{x}_p$ désigne la position de P par rapport au repère $\Re$ et $\vec{x}_{0P}$ est la position P par rapport au repère $\Re_0$.

Les vecteurs $\overrightarrow{AB}$, $\vec{u}$ et $\vec{v}$ n'étant pas coplanaires, ils forment une base de l'espace 3D. La position du point P s'écrit alors dans cette base:

$$\vec{x}_P = \vec{x}_A + c_1\left(\vec{x}_B - \vec{x}_A\right) + c_2\vec{u} + c_3\vec{v} \tag{A1.1}$$

ou encore sous forme matricielle:

$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \begin{bmatrix} (1-c_1)\mathrm{I}_3 & c_1\mathrm{I}_3 & c_2\mathrm{I}_3 & c_3\mathrm{I}_3 \end{bmatrix} \begin{Bmatrix} x_A \\ x_B \\ u \\ v \end{Bmatrix} = Cq \tag{A1.2}$$

où

- q est le vecteur des coordonnées naturelles.
- C est une matrice indépendante du temps (c_1, c_2 et c_3 sont les coordonnées locales).

L'énergie cinétique du solide est donnée par l'expression:

$$Ec = \int_S (\dot{x}_P)^T (\dot{x}_P) dm \tag{A1.3}$$

La matrice C étant constante, alors $\dot{x}_P = C\dot{q}$, donc:

$$E_c = \frac{1}{2}\dot{q}^T \left(\int_S C^T C dm \right) \dot{q} \tag{A1.4}$$

En exprimant C en fonction des paramètres connus que sont les coordonnées des points et des vecteurs dans le repère local $\Re_0$, il vient:

$$\vec{x}_{0P} - \vec{x}_{0A} = c_1 \left(\vec{x}_{0B} - \vec{x}_{0A} \right) + c_2 \vec{u}_0 + c_3 \vec{v}_0 \tag{A1.5}$$

ou encore sous forme matricielle:

$$\begin{Bmatrix} x_{0P} \\ y_{0P} \\ z_{0P} \end{Bmatrix} - \begin{Bmatrix} x_{0A} \\ y_{0a} \\ z_{0A} \end{Bmatrix} = \begin{bmatrix} x_{0B} - x_{0A} & u_0 & v_0 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \end{Bmatrix} = X_0 \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \end{Bmatrix} \tag{A1.6}$$

où

- X_0 est la matrice formée en colonne des composantes de $\vec{x}_{0B} - \vec{x}_{0A}$, $\vec{u}_0$ et $\vec{v}_0$ dans $\Re_0$.

Pour obtenir l'énergie cinétique, il faut donc déterminer les dix paramètres de la matrice symétrique $M = C^T C$ suivante:

$$M = \int_S \begin{bmatrix} (1 - 2c_1 + c_1^2)\mathrm{I}_3 & (c_1 - c_1^2)\mathrm{I}_3 & (c_2 - c_1 c_2)\mathrm{I}_3 & (c_3 - c_1 c_3)\mathrm{I}_3 \\ & c_1^2 \mathrm{I}_3 & c_1 c_2 \mathrm{I}_3 & c_1 c_3 \mathrm{I}_3 \\ & & c_2^2 \mathrm{I}_3 & c_2 c_3 \mathrm{I}_3 \\ & & & c_3^2 \mathrm{I}_3 \end{bmatrix} dm \tag{A1.7}$$

et calculer les termes c_i dont l'expression est:

$$c_i = [\mathrm{X}_0]_i^{-1}(x_{0P} - x_{OA}) \tag{A1.8}$$

où $[X_0]_i^{-1}$ est la ième ligne de $X_0{}^{-1}$.

Les termes de la matrice se calculent alors de la façon suivante:

- $\int_s dm = m$
- $\int_s c_i dm = \int_s [X_o^{-1}]_i (x_{oP} - x_{0A}) dm = [X_o^{-1}]_i . \int_s x_{oP} dm$

par définition du centre d'inertie, $\int_s x_{oP} dm = m x_{oG}$, donc:

$$\int_s c_i . dm = [X_0]_i^{-1} m(x_{oG} - x_{0A}) = m a_i \tag{A1.9}$$

- $\int_s c_i c_j . dm = \int_s \left([\mathrm{X}_o^{-1}]_i (x_{oP} - x_{0A})\right)\left([\mathrm{X}_o^{-1}]_j (x_{oP} - x_{0A})\right) dm$

$$\int_s c_i c_j . dm = [\mathrm{X}_o^{-1}]_i J [\mathrm{X}_o^{-1}]_j{}^T = z_{ij} \tag{A1.10}$$

avec la matrice *J* définie comme suit :

$$J = \int_s (\mathrm{x}_{oP} - x_{0A})(\mathrm{x}_{oP} - x_{0A})^T dm = \frac{1}{2}\mathrm{Tr}(J_A)\mathrm{I} - J_A \tag{A1..11}$$

où J_A est la matrice d'inertie du corps au point *A*.

Ainsi, la matrice masse prend la forme:

$$M = \begin{bmatrix} (m-2ma_1+z_{11})I_3 & (ma_1-z_{11})I_3 & (ma_2-z_{12})I_3 & (ma_3-z_{13})I_3 \\ (ma_1-z_{11})I_3 & z_{11}I_3 & z_{12}I_3 & z_{13}I_3 \\ (ma_2-z_{12})I_3 & z_{21}I_3 & z_{22}I_3 & z_{23}I_3 \\ (ma_3-z_{13})I_3 & z_{31}I_3 & z_{32}I_3 & z_{33}I_3 \end{bmatrix} \quad \text{(A1.12)}$$

M ne dépend que des coordonnées locales des points et des vecteurs définissant les coordonnées naturelles: la matrice masse M est constante.

Annexe 2: Contraintes et liaisons

Les coordonnées naturelles ayant été principalement étudiées, cette annexe détaille l'expression des contraintes de liaison et des jacobiennes associées pour les principales liaisons utilisées dans les exemples présentés au chapitre V. Les corps sont représentés de manière générique par deux points et deux vecteurs correspondants aux centres et aux axes des liaisons. Pour toutes ces liaisons, les expressions sont données pour deux corps C_j et C_k liés par la liaison i.

Le vecteur q_j des coordonnées associées au corps C_j est noté :

$$q = \left(x_{A_j} \quad y_{A_j} \quad z_{A_j} \quad x_{B_j} \quad y_{B_j} \quad z_{B_j} \quad x_{u_j} \quad y_{u_j} \quad z_{u_j} \quad x_{v_j} \quad y_{v_j} \quad z_{v_j} \right)^T$$

A2.1 Liaison sphérique

La liaison sphérique, ou rotule, possède trois degrés de liberté de rotation (*d = 3*).

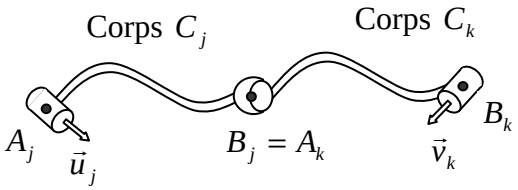


Figure A2-1 – Liaison sphérique

L'expression III.9 exprimant le blocage des trois translations relatives s'écrit:

$$\phi_i(q_j, q_k) = \begin{cases} x_{Bj} - x_{A_k} = 0 \\ y_{Bj} - y_{A_k} = 0 \\ z_{Bj} - z_{A_k} = 0 \end{cases} \qquad \text{(A2.1)}$$

Les jacobiennes associées intervenant dans le calcul explicite des efforts extérieurs sont donc:

$$J_{ij} = \left[\begin{array}{ccc|c|c|c} 1 & 0 & 0 & & & \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & & & \end{array}\right] \qquad (a)$$

$$J_{ik} = \left[\begin{array}{c|ccc|c|c} & -1 & 0 & 0 & & \\ 0 & 0 & -1 & 0 & 0 & 0 \\ & 0 & 0 & -1 & & \end{array}\right] \qquad (b) \qquad \text{(A2.2)}$$

A2.3 Liaison pivot-glissant

Une liaison cylindrique ou pivot-glissant possède deux degrés de liberté, un en translation et l'autre en rotation (*d = 2*).

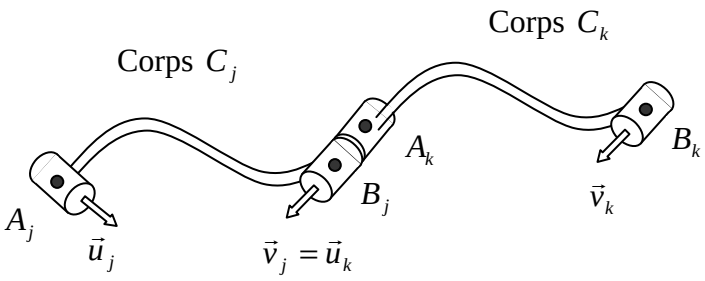


Figure A2.2 – Liaison pivot-glissant

Les équations III.10.a et b deviennent :

$$\begin{cases} \vec{v}_j \times \vec{u}_k = 0 & (a) \\ \overrightarrow{B_j A_k} \times \vec{v}_j = 0 & (b) \end{cases} \tag{A2.3}$$

Pour chaque relation issue d'un produit vectoriel, deux seulement des trois équations algébriques sont indépendantes donc la fonction vectorielle $\phi_i(q_j, q_k)$ peut prendre par exemple la forme :

$$\phi_i(q_j, q_k) = \begin{cases} y_{v_j} z_{u_k} - z_{v_j} y_{u_k} = 0 \\ z_{v_j} x_{u_k} - x_{v_j} z_{u_k} = 0 \\ (y_{A_k} - y_{B_j}) z_{v_j} - (z_{A_k} - z_{B_j}) y_{v_j} = 0 \\ (z_{A_k} - z_{B_j}) x_{v_j} - (x_{A_k} - x_{B_j}) z_{v_j} = 0 \end{cases} \tag{A2.4}$$

donc la jacobiennes associées :

$$J_{ij} = \left[\begin{array}{ccc|ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -z_{v_j} & y_{v_j} & 0 & 0 & 0 \\ 0 & 0 & 0 & z_{v_j} & 0 & -x_{v_j} & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{ccc} 0 & z_{u_k} & -y_{u_k} \\ -z_{u_k} & 0 & x_{u_k} \\ 0 & -(z_{A_k} - z_{B_j}) & (y_{A_k} - y_{B_j}) \\ (z_{A_k} - z_{B_j}) & 0 & -(x_{A_k} - x_{B_j}) \end{array}\right] \tag{A2.5}$$

$$J_{ik} = \left[\begin{array}{ccc|ccc|ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & -z_{v_j} & y_{v_j} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & z_{v_j} & 0 & -x_{v_j} & 0 & 0 & 0 \\ 0 & z_{v_j} & -y_{v_j} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -z_{v_j} & 0 & x_{v_j} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

A2.2 Liaison pivot

Pour la liaison pivot, il convient d'ajouter la contrainte traduisant que la distance entre les points B_j et A_k reste constante et exprimée par la relation scalaire, donnée par la relation III.11:

$$\frac{1}{2}\left((x_{A_k} - x_{B_j})^2 + (y_{A_k} - y_{B_j})^2 + (z_{A_k} - z_{B_j})^2\right) = k \qquad \text{(A2.6)}$$

où k est une constante. Les lignes suivantes doivent donc être ajoutée respectivement à J_{ij} et J_{ik}.

$$\begin{bmatrix} 0 & 0 & 0 & -(x_{A_k} - x_{B_j}) & -(y_{A_k} - y_{B_j}) & -(z_{A_k} - z_{B_j}) & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (a)$$
$$\begin{bmatrix} (x_{A_k} - x_{B_j}) & (y_{A_k} - y_{B_j}) & (z_{A_k} - z_{B_j}) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (b) \qquad \text{(A2.7)}$$

A2.4 Liaison prismatique

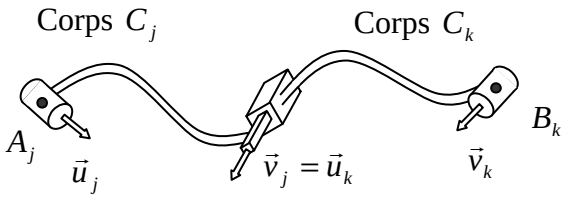


Figure A2.3 – Liaison prismatique

Une liaison prismatique est une liaison cylindrique à laquelle est ajoutée la contrainte supplémentaire, obtenue à partir du produit scalaire, équivalente de la relation III.12:

$$\overrightarrow{A_j B_j} \cdot \overrightarrow{A_k B_k} = k \qquad \text{(A2.8)}$$

où k est une constante. La forme développée est:

$$(x_{B_j} - x_{A_j})(x_{B_k} - x_{A_k}) + (y_{B_j} - y_{A_j})(y_{B_k} - y_{A_k}) + (z_{B_j} - z_{A_j})(z_{B_k} - z_{A_k}) = k$$

Les lignes suivantes doivent donc être ajoutée respectivement à J_{ij} et J_{ik}.

$$\left[-(x_{B_j}-x_{A_j}) \quad -(y_{B_j}-y_{A_j}) \quad -(z_{B_j}-z_{A_j}) \middle| (x_{B_j}-x_{A_j}) \quad (y_{B_j}-y_{A_j}) \quad (z_{B_j}-z_{A_j}) \middle| 0 \quad 0 \quad 0 \middle| 0 \quad 0 \quad 0\right]$$

$$\left[-(x_{B_k}-x_{A_k}) \quad -(y_{B_k}-y_{A_k}) \quad -(z_{B_k}-z_{A_k}) \middle| (x_{B_k}-x_{A_k}) \quad (y_{B_k}-y_{A_k}) \quad (z_{B_k}-z_{A_k}) \middle| 0 \quad 0 \quad 0 \middle| 0 \quad 0 \quad 0\right]$$

Bibliographie

[And98] K. S. Anderson and S. Duan, *A hybrid parallelizable low-order alogorithm for dynamics of multi-rigid-body systems : Part I, Chain systems,* Mathematical and Computer Modelling, Vol. 30, pp. 193-215, 1999.

[Arm79] W. W. Armstrong, *Recursive solution of the equations of motion of an N-link manipulator,* Proceedings of the fifth World Congress on the Theory of Machines and Mechanism, Vol. 2, pp. 1342-1346, Montréal, Canada, 1979.

[Asc97] U. M. Ascher, D. K. Pai and B. P. Cloutier, *Forward dynamics, elimination methods and formulation stiffness in robot simulation,* Int. J. Robotics Res., Vol. 16, No. 6, pp. 749-758, Dec. 1997.

[Bae87a] D. S. Bae and E. J. Haug, *A recursive formulation for constrained mechanical system dynamics: Part I, open loop systems,* Mechanisms, Structures and Machines, 15 (3), pp. 359-382, 1987.

[Bae87b] D. S. Bae and E. J. Haug, *A recursive formulation for constrained mechanical system dynamics: Part II, closed loop systems,* Mechanisms, Structures and Machines, 15 (4), pp. 481-506, 1987.

[Bae87c] D. S. Bae and E. J. Haug, *A recursive formulation for constrained mechanical system dynamics: Part III, parallel processor implementation,* Mechanisms, Structures and Machines, 15 (4), pp. 481-506, 1987.

[Bar93] A. Barraco, B. Cuny, *Dynamique des systèmes évolutifs,* 1er Colloque National en Calcul de Structures, Giens, Mai, 1993.

[Bau72] J. Baumgarte, *Stabilisation of constraints and integrals of motion in dynamical systems,* Comp. Meth. in Appl. Mech. and Eng., 1:1-16, 1972.

[Bay88] E. Bayo, J. Garcia de Jalon and M.A. Serna, *A modified lagrangian formulation for the dynamic analysis of constrained mechanical systems,* Computer Methods in Applied Mechanics and Engineering, Elsevier Science Publishers B.V., North-Holland, 1988.

[Bay91] E. Bayo, J. G. Garcia de Jalon, A. Avello and J. Cuadrado, *An efficient computational method for real time multibody dynamic simulation in fully cartesian coordinates,* Comput. Methods Appl. Mech. Engrg. 92, pp. 377-395, 1991.

[Bay94] E. Bayo, A. Avello, *Singularity-free augmented lagrangian algorithms for constrained multibody dynamics,* Nonlinear Dynamics 5 , pp. 209-231, 1994.

[Bel97] S. Belot, *GRAPS : Gradient propagation solving – une méthode de résolution par coopération pour le prototypage rapide,* Thèse de Doctorat, Université d'Evry, 1997.

[Bla02] W. Blajer, *Augmented lagrangian formulation: geometrical interpretation and its application to systems with singularities and redundancy*, Multibody System Dynamics, Vol. 8, Issue 2, pp. 117-140, Sept., 2002.

[Bra97] M. Brain, L. Lovette, *Developing professional applications for Windows 95 and NT using MFC,* Prentice Hall, 1997.

[Buc98] M. Buck and E. Schömer, *Interactive rigid body manipulation with obstacle contacts,* The Journal of Visualization and Computer Animation, Vol. 9, Issue 4, pp. 243-257, 1998.

[Cap94] D. M. Capper, *C++ for scientists, engineers and mathematicians,* Springer-Verlag, ISBN 0-387-19847-4, 1994.

[Che02] K.-Z. Chen, X.-A. Feng and L. Ding, *Intelligent approaches for generating assembly drawings from 3-D computer models of mechanical products,* Computer-Aided Design, Vol.. 34, pp. 347-355, 2002.

[Chu97] C.-C. P. Chu, T. H. Dani and R. Gadh, *Multi-sensory user interface for a virtual-reality-based computer-aided design system,* Computer-Aided Design, Vol. 29, No. 10, pp. 709-725, 1997.

[Cip81a] R. J. Cipra and J. J. Uicker Jr., *On the dynamic simulation of large nonlinear mechanical systems. Part.1 : An overview of the simulation techniques. Substructuring and frequency domain considerations*. Trans. ASME. Journal of Mechanical Design, Vol. 103, pp. 849-856, Oct., 1981.

[Cip81b] R. J. Cipra and J. J. Uicker Jr., *On the dynamic simulation of large nonlinear mechanical systems. Part.2 : The time intergation technique and time response loop.* Trans. ASME. Journal of Mechanical Design, Vol. 103, pp. 857-865, Oct., 1981.

[Cog97] L. Cogné, *Simulation de systèmes physiques lagrangiens : de la représentation symbolique aux évaluations numériques séquentielles et parallèles,* Thèse de Doctorat, Université de Rennes 1, Dec., 1997.

[Cua00] J. Cuadrado, J. Cardenal, P. Morer and E. Bayo, *Intelligent simulation of multibody dynamics : Space-state and descriptor methods in sequential and parallel computing environments,* Multibody System Dynamics, Vol. 4, pp. 55-73, Feb. 2000.

[Cur93] J.-H. Heegaard, A. Curnier, *An augmented lagrangian method for discrete large-slip contact problems,* Int. J. Numer. Methods Eng., Vol. 36, pp. 569-593. 1993.

[Den55] Denavit, J. and Hartenberg, R. S., *A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices,* Journal of Applied Mechanics, Vol. 22, pp. 215-221, June 1955.

[Dix99] D. S. Dixit, S. H. Shanbhag, S. P. Mudur, K. Isaac and S. Chinchalkar, *Object-orientd design of an interactive mechanism simulation system – Clodion,* Computers and Graphics, Vol. 23, pp. 85-94, 1999.

[Dje99] S. Djerassi, *Dynamic Analysis of systems with varying constraints,* Trans. of the ASME, Journal of Applied Mechanics, Vol. 66, Sept. 1999.

[Doh95] M. Dohmen, *A survey of constraint satisfaction techniques for geometric modelling,* Computer and Graphics, Vol. 19, N° 6, pp. 831-845, 1995.

[Dub93] Y. Dubois-Pèlerin and T. Zimmermann, *Object-oriented finite element programming: 3. An efficient implementation in C++*. Computer Methods in Applied Mechanics and Engineering, 1993, pp.165-183.

[Fat97] S. Fatikow, U. Rembold, *Microsystem technology and microrobotics,* Springer-Verlag, Berlin Heidelberg, 1997.

[Fea83] R. Featherstone, *The calculation of robot dynamics using articulated-body inertia,* Int. J. Robotics Res., Vol. 2, No. 2, pp. 13-30, 1983.

[Fea99a] R. Featherstone, *A divide-and-conquer articulated-body algorithm for parallel O(log(n)) calculation of rigid-body dynamics. Part.1 : Basic algorithm,* Int. J. Robotics Res., Vol. 18, No. 9, pp. 867-875, Sept., 1999.

[Fea99b] R. Featherstone, *A divide-and-conquer articulated-body algorithm for parallel O(log(n)) calculation of rigid-body dynamics. Part.2 : Trees, loops and accuracy,* Int. J. Robotics Res., Vol. 18, No. 9, pp. 867-875, Sept., 1999.

[Fen03a] Z.-Q. Feng, P. Joli, N. Séguy, *FER/Mech – A software with interactive graphics for dynamic of multibody system,* To appear in Advances in Engineering Software.

[Fen03b] Z.-Q. Feng, P. Joli, N. Séguy, *FER/Mech: Logiciel interactif d'aide à l'analyse des systèmes articulés,* 6ème Colloque National en Calcul des Structures, Giens, Mai, 2003.

[Fer95] J. Ferber, *Les systèmes Multi-Agents, Vers une intelligence collective,* InterEditions, Sept., 1995.

[Fij95] A. Fijany, I. Sharf and G. d'Eleuterio, *Parallel O(log N) algorithms for computation of manipulator forward dynamics,* IEEE Trans. on Robot. and Autom., Vol. 11, No. 3, pp. 389-400, June, 1995.

[Fis98] P. Fisette, J.M. Péterkenne, *Contribution to parallel and vector computation in*

multibody dynamics, Parallel Computing, Vol. 24, pp. 717-728, 1998.

[Fos90] R.O. Foshi, S.F. Stiemer and B.W.R. Forde, *Object-oriented finite element analysis*, Computer and Structures, pp.335-374, 1990.

[Füh90] C. Führer and R. Schwertassek, *Generation and solution of multibody system equations*, Int. J. Non-linear Mechanics, Vol. 25, No. 2/3, pp. 127-141, 1990.

[Gao00] S. Gao, H. Wan and Q. Peng, *An approach to solid modeling in a semi-immersive virtual environment*, Computers and Graphics, Vol. 24, pp. 191-202, 2000.

[Gar86] J. Garcia de Jalon, J. Unda and A. Avello, *Natural coordinates for the computer analysis of multibody systems*, Computer Methods in Applied Mechanics and Engineering 56, pp. 309-327, 1986.

[Gar87] J. Garcia de Jalon, J. Unda, A. Avello and J. M. Jimenez, *Dynamic analysis of three-dimensional mechanisms in 'natural' coordinates*, Trans. of the ASME, Journal of Mechanisms, Transmissions and Automation in Design, Vol. 109, pp. 460-465, Dec., 1987.

[Gar93] J. Garcia de Jalon and E. Bayo, *Kinematic and dynamic simulation of multibody systems*, Mechanical Engineering Series, Springer-Verlag, 1993.

[Gas94] J. D. Gascuel, M. P. Gascuel, *Displacement constraints for interactive modeling and animation of articulated structures*, The Visual Computer, Vol. 10, pp. 191-204, 1994.

[Gea85] C. W. Gear, B. Leimkuhler et G. K. Gupta, *Automatic integration of Euler-Lagrange equations with constraints*, J. Compt. Appl. Math., Vol. 12 et 13,pp. 77-90, 1985.

[Ger96] M. Géradin et D. Rixen, *Théorie des vibrations, Application à la dynamique des structures*, 2ème édition, Recherche en Mécanique, Ed. Masson, 1996.

[Gom99] A. Gomes de Sa, G. Zachmann, *Virtual reality as a tool for verification of assembly and maintenance processes*, Computers and Graphics, Vol. 23, pp. 389-403, 1999.

[Hah88] J. K. Hahn, *Realistic animation of rigid bodies*, Computer Graphics, Vol. 22, pp. 179-188. Aug., 1988

[Hen00] B. Hendrickson, T. G. Kolda, *Graph partitionning models for parallel computing*, Parallel computing, Vol. 26, pp. 1519-1534, Elsevier, 2000.

[Hil77] H. Hilber, T. Hughes, R. Taylor, *Improved numerical dissipation for time integration algorithm in structural dynamics*, Earthquake Engineering and Structural Dynamics, Vol. 5, pp. 283-292, 1997.

[Hof98] H. Hoffman, *Physically Touching Virtual Objects Using Tactile Augmentation Enhances the Realism of Virtual Environments,* Proceedings of Virtual Reality Annual International Symposium, pp. 59-63, 1998.

[Hol80] J.M. Hollerbach, *A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity,* IEEE Trans. Systems, Man and Cybernetics SMC-10 (11), pp. 730-736, Nov., 1980.

[Imb95] J.-F. Imbert, *Analyse des structures par éléments finis,* 3ème édition, CEPADUES-EDITIONS, Mai, 1995.

[Isn97] F. Isnard, *Génération des équations du mouvement de systèmes polyarticulés avec prise en compte de rigidités par des multiplicateurs de Lagrange,* Thèse de Doctorat, Université de Poitiers, 1997.

[Jay97] S. Jayaram, H. I. Connacher, K. W. Lyons, *Virtual assembly using virtual reality techniques,* Computer-Aided Design, Vol. 29, No. 8, pp. 575-584, 1997.

[Jim01] P. Jiménez, F. Thomas and C. Torras, *3D collision detection : a survey,* Computers and Graphics, Vol. 25, pp. 269-285, Elsevier, 2001.

[Jol93] P. Joli, *Modélisation et simulation de systèmes à topologie variable dûe à des liaisons de contact* ,Thèse de Doctorat, Université Paris 6, Nov., 1993.

[Keil99] A. Keil, H. Hermsdorf and V. Enderlein, *On the description of multibody systems models,* Advances in Computational Multibody Dynamics, IDMEC/IST, Lisbon, Portugal, Sept., 1999.

[Ket01] S. Kettebov and R. Sharma, *Toward natural gesture/speech control of a large display*, in Engineering for Human-Computer Interaction, Lectures Notes in Computer Science, Springer-Verlag, 2001.

[Kha99] W. Khalil et E. Dombre, *Modélisation, identification et commande des robots,* 2ème édition revue et augmentée, Hermès, 1999.

[Kim99] S.-S Kim and Y.-S. Oh, *A real-time vehicle dynamics model using a subsystem synthesis method,* Proc. ASME Design Engineering Technical Conferences, DETC2001/VIB-21311, Sept. 9-12, Pittsburgh, Pensylvania, 1999.

[Küb99] R. Kübler and W. Schielen, *Modular modeling and simulation of multibody systems,* Proc. ASME, Design Engineering Technical Conferences, Las Vegas, Nevada, Sept., 1999.

[Küb00] R. Kübler and W. Schielen, *Modular simulation in multibody system dynamics,* Multibody System Dynamics, Vol. 4, pp. 107-127, Aug. 2000.

[Kun98] D. L. Kunz, *An object-oriented approach to multibody systems analysis,* Computers & Structures, Vol. 69, pp. 209-217, 1998.

[Lee88] C. S. G. Lee and P. R. Chang, *Efficient parallel algorithms for robot forward dynamics computation,* IEEE Trans. Syst. Man Cybern., Vol. 18, No. 2, pp. 238-251, March/April, 1988.

[Mcm92] S. McMillan, D.E. Orin and P. Sadayappan, *Toward super real-time simulation of robotic mechanisms using a parallel integration method*, IEEE Trans. Syst. Man Cybern., Vol. 22, No. 2, pp. 384-391, March/April, 1992.

[Mir96] B. V. Mirtich, *Impulse-based dynamic simulation of rigid body systems*, PhD thesis, University of California, Berkeley, 1996.

[Nik94] P. E. Nikravesh and H.A. Affifi, *Construction of the equations of motion for multibody dynamics using point and joint coordinates,* Computer-Aided Analysis of Rigid and Flexible Mechanical Systems, pp. 31-60, Kluwer Academic Publishers, 1994.

[Noc01] O. Nocent, *Animation dynamique de corps déformables continus : application à la simulation de textiles tricotés*, Thèse de Doctorat, Université de Reims Champagne – Ardenne, Déc., 2001.

[Par97] J. Park and D. S. Fussell, *Forward dynamics based realistic animation of rigid bodies,* Computer and Graphics, Vol. 21, No. 4, pp. 483-496, 1997.

[Red02] S. Redon, *Algorithmes de simulation dynamique interactive d'objets rigides*, Thèse de Doctorat, Université d'Evry, Oct., 2002.

[Rég97] S. Régnier, F. B. Ouezdou et P. Bidaud, *Résolution distribuée de la synthèse géométrique de systèmes robotiques,* Journal Européen des Systèmes Automatisés, Vol. 31, No. 1, 1997.

[Sch97] W. Schielen, *Multibody system dynamics : roots and perpectives,* Multibody System Dynamics, Vol. 1, pp. 149-188, Jun., 1997.

[Seg03] N. Séguy, P. Joli, Z.-Q. Feng, M. Pascal, *A modular dynamic mode lfor multibody system adapted to interactive simulation,* Proc. ASME, Design Engineering Technical Conferences DETC03/VIB48311, Chicago, Illinois, Sept, 2003.

[Sha01] A. A. Shabana, *Computational Dynamics,* Second Edition, John Wiley and Sons, 2001.

[Sin85] R. P. Singh and P. W. Likins, *Singular value decomposition for constrained dynamical systems,* Trans. of the ASME, Journal of Applied Mechanics, Vol. 52, pp. 943-948, Dec. 1985.

[Spo89] M. W. Spong and M. Vidyasagar, *Robot dynamics and control*, John Wiley and Sons, 1989.

[Stur94] D. J. Sturman and D. Zeltzer, "*A survey of glovebased input*," IEEE Computer Graphics and Applications, Vol. 14, pp. 30-39, Jan. 1994.

[Tse01] F.-C. Tseng, G. M. Hulbert, *A gluing algorithm for network-distributed dynamics simulation,* Multibody System Dynamics, Vol. 6, pp. 377-396, 2003.

[Und87] J. Unda, J. Garcia de Jalon, F. Losantos, R. Enparantza, *A comparative study on some different formulations of the dynamic equations of constrained mechanical systems,* Trans. of the ASME, Vol. 109, pp. 466-474, Dec., 1987.

[Val99] B. Valton, *Gestion de la complexité des scènes animées et interactives : contributions à la conception et à la représentation,* Thèse de Doctorat, Université de Rennes 1, Dec., 1999.

[Wal82] M. W. Walker and D. E. Orin, *Efficient dynamic computer simulation of robotic mechanism,* IEEE Trans. Syst., Man, Cybern., pp. 384-391, Mar., 1982.

[Wan94] P.S. Wang, *C++ with object – oriented programming,* PXS Publishing company, ISBN 0-534-19644-6, 1994.

[Wan02] J. Wang, C. M. Gosselin, L. Cheng, *Modeling and simulation of robotic systems with closed kinematics chains using the virtual spring approach,* Multibody System Dynamics, Vol. 7, pp. 145-170, 2002.

[Wan03] J. Wang, Z.-D. Ma, G. M. Hulbert, *A gluing algorithm for distributed simulation of multibody systems*, Proc. ASME, Design Engineering Technical Conferences DETC03/VIB48313, Chicago, Illinois, Sept, 2003.

[Weh82] R. A. Wehage and E. J. Haug, *Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems,* Trans. of the ASME, Journal of Mechanical Design, Vol. 104, pp. 247-255, Jan., 1982.

[Wit90] A. Witkin, M. Gleicher and W. Welch, *Interactive dynamics,* Computer Graphics, Vol. 24, No. 2, pp. 11-21, Mar. 1990.

[Wri96] R. Wright, M. Sweet Jr., *OpenGL superbible: the complete guide to OpenGL programming for Windows NT and Windows 95,* Waite Group Press, 1996.

Sites Web

[Www1] http://www.adams.com

[Www2] http://www.3ds.com/en/home.asp

[Www3] http://www.solidworks.com

[Www4] http://www.ideas-simulation.com

[Www5] http://gmfe16.cemif.univ-evry.fr:8080/~feng

[Www6] http://www.lms.be

MODELISATION MODULAIRE DE SYSTEMES ARTICULES: CONCEPTION ORIENTEE-OBJET DE PLATES-FORMES DE SIMULATION

En vue de la réalisation d'une plate-forme de simulation interactive, ce travail de thèse propose une modélisation modulaire des équations du mouvement de systèmes multicorps à partir de méthodes de pénalité. Un système dynamique est alors décrit comme un assemblage de sous-systèmes contraints possédant leurs propres modèles et méthodes de résolution. La stabilité numérique est assurée par un calcul automatique des facteurs de pénalité introduits au niveau des liaisons et un processus itératif basé sur un formalisme de Lagrangien augmenté au niveau des composants.

Les formalismes et les méthodes de résolution retenues respectent les propriétés de flexibilité afin d'assurer une modification aisée de la topologie du système. Cet assemblage de composants autonomes est adapté à une programmation orientée-objet qui permet l'ajout ou la suppression dynamiques d'instances des classes de bases. L'outil de simulation autorise alors des modifications interactives de la scène virtuelle.

Mots-clés: systèmes multicorps, dynamique, modélisation modulaire, méthodes de pénalité, lagrangien augmenté, critère de stabilité, interactivité

MODULAR MODELLING OF MULTIBODY SYSTEMS: OBJECT-ORIENTED CONCEPTION OF CAD-TOOLS

This work presents a modular model of rigid multibody systems using penalty methods in the context of interactive simulation in computer-aided design. A dynamic system can be viewed as an assembly of constrained sub-systems with their own formalism and solving methods. Automatic calculation of penalty factors at each joint and the use of an iterative process based on an augmented Lagrangian formulation in components insure numerical stability.

Formalisms and solving methods are chosen for flexibility in order to easily modify the topology of the system. This assembly of autonomous components is well suited to object-oriented programming in which instances of the basic classes can dynamically be added or removed. The simulation tool can then interactively modify the virtual scene.

Keywords: multibody systems, dynamics, modular modelling, penalty methods, augmented Lagrangian formulation, numerical stability, interactivity.