THÈSE

délivrée par **L'USTL**

par

Guillaume Saupin

pour obtenir le diplôme de

DOCTORAT DE L'UNIVERSITÉ des Sciences et Technologies de Lille

spécialité Informatique

Vers la simulation interactive réaliste de corps déformables virtuels

Soutenance prévue le

JURY

.

M.	Y.REMION	Professeur à l'Université de Reims Champagne-Ardenne	Rapporteur
D^r	M.TESCHNER	Professor at the University of Freiburg	Rapporteur
M.	C.DURIEZ	Co-Encadrant de thèse Chargé de recherche à l'INRIA	Co-Encadrant
M.	A. MICAELLI	Co-encadrant de thèse Directeur de recherche au Commissariat à l'Energie Atomique	Co-Encadrant
M.	L.GRISONI	Directeur de thèse Professeur (HDR) à l'Université Des Sciences Et Technologies de Lille 1	Examinateur

Table des matières

Table des figures

Liste des tableaux

Mon lot était la cruelle inquiétude du chercheur.

Marcel Proust

Remerciements

Je tiens à remercier en premier lieu ...

Chapitre 1

Introduction

Les environnements virtuels, complètement absents de nos vies il n'y a même pas quelques années, s'en sont largement emparés récemment. Dans le domaine des jeux vidéos, il existe désormais des univers très riches, à l'instar de Second Life, qui se substituent à la réalité. Dans l'industrie, ces environnements sont encore plus présents, et depuis plus longtemps. Il n'est plus imaginable de s'en passer. Enfin, dans le domaine de la santé, l'imagerie médicale fait beaucoup appel aux techniques de visualisation 3D, mais pour l'instant principalement dans un but de diagnostique. Il existe encore un déficit de confiance de la part des praticiens pour utiliser ces méthodes afin d'assister leur opérations.

Cette réticence est assez compréhensible dans la mesure où jusqu'à présent, les environnements virtuels proposés se limitaient à la représentation graphique du réel. Le comportement des objets dans leur environnement virtuel ne suivait en rien des lois physiques. D'où la méfiance justifiée du corps médical et l'impatience du monde industriel pour des méthodes plus physiques.

A présent, la puissance de calcul toujours croissante des ordinateurs, couplée à la vigueur de la communauté scientifique, ont rendu possible la création d'univers virtuels plus proches de la réalité. Sont ainsi apparus des simulateurs interactifs des solides déformables. Ces derniers ont permis de multiples applications.

Cependant, ces simulateurs, lorsqu'ils sont utilisés dans un contexte interatif, sont sujets à des limitations qui restreignent leur utilisation. Par exemple, les objets simulés ne peuvent entrer en contact avec d'autres objets, ou il le font mais selon des lois non physiques, ou encore ils ne peuvent se déformer que suivant de petites amplitudes ou en le faisant de manière non réaliste, ou suivant des modes de déformations limités.

Nous avons donc essayé dans cette thèse de trouver des réponses à ces limitations, avec pour objectif de simuler la déformation en grand déplacement d'objets, et ce de manière aussi physique que possible. Nous avons aussi voulu autoriser la collision de nos modèles déformables avec d'autres objets, avec toujours le soucis d'un réalisme physique. Enfin, pour améliorer la perception de notre environnement virtuel pour l'utilisateur nous avons essayé de mettre en place un retour haptique.

Les méthodes que nous avons proposées pour atteindre ces objectifs sont présentées ici, en commençant par une revue des méthodes rencontrées régulièrement dans la communauté Computer Graphics pour déformer des objets. Cette revue se tient dans le chapitre 2 et présente ces méthodes suivant une complexité croissante en terme de modèle physique mais aussi d'implémentation. Cette première partie nous conduira à choisir le modèle corotationnel élémentaire comme base physique pour nos simulations.

Vient ensuite le chapitre 3 sur la résolution des équations induites par les modèles simulés. Nous y présenterons les schémas d'intégration numérique de type implicite et explicite qui permettent de faire évoluer nos modèles dans le temps. Leurs qualités et défauts seront mis en exergue. Nous verrons aussi quels solveurs peuvent être utilisés, et avec quelles performances pour

résoudre les équations découlant de l'étape d'intégration. Au terme de cette présentation nous verrons que les solveurs multigrilles semblent particulièrement intéressants bien qu'ils souffrent de limitations auxquelles nous essaierons d'apporter une solution. La nature hiérarchique de ces solveurs nous amènera aussi à voir comment utiliser ces derniers dans un contexte ondelette.

Après ce chapitre, au terme duquel nous aurons les moyens de déformer un objet soumis à un effort, nous nous intéresserons à la gestion des contacts entre cet objet et son environnement. Nous présenterons alors dans ce chapitre 4 les méthodes développées jusqu'à présent, non sans avoir présenté auparavant les lois de SIGNORINI et COULOMB qui forment un bon modèle du contact et du frottement. Nous montrerons que les méthodes basées sur des problèmes de complémentarité apportent un réalisme physique intéressant. Nous verrons aussi que ce réalisme a un coût induit par la nécessité de connaître la matrice de compliance. Nous proposerons alors notre méthode de COMPLIANCE WARPING qui permet de gérer plus rapidement les contacts avec néanmoins le même respect des lois de contact de SIGNORINI et COULOMB. Cette accélération repose sur l'utilisation d'un modèle de contact simplifié basé sur un modèle corotationnel nodal.

Enfin, dans un dernier chapitre 5, nous présenterons le retour haptique à travers les sens qu'il stimule chez l'homme, et les interfaces existantes pour exciter ces sens. Nous verrons aussi que l'inclusion de l'homme dans la boucle de simulation entraîne des contraintes en fréquence de simulation et en stabilité. La littérature fournit des solutions pour les problèmes de stabilité, notamment avec le concept du GOD OBJECT. Néanmoins, dans notre cas, il faut ajouter à ces difficultés la lenteur des simulations de solides déformables, qui est incompatible avec la fréquence de 1Khz du rendu haptique. Nous tenterons donc de contourner cette difficulté en proposant de découpler boucle physique et boucle haptique.

Chapitre 2

Simulation interactive de corps déformables

Sommaire

2.1 Introduction

Cette thèse a pour objectif la simulation de solides déformables avec une contrainte d'interactivité. Elle s'appuie donc sur des travaux issues de la communauté Computer Graphics.

Dans cette communauté, la simulation de corps déformables à fait l'objet de nombreux travaux depuis le début des années 80, motivés principalement par un soucis de rapidité et par la puissance de calcul limitée des ordinateurs. Cette double contrainte restreignait à des modèles relativement simples l'utilisation de la Mécanique des Milieux Continus, MMC, qui décrit le comportement macroscopique de matériaux soumis à des efforts, et ce en raison de son coût calculatoire trop important.

Cette nécessité de s'affranchir de la Mécanique des Milieux Continus pour pouvoir simuler des objets complexes, a donc conduit à l'élaboration de nombreuses méthodes de simulation qui se caractérisent par un temps de calcul réduit et des propriétés de déformation au

moins *visuellement* réalistes. Tout cela se fait au prix d'un réalisme physique moins important que celui apporté par la MMC, qui est elle même une modélisation simplifiée de la réalité. Nous présentons ces méthodes dans la section 2.2.

Cependant, du fait du doublement tout les dix-huit mois[1] de la puissance de calcul des ordinateurs, l'utilisation de méthodes basées sur la MMC est devenue envisageable sur des modèles de plus en plus complexes. Nous rappelons donc dans la section 2.3 les bases de la MMC dans le cadre de la théorie de l'élasticité, puis nous présentons en 2.4 les solutions classiquement mises en œuvre pour résoudre ces équations.

Enfin, dans une dernière section 2.5, nous montrons comment certains auteurs ont essayé de combiner les avantages des méthodes *visuelles*, c'est à dire leur rapidité, avec le respect des lois physiques inhérent à celle basées sur la MMC.

La présentation de l'ensemble de ces méthodes nous permettra de dégager la méthode répondant le mieux à nos attentes, c'est à dire permettre la modélisation interactive et physique d'objets de géométries quelconques et potentiellement constitués de matériaux hétérogènes.

2.2 Les méthodes visuellement correctes

En informatique, et en particulier dans le domaine de la simulation, la puissance de calcul est souvent un facteur limitant. Ainsi, la simulation de corps déformables ne déroge pas à la règle, et de nombreuses méthodes ont été développées pour obtenir des simulateurs rapides, et ce au prix d'une certaine distance par rapport à la réalité physique.

Cependant, dans de nombreuses applications, comme les jeux vidéo, l'animation, ou encore certaines applications médicales, il n'est pas nécessaire de fournir une déformation qui soit rigoureusement physique. Souvent un rendu graphique qui ne choque pas l'utilisateur est suffisant.

Dans ce cadre là, l'utilisation des méthodes que nous présentons ci-dessous, qui fournissent une déformation visuellement plausible, est parfaitement légitime.

2.2.1 Les modèles Shape Matching

Les modèles Shape Matching sont un fruit des travaux de Müller [MHTG05] qui exploite les Moindres Carrés, sans se baser sur une loi de comportement physique.

Le principe consiste à essayer de faire correspondre des points de référence $\mathbf{x}_i^0$ avec des points déformés $\mathbf{x}_i$, de manière à trouver un ensemble de points $\mathbf{g}_i$ minimisant la différence entre ces deux ensembles de points. Les point $\mathbf{x}_i$ sont simplement des masses m_i, sans lien entre elles, et qui se déplace librement sous l'effet des forces qui leur sont appliquées. Les déformées ainsi obtenues sont généralement peu vraisemblables.

Cette méthode implique les étapes suivantes :

1. Dans un premier temps, après que les particules $\mathbf{x}_i$ se soient déplacées librement, on cherche à trouver une rotation et une translation qui permettent de minimiser la fonction suivante :

$$\sum_i w_i \left(\mathbf{R}(\mathbf{x}_i^0 - \mathbf{t}_0) + \mathbf{t} - \boldsymbol{x}_i\right)^2 \tag{2.1}$$

[1] Selon la loi de Moore.

2. Pour cela, il s'avère que la meilleure translation est simplement et naturellement le centre de masse de l'objet, *i.e.* $\mathbf{t}_0 = \frac{\sum_i m_i \boldsymbol{x}_i^0}{\sum_i m_i}$ et $\mathbf{t} = \frac{\sum_i m_i \boldsymbol{x}_i}{\sum_i m_i}$
3. Ensuite plutôt que de chercher directement la rotation, on tente de minimiser l'équation 2.1 en utilisant un opérateur linéaire $\mathbf{A}$ à la place :

$$\sum_i w_i \left(\mathbf{A}(\mathbf{x}_i^0 - \mathbf{t}_0) + \mathbf{t} - \boldsymbol{x}_i\right)^2$$

dont l'expression, après dérivation de l'équation, s'avère être :

$$\mathbf{A} = \left(\sum_i m_i \mathbf{p}_i \mathbf{q}_i^T\right)\left(\sum_i m_i \mathbf{q}_i \mathbf{q}_i^T\right)^{-1} = \mathbf{A}_{pq}\mathbf{A}_{qq}$$

où $\mathbf{q}_i$ et $\mathbf{p}_i$ sont respectivement les points de référence $\mathbf{x}_i^0$ exprimés dans le repère centré sur le centre de masse $\mathbf{x}_{cm}^0$, et les points déformés $\mathbf{x}_i$ exprimés dans le repère centré sur le centre de masse déformé $\mathbf{x}_{cm}$. La rotation $\boldsymbol{R}$ est alors extraire de la matrice $\mathbf{A}_{pq}$, la matrice $\mathbf{A}_{qq}$ étant symétrique et ne contenant par conséquent pas de composante rotation. $\mathbf{R}$ est alors extraite par décomposition polaire ou othonormalisation de la matrice $\mathbf{A}_{pq}$.
4. Enfin, sur la base de cette rotation, les points $\mathbf{g}_i$ sont calculés :

$$\mathbf{g}_i = \mathbf{R}(\mathbf{x}_i^0 - \mathbf{t}^0) + \mathbf{t} \tag{2.2}$$

Une fois les points $\mathbf{g}_i$ déterminés, ils sont utilisés pour intégrer le déplacement des points déformables suivant le schéma :

$$\begin{array}{rcl} \mathbf{v}_i(t+h) & = & \mathbf{v}_i(t) + \alpha\frac{\mathbf{g}_i(t)-\mathbf{x}_i(t)}{h} + h\frac{f_{ext}(t)}{m_i} \\ \mathbf{x}_i(t+h) & = & \mathbf{x}_i(t) + h\mathbf{v}_i(t+h) \end{array} \tag{2.3}$$

Le paramètre $\alpha \in [0,1]$ permet de moduler la raideur.

Grands Déplacements La méthode présentée restreint la déformation à de petits déplacements. Ainsi, si les particules $\mathbf{x}_i$ se déplacent de manière trop conséquente, la déformée calculée par cette méthode devient aberrante. Pour accroître l'amplitude de la déformée, [MHTG05] propose d'utiliser la transformation $\beta\mathbf{A} + (1-\beta)\mathbf{R}$ à la place de la rotation $\mathbf{R}$ dans l'équation 2.2. L'utilisation de $\boldsymbol{A}$ permet d'autoriser une déformation linéaire, tandis que $\boldsymbol{R}$ garantie la ressemblance avec l'objet de référence.

Le type de déformée peut encore être étendue en autorisant une déformation quadratique. Pour cela les points $\boldsymbol{g}_i$ sont définis par une fonction quadratique :

$$\boldsymbol{g}_i = [\boldsymbol{AQM}]\,\tilde{\boldsymbol{q}}_i \tag{2.4}$$

où $\tilde{\boldsymbol{q}}_i = \left[q_x, q_y, q_z, q_x^2, q_y^2, q_z^2, q_xq_y, q_yq_z, q_zq_x\right]^T$. Si l'on pose $\tilde{\boldsymbol{A}} = [\boldsymbol{AQM}]$, alors la fonction à minimiser est $\sum_i m_i \left(\tilde{\boldsymbol{A}}\tilde{q}_i - \boldsymbol{p}_i\right)^2$ qui est minimale pour :

$$\tilde{\boldsymbol{A}} = \left(\sum_i m_i \mathbf{p}_i \tilde{\mathbf{q}}_i^T\right)\left(\sum_i m_i \tilde{\mathbf{q}}_i \tilde{\mathbf{q}}_i^T\right)^{-1} = \tilde{\mathbf{A}}_{pq}\tilde{\mathbf{A}}_{qq} \tag{2.5}$$

Les différents types de déformation obtenus sont représentés dans la figure 2.1.

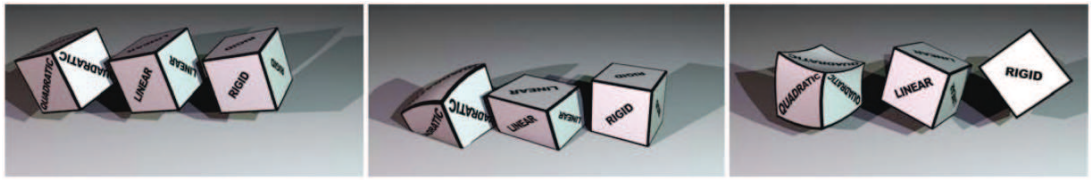


FIG. 2.1 – Les déformations dans les cas rigide, linéaire et quadratique. Issu de [MHTG05]

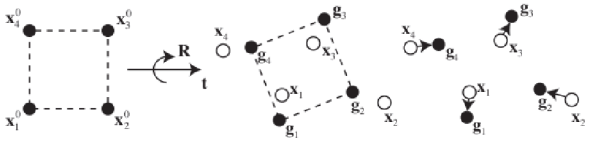


FIG. 2.2 – Le principe. Issu de [MHTG05]

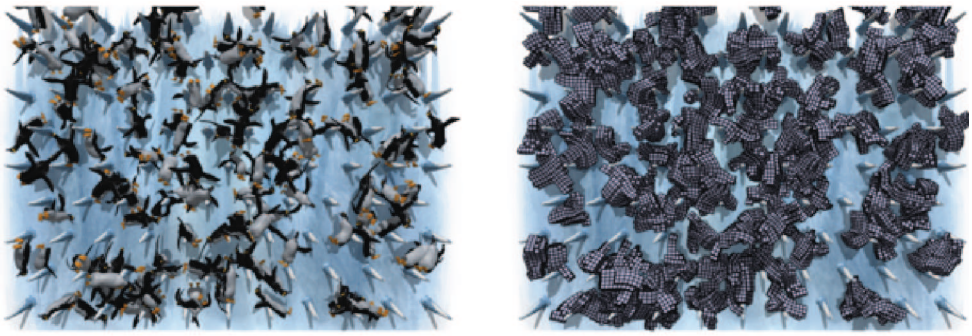


FIG. 2.3 – Le principe. Issu de [RJ07]

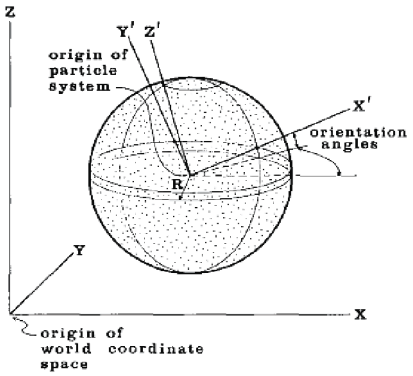


FIG. 2.4 – Un système de particules sphérique. Issue de [Ree83].

Remarques Ce type de modèle est rendu particulièrement stable par le schéma d'intégration utilisé et permet du fait de sa simplicité des simulations interactives même pour des modèles de grandes dimensions. Il souffre néanmoins d'un réalisme peu physique comme en attestent les images de la 2.1. Cela dit, ce réalisme peu être amélioré en gardant le même principe de minimisation d'une fonction, mais en basant cette dernière sur des considérations physiques. C'est ce que font certaines des méthodes présentées ci-dessous.

Variations du modèle RIVERS propose dans[RJ07] d'améliorer encore ce modèle, de manière à tendre vers une déformée plus plausible, en déformant en fait un treillis de cubes. Voir figure 2.3. Ce treillis de cubes est construit de telle manière qu'il enveloppe le maillage surfacique de l'objet à déformer. Chacun des cubes du treillis est déformé en minimisant des déplacements par rapport à ce même cube, mais dans la position initiale. La déformation de la surface est ensuite calculée en utilisant une interpolation trilinéaire. L'apport de l'article réside surtout dans la mise au point d'algorithmes permettant de réduire la complexité des calculs de $O(w^3)$, la méthode *force brute*, en $O(1)$, la méthode FASTSLM, où w_3 est le nombre de cubes dans le treillis.

STEINEMANN améliore encore l'efficacité et le réalisme dans [SOG08] en présentant les algorithmes utilisables pour ajouter une caractère adaptatif à cette méthode, ce qui permet de simuler encore plus finement la déformation, et ce en un temps toujours restreint.

BOTSCH étend aussi son modèle [BPGK06] basé sur les SHAPE MATCHING pour les surfaces aux volumes dans [BPWG07]. Comme RIVERS et STEINEMANN, ses objets déformables sont modélisés par des cubes rigides. Là où il se distingue, par contre, c'est qu'il se rapproche un peu plus d'un modèle physique puisque la déformation des cubes rigides est gouvernée par la minimisation d'une énergie potentielle élastique. La géométrie de l'objet reste par contre approximée, et l'énergie potentielle élastique porte sur la position des petits cubes, ce qui reste peu physique.

SHI *et al.* [SYBF06] utilisent aussi une version modifiée de ces méthodes avec en plus une approche multigrille qui lui permet d'accroître les performances.

2.2.2 Les systèmes de particules

Les systèmes de particules sont des modèles utilisés pour modéliser des objets déformables diffus, sans contours, informes, tels que des nuages de fumée, de la pluie [WW04], de la neige

[WW04], des cascades [HW04], des matériaux granulaires [BYM05], du feu [Ree83], des fluides visco-élastiques [CBP05], *et cætera*.

Leur fonctionnement est basé sur l'utilisation de particules indépendantes qui peuvent être dotées de diverses propriétés : vitesse, accélération, masse, couleur, température, durée de vie, forme.

Ces particules sont créées par un système de particules qui possède lui aussi des propriétés, notamment une origine, un volume de contrôle qui définie l'espace mais aussi les directions dans lesquels les particules évoluent. Il définit aussi où, quand et dans quelle quantité les particules sont crées. Il se charge de définir la valeur initiale de chacune des particules créées. Initialement, REEVES proposait dans [Ree83] de calculer ces valeurs de manière stochastique en spécifiant une gaussienne pour chacune des propriétés :

$$p = \bar{p} + rand\sigma_p \tag{2.6}$$

Une fois créées, ces particules voient leur propriétés changer dynamiquement dans le temps. Le mouvement est généralement décrit par les équations classiques de la mécaniques du point, tandis que pour les autres propriétés, toute sorte de loi est envisageable.

Enfin, une fois la durée de vie de la particule écoulée, cette dernière est détruite. L'algorithme 1 résume ces étapes.

Algorithm 1 Évolution d'un système de particules

```
1: while 1 do
2:     création de particules
3:     calcul des propriétés
4:     calcul des forces
5:     mise à jour des vitesses et positions
6: end while
```

Interaction entre particules Ce type de système de particules reste cependant limité à quelques types particuliers de simulation, et ce en raison de l'absence d'interaction entre particules. C'est pourquoi un autre type de système de particules, proche des modèles MASSE-RESSORT a été développé : les systèmes de particules spatialement couplés, si l'on suit l'appellation consacrée que TONNESEN donne dans [Ton92]. Ces systèmes de particules s'inspirent des modèles moléculaires. L'idée est d'associer à ces particules une fonction support W_h, appelée aussi *kernel* qui définit en quelque sorte l'influence spatiale de la particule. h donne la dimension du support.

Idéalement c'est la fonction de LENNARD-JONES :

$$W_h = V_{LJ}(r) = 4\pi \left[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6} \right] \tag{2.7}$$

définissant l'énergie potentielle entre deux particules distantes de r qui sert à définir la force $\mathbf{f}_{ij}$ liant les deux particules i et j :

$$\mathbf{f_{ij}} = \bigtriangledown V_{LJ} \tag{2.8}$$

L'allure de V_{LJ} est donnée dans la figure 2.5. Dans la pratique, d'autres fonctions peuvent être utilisées. Des fonctions de type gaussiennes sont ainsi souvent rencontrées.

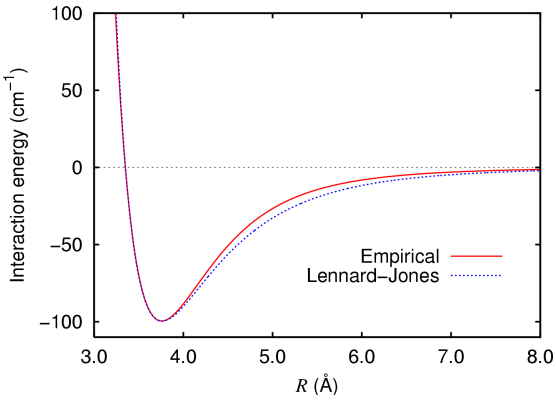


FIG. 2.5 – L'energie potentielle de LENNARD-JONES.

L'utilisation du *kernel* ne se cantonne pas uniquement aux calculs de forces inter-particules. Il peut être utilisé pour interpoler n'importe quelle grandeur physique A :

$$A(\boldsymbol{x}) = \sum_j \frac{m_j}{\rho_j} W_h(\boldsymbol{x}) \tag{2.9}$$

dont le gradient peut être calculée facilement pour peu que $\bigtriangledown W_h$ le soit :

$$\bigtriangledown A(\boldsymbol{x}) = \sum_j \frac{m_j}{\rho_j} \bigtriangledown W_h(\boldsymbol{x}) \tag{2.10}$$

L'ajout de forces inter-particules, qu'elles suivent la loi de LENNARD-JONES ou autre, permet la simulation d'objets déformables plus nombreux, comme des tissus [CMT02], des fluides [HHK08, CBP05], des objets déformables [TPF91, DG96], ou même des matériaux granulaires de type sable [BYM05].

Le calcul du mouvement des particules se fait alors simplement en utilisant indépendemment pour chaque particules les lois de la mécanique du point.

Remarque Ce type de modèle se rapproche des modèles MASSE-RESSORT dans le sens où ce sont des masses ponctuelles qui interagissent entre elles. Cependant, les SYSTÈMES DE PARTICULES se distinguent par le fait qu'il n'y a pas de lien topologique entre les diverses particules. Leurs interactions sont simplement fonction de la distance qui les sépare.

2.2.3 Les modèles Masse-Ressort

Les modèles MASSE-RESSORT partent de la constatation suivante : dans la grande majorité des modèles physiques étudiés précédemment, les objets modélisés sont constitués d'un nuage de points qui interagissent au moins localement les uns avec les autres. Le principe des modèles MASSE-RESSORT est alors simplement de rendre compte de ces couplages locaux en liant des points voisins, dotés d'une masse, à l'aide de ressorts et d'amortisseurs pour permettre la dissipation d'énergie. Voir la figure 2.8

FIG. 2.6 – Simulation de sable. Issu de [BYM05]

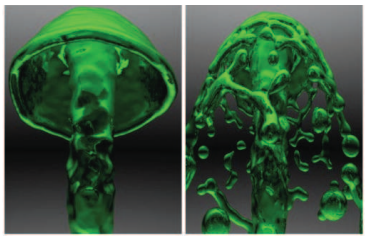


FIG. 2.7 – Simulation de fluide visco-élastique. Issu de [CBP05]

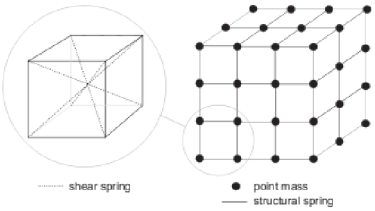


FIG. 2.8 – Le Masse Ressort. Issue de [NAM+06].

L'équation de la dynamique :

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{v}) \tag{2.11}$$

est donc particulièrement simple a résoudre, dans la mesure où $\mathbf{M}$ est diagonale, et où le calcul des forces internes l'est tout autant puisqu'il ne fait intervenir que la raideur k_{ij}, la longueur au repos l_{ij}, la position de nœuds voisins $\mathbf{x}_i$ et $\mathbf{x}_j$, et éventuellement la vitesses $\mathbf{v}_i$ et $\mathbf{v}_j$ de ces nœuds s'il y a un amortissement k_d :

$$\mathbf{f}_i = k_{ij}(|\mathbf{x}_{ij}| - l_{ij})\frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|} + k_d(\mathbf{v}_i - \mathbf{v}_j), \text{ où } \mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j \tag{2.12}$$

Ce principe peut être décliné à l'infini pour offrir des simulations diverses et originales : dans [TPF91] TERZOPOULOS fait varier la raideur de façon inversement proportionnelle à la température, de telle sorte qu'à un certain point, l'objet fond complètement ; dans [BHW94] un modèle similaire est utilisé pour simuler le drapé d'un tissu ; dans [Mil88] MILLER modélise la locomotion de serpents et autres vers en jouant sur la longueur au repos pour simuler la contraction des muscles ; TU propose un modèle semblable pour animer des poissons [TT94].

Remarques Les modèles masse-ressort souffrent d'une mauvaise convergence lorsque l'on raffine le maillage utilisé. Ainsi, un maillage affiné ne convergera pas forcément vers la même solution qu'un maillage grossier. Cela complique notablement le développement de modèle masse-ressort adaptatif. Et même s'ils permettent la simulation d'une large gamme de phénomènes physiques, ils sont rarement valables physiquement. Enfin, la déformée observée dépend du maillage.

Plus de physique La simplicité de ce modèle, son efficacité, sa facilité d'implémentation, et sa large gamme d'application l'ont rendu particulièrement intéressant, et sont à l'origine de travaux

visant à pallier à la principale limitation de ce modèle : son absence de justification physique.

Dans un premier temps, la notion d'énergie interne $U(\mathbf{x})$ a été introduite [BHW94, BW98]. Les forces internes sont alors définies comme le gradient de cette énergie :

$$\mathbf{f}_{int}(\mathbf{x}) = \frac{\partial U(\mathbf{x})}{\partial \mathbf{x}} \tag{2.13}$$

En s'abstrayant ainsi des ressorts, le modèle s'approche un peu plus de la réalité physique.

Dans [THMG04], TESCHNER précise les énergies linéiques, surfaciques et volumiques qui tendent à conserver respectivement les distances, les surfaces et les volumes. Cependant, ces énergies ne sont définies que sur des critères géométriques et ne rendent pas compte du comportement physique du matériau.

Plusieurs tentatives ont donc été faites pour dériver un modèle MASSE-RESSORT d'une comportement physique de matériau. Ainsi ETZMUSS montre [EGS03] qu'un système masse-ressort bien paramètré peut approximer efficacement un modèle élément finis dans le cas de petits déplacements. De même SERVIN propose dans [MLM06] d'utiliser une fonction d'énergie interne dérivée de la théorie de l'élasticité.

Il est aussi envisageable d'utiliser des méthodes d'apprentissage pour estimer les paramètres physiques de ces modèles, comme l'a montré [BTH+03].

2.3 La mécanique des milieux continus

Nous avons vu comment simuler la déformation d'objets de manière relativement plausible à l'aide de modèles simples et originellement non physiques. Cette approche, bien que notablement efficace en temps de calcul et en simplicité d'implémentation, souffre d'un piètre réalisme physiques. Pour assurer une plus grande fidélité, il faut se tourner vers les lois physiques régissant le comportement de matériaux soumis à des efforts.

La théorie qui décrit ces comportements est la mécanique des milieux continus. Elle fait l'objet d'une rapide présentation ci-dessous.

2.3.1 La théorie de l'élasticité

Parmis les modèles issus de la mécanique des milieux continus, MMC, on trouve l'étude des déformations des solides. Pour atteindre cet objectif, elle se propose d'étudier les efforts subis par une infime particule de matière. Elle définie pour cela un batterie de grandeurs tensorielles qui permettent de caractériser l'état mécanique local du matériau.

A cet ensemble de grandeurs s'ajoute une loi de comportement qui régit les relations entre ces différentes grandeurs. La plupart des travaux en simulation interactive utilisent la théorie de l'élasticité, qui décrit le comportement de matériaux purement élastiques.

Le tenseur des déformations Afin de caractériser l'état de déformation d'un objet, on utilise le tenseur des déformations ϵ. Ce tenseur rend compte du déplacement subit par un point de l'objet lorsque cet objet est déformé et est construit comme suit :

Soit un point P situé en $\mathbf{r}$ lorsque l'objet est au repos, et en $\mathbf{r}'$ lorsque l'objet est déformé. Le déplacement du point P entre ces deux configurations est noté $\mathbf{u} = \mathbf{r} - \mathbf{r}'$.

Si l'on considère deux points infiniment voisins P et P' initialement distants de $d\mathbf{x}$, alors après déformation, ces deux point sont distants de $d\mathbf{x} + d\mathbf{u}$. Ainsi, la distance séparant P et P' était initialement $dl = \sqrt{dx_1^2 + dx_2^2 + dx_3^2}$, et après déformation $dl' = \sqrt{dx'^2_1 + dx'^2_2 + dx'^2_3}$.

En linéarisant le déplacement au voisinage de P, on peut écrire $d\mathbf{u} = \bigtriangledown\mathbf{u}d\mathbf{x}$, d'où :

$$dl'^2 = dl^2 + 2d\mathbf{x}^T \bigtriangledown \mathbf{u}d\mathbf{x} + (\bigtriangledown\mathbf{u}d\mathbf{x})^T \bigtriangledown \mathbf{u}d\mathbf{x} \tag{2.14}$$

$$dl'^2 = dl^2 + d\mathbf{x}^T \bigtriangledown \mathbf{u}d\mathbf{x} + d\mathbf{x}^T\bigtriangledown\mathbf{u}^T d\mathbf{x} + d\mathbf{x}^T\bigtriangledown\mathbf{u}^T \bigtriangledown \mathbf{u}d\mathbf{x} \tag{2.15}$$

$$dl'^2 = dl^2 + d\mathbf{x}^T(\bigtriangledown\mathbf{u} + \bigtriangledown\mathbf{u}^T + \bigtriangledown\mathbf{u}^T \bigtriangledown \mathbf{u})d\mathbf{x} \tag{2.16}$$

$$dl'^2 = dl^2 + 2d\mathbf{x}^T\epsilon_G d\mathbf{x} \tag{2.17}$$

avec $\epsilon_G = \frac{1}{2}\left[\bigtriangledown\mathbf{u} + \bigtriangledown\mathbf{u}^T + \bigtriangledown\mathbf{u}^T \bigtriangledown \mathbf{u}\right]$ le tenseur des déformation de GREEN. La déformation en un point du matériau est donc décrite par 9 coefficients, *i.e* pour chaque direction de l'espace le matériau se déforme suivant la direction donné par un vecteur de dimension 3. On peut réduire ce nombre de cœfficients à 6 en notant la symétrie de construction de ce tenseur.

Le tenseur des contraintes Maintenant que nous avons une grandeur, ϵ_G qui nous permet de décrire l'état de déformation interne du matériau, nous aimerions pouvoir relier cette grandeur aux forces internes que subit l'objet. Pour cela, comme nous sommes dans le cadre de la théorie de l'élasticité, nous considérons que la loi de Hooke s'applique. C'est à dire qu'à chaque déformation correspond une contrainte qui lui est proportionnelle. Une contrainte est défini comme étant une force par unité de surface perpendiculaire à un axe particuliers. Une contrainte suivant un axe est donc un vecteur de dimension 3. L'ensemble des contraintes en un point est donc décrit par 9 coefficients. Il faut donc 81 coefficients pour relier les contraintes au déformations. C'est le tenseur d'élasticité $\mathbf{C}$, qui est d'ordre 4, qui stocke ces données :

$$\sigma = \mathbf{C} : \epsilon_G \tag{2.18}$$

Où σ est le tenseur des contraintes. Compte tenu de leur symétrie les tenseurs σ et ϵ_G peuvent être en fait décrit avec chacun 6 coefficients. Le tenseur $\mathbf{C}$ est donc défini avec uniquement 36 coefficients. Dans la pratique, pour des matériaux isotropes, deux coefficients sont utilisés pour construire C : λ et μ, les coefficients de LAMÉ :

$$\mathbf{C}_{xxyy} = \lambda \tag{2.19}$$

$$\mathbf{C}_{xyxy} = 2\mu \tag{2.20}$$

$$\mathbf{C}_{xxxx} = 2\mu + \lambda \tag{2.21}$$

Ce qui donne, d'après RICHARD FEYNMAN dans [FLS05] :

$$\sigma = 2\mu\epsilon_G + \lambda \bigtriangledown \cdot\mathbf{u} \tag{2.22}$$

Une fois σ construit il est possible de calculer les forces s'appliquant sur point de matière en calculant pour un volume infiniment petit l'intégrale des contraintes sur sa surface. Cette

grandeur n'est autre que $\bigtriangledown \cdot \sigma$, la divergence de σ, soit le flux de σ à travers la surface du volume infinitésimal.

$$\bigtriangledown \cdot \sigma = (\lambda + \mu) \bigtriangledown (\bigtriangledown \cdot \mathbf{u}) + \mu \bigtriangledown^2 \mathbf{u} \tag{2.23}$$

Les équations du mouvement Nous avons désormais les informations nécessaires pour calculer les forces internes d'un objet élastique qui se déforme. Nous allons pouvoir appliquer le principe fondamental de la dynamique :

$$\mathbf{f}_{int} + \mathbf{f}_{ext} = \rho \ddot{\mathbf{u}} \tag{2.24}$$

avec $\mathbf{f}_{ext}$ les forces extérieures et $\mathbf{f}_{int} = \bigtriangledown \cdot \sigma$ les forces internes. A noter que cette équation n'intègre pas de terme dissipatif. Généralement la dissipation d'énergie due à la déformation est prise en compte en utilisant une loi visco-élastique qui introduit un terme d'amortissement $b\dot{\mathbf{u}}$. D'où l'équation souvent utilisée :

$$\bigtriangledown \cdot \sigma + b\dot{\mathbf{u}} + \mathbf{f}_{ext} = \rho \ddot{\mathbf{u}} \tag{2.25}$$

2.4 Les modèles basés sur la théorie de l'élasticité

Le cadre de la théorie de l'élasticité permet de calculer la déformée d'un objet quelconque soumis à des forces ou contraintes. Les lois que nous avons donnés jusqu'à présent s'expriment à un niveau infinitésimal. Il faut donc les intégrer dans l'espace pour obtenir la déformée globale. Il existe pour cela plusieurs méthodes, dont les plus usitées sont présentées ci-dessous.

2.4.1 Les modèles Point-Based ou Meshless Finite Element

MÜLLER présente dans [MKN+04] un modèle s'inspirant des SMOOTH PARTICLES HYDRODYNAMICS, SPH, dans l'esprit des systèmes de particules, et adapté à la déformation d'objets. L'avantage de sa méthode réside dans la représentation uniquement ponctuelle aussi bien du volume que de la surface de l'objet. Il n'y a aucune information topologique entre les différents points.

Initialisation Étant basé sur la théorie de l'élasticité, ce modèle requière le calcul de propriétés définies sur des volumes. Dans l'esprit des SPH, les volumes à déformer sont donc approximés par des points, nommés *phyxels*, auxquels on associe une masse m_i, un volume v_i et une densité ρ_i. La masse est fixée de manière définitive à l'initialisation, tandis que la densité et donc le volume varient. La masse est distribuée autour du point suivant la densité locale de point. Comme pour les méthodes par particules, on utilise un *kernel* :

$$W_{h_i}(r_i) = \begin{cases} \frac{315}{64\pi h^9}(h_i^2 - r_i^2)^3 & \text{si } r_i < h_i \\ 0 & \text{sinon} \end{cases} \tag{2.26}$$

où r est la distance au *phyxel* et h définit le support.

Pour chacune des particules, la distance moyenne à chacune des k particules les plus voisines est utilisée comme support.

Simulation Ensuite, le gradient du déplacement $\bigtriangledown\mathbf{u}$, intervenant dans le calcul des forces internes, est estimé. Pour cela, les déplacements $\mathbf{u}_j$ voisins d'un point sont utilisés conjointement à une régression de type *Moving Least Square*.

De ce gradient, on déduit l'énergie potentielle élastique dont on dérive les forces s'appliquant sur les points. L'intégration dans le temps peut alors avoir lieu, et est faite dans [MKN+04] avec des schémas explicite et implicite.

Il est possible d'ajouter un comportement plastique à ce modèle en conservant simplement pour chaque point un déformation plastique ϵ^p que l'on soustraie à la contrainte courante ϵ. Ainsi si cette contrainte plastique est non nulle, l'objet est soumis à des forces qui lui assure une déformation constante.

En terme de performances, ce type de modèle n'est pas particulièrement efficace. Ainsi, pour un objet comportant seulement 200 *phyxels*, la fréquence de simulation est de 27 fps avec un intégrateur explicite, et 22 fps pour un intégrateur implicite. A noter qu'originellement plus de 50% du temps était passé en rendering, qui est coûteux du fait de l'absence de définition explicite de surface. Avec la puissance de calcul et la souplesse de programmation des cartes graphiques actuelles, ce pourcentage est réduit.

Variations Dans [AOW+08] Adams *et al.* dérivent ce modèle en partant du principe inverse, c'est à dire que partant de contraintes en déplacement ils calculent les contraintes et déformations. Pour cela ils utilisent l'expression littérale du gradient du déplacement. A noter que cette méthode ne permet pas la simulation de la dynamique de la déformation. Elle autorise seulement à déformer un objet en lui imposant des contraintes en position et vitesse.

Pauly étend aussi ce modèle à la simulation de fracture dans [PKA+05] en tirant parti de l'absence d'information topologique.

2.4.2 Les modèles Volumes Finis

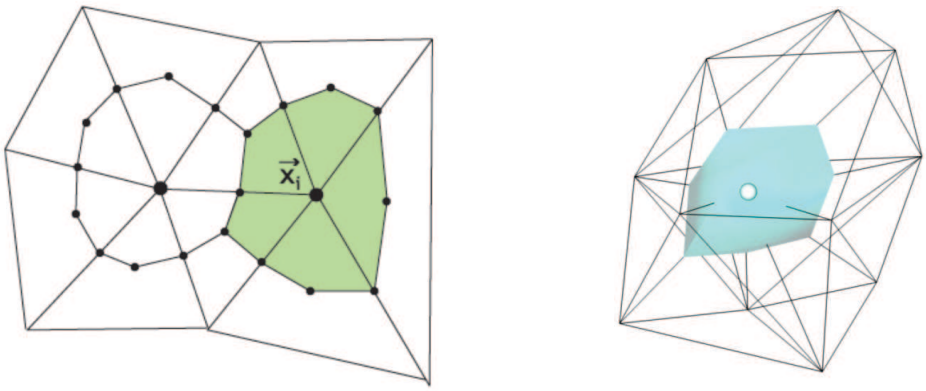
Cette méthode est basée sur un modèle proche des éléments finis, dans la mesure où l'objet à simuler est modélisé par un ensemble $\{T_k\}$ de petits volumes, les éléments, potentiellement non structuré. Cependant, contrairement à la méthode des élément finis, présentés en 2.4.5, ni le calcul variationnel, ni le principe des travaux virtuels ne sont utilisés. De plus, les calculs ne sont pas faits aux noeuds des éléments, mais à leur centre. Le maillage utilisé n'est donc pas à proprement parler le maillage tétraèdrique, mais plus exactement son dual, le diagramme de Voronï. C'est à dire qu'on s'intéresse au volume entourant le centre, comme l'illustre la figure 2.9.

Ainsi l'équation sous la forme forte 2.25 est simplement intégrée sur le domaine Ω :

$$\int_\Omega (\bigtriangledown \cdot \sigma + b\dot{\mathbf{u}} + \mathbf{f}_{ext} - \rho\ddot{\mathbf{u}})\, d\Omega = 0 \tag{2.27}$$

L'utilisation du théorème de la divergence nous permet de transformer l'intégrale sur un volume du terme $\bigtriangledown \cdot \sigma$ en une intégrale sur une surface :

$$\int_\Omega (b\dot{\mathbf{u}} + \mathbf{f}_{ext} - \rho\ddot{\mathbf{u}})\, d\Omega + \int_{\partial\Omega} \sigma\mathbf{n} d\partial\Omega = 0 \tag{2.28}$$



(a) (b)

FIG. 2.9 – Un volume fini en 2D (a), issue [TBHF03], et en 3D (b).

Comme on s'intéresse à la valeur du déplacement au centre de l'élément, le déplacement moyen u_k sur un volume T_k est une bonne approximation du déplacement au centre :

$$u_k = \frac{1}{V_k} \int_{T_k} u dT_k \tag{2.29}$$

où V_k est le volume de T_k. Il est alors possible d'écrire pour l'élément T_k :

$$\sum_{T_n \in \mathcal{N}(k)} \left(\int_{T_k \cup T_n} \sigma \mathbf{n} ds \right) + b V_k \dot{\mathbf{u}}_k + \mathbf{f}_{ext} - \rho V_k \ddot{\mathbf{u}}_k = 0 \tag{2.30}$$

où $\mathcal{N}(k)$ contient les élément voisins de T_k.

On s'aperçoit alors que l'équation obtenue est quasiment la forme forte de l'équation de la dynamique 2.25. En effet, les termes en vitesse et accélération sont les mêmes, au scalaire V_k près. Seul le terme faisant intervenir la contrainte σ est différent. Cependant, dans l'équation 2.25, le terme $\bigtriangledown \cdot \sigma$ n'est autre que le flux de la contrainte à travers la surface d'un volume infiniment petit[FLS63]. Or le terme $\sum_{T_n \in \mathcal{N}(k)} \left(\int_{T_k \cup T_n} \sigma \mathbf{n} ds \right)$ est justement le flux de la contrainte à travers le volume T_k, soit la force créée par la contrainte et s'appliquant sur le centre.

Remarque La méthode des VOLUMES FINIS peut donc être considérée comme une méthode travaillant directement avec la forme forte d'une équation, les petits volumes T_k étant considérés comme infinitésimaux.Son caractère local, sa simplicité de compréhension et d'implémentation la rendent comparable aux méthodes masses-ressorts, et en font une alternative physique intéressante.

Dans la pratique, cette méthode, utilisant le théorème de la divergence, assure la conservation du flux et a donc été principalement utilisée pour la simulation de fluides. Cependant, TERAN [TBHF03] a proposé son utilisation pour simuler des muscles, mettant en avant la simplicité d'implémentation et d'optimisation découlant du caractère intuitif de cette méthode. Il prouve aussi, que dans lorsqu'on utilise des fonctions d'interpolation linéaires, ÉLEMENTS FINIS et VOLUMES FINIS sont identiques.

2.4.3 Les modèles Différences Finies

A l'instar des modèles éléments finis, les méthodes de résolution par différences finies cherchent à discrétiser une équation différentielle pour rendre son calcul réalisable par des méthodes numériques. Pour cela, au lieu de se reposer sur l'intégration sur un maillage d'éléments, la méthode des différences finies s'appuie sur l'utilisation d'opérateurs de différentiation associés à une grille. Ainsi, l'objet dont on simule la déformation est approximé par une grille aux nœuds de laquelle la fonction à calculer sera évaluée. Cette méthode a été employée pour la première fois dans [TPBF87] pour simuler des corps élastiques, puis a été étendue à la plasticité et la fracture dans [TF88, TW88].

Remarque Bien qu'efficaces, les schémas de type différences finies souffrent du manque de souplesse inhérent à l'utilisation d'une grille : il est difficile d'approximer des objets à la géométrie complexe. Cela rend aussi la gestion des conditions aux limites problématique. Enfin, ce type de méthode ne permet pas de rendre compte finement des lois constitutives des matériaux, contrairement aux éléments finis. C'est pourquoi cette méthode est généralement confinée au traitement de problèmes temporels, comme la construction de schéma de time-stepping.

2.4.4 Les modèles Éléments Frontières

Toutes les méthodes vues jusqu'à présent sont basées sur la discrétisation volumique du domaine étudié. Cela implique la résolution de systèmes numériques relativement grands, avec l'avantage que la déformation de l'objet est connue pour l'ensemble du domaine Ω.

Cependant, lorsque l'on simule des corps déformables, que ce soit pour la simulation médicale ou le prototypage virtuel, souvent seule la connaissance des déplacements en surface est nécessaire. L'utilisation de méthodes volumiques entraîne donc un surcoût inutile.

C'est pourquoi la méthode Éléments Frontières, permettant de calculer uniquement les déplacements en surface, a été développée. Cette méthode est basée sur l'utilisation de solutions fondamentales[2] associées à un opérateur différentiel linéaire $\mathcal{D}$.

Solution Fondamentale

La solution fondamentale pour l'équation différentielle

$$\mathcal{D}F(\mathbf{x}) = 0 \tag{2.31}$$

est la solution G à l'équation :

$$\mathcal{D}G(\mathbf{x}) = \delta(\mathbf{x} - \eta), \delta \text{ étant la distribution de Dirac.} \tag{2.32}$$

Cette solution fondamentale G, est aussi appelée fonction de Green[3]. Son intérêt réside dans la propriété suivante :

[2] La généralisation des fonctions de Green. Voire Section 2.5.1

[3] Les fonctions de Green ont été introduites par George Green alors qu'il travaillait sur l'électromagnétisme.

Soit $\phi(x)$ solution de l'equation differentielle $\mathcal{D}\phi(x) = j(x)$, alors :

$$\phi(x) = (G * j)(x) = \int G(x-y)j(y)dy \tag{2.33}$$

C'est à dire que, connaissant la function de GREEN pour l'opérateur différentiel $\mathcal{D}$ et pour une condition aux limites données, il est possible de calculer la fonction $\phi(x)$ par convolution entre G et j.

Dit autrement, si l'on calcule les fonctions de Green associées à $\mathcal{D}$ pour différentes conditions aux limites, on peut alors calculer par superposition linéraire toutes les solutions engendrées par superpositions de ces contraintes. Ainsi, dans le cas de la simulation de corps déformables, on obtient une base représentant toutes les déformations possibles d'un object soumis à des contraintes fixées.

Fonctions de Green et élastostatique

L'utilisation des fonctions de Green est directement envisageable dans le cas de la simulation de corps déformables, dans le mesure où, pour de petits déplacements, le comportement d'un objet déformable est régi par une équation différentielle linéaire. C'est le cas par exemple des modèles élastostatiques linéaires.

En ne conservant que la partie statique de l'équation 2.25, et en y injectant l'expression 2.23, l'équation de NAVIER, qui décrit le comportement statique d'un objet déformable, est obtenue.

L'intérêt de cette équation réside dans la connaissance, sous leur forme analytique, des solutions fondamentales, ou fonction de GREEN, connues aussi pour cette équation sous le nom de solutions fondamentales de KELVIN. A l'aide de ces fonctions, la déformée engendrée par l'application d'une contrainte en position en n'importe quel point de l'objet est connue. Pour un objet discrétisé, le système matriciel suivant est alors obtenu :

$$\mathbf{Au} = \mathbf{Bt} + \mathbf{b} \tag{2.34}$$

où $\mathbf{b}$ sont les forces volumiques, et $\mathbf{t}$ les contraintes sur les nœuds en surface.

Boundary Value Problem

Le système précédent possède $2n$ inconnues pour seulement n équations. Il faut donc fixer n inconnues pour obtenir un système inversible. Un tel système est décrit par un couple (Λ_u, Λ_p) d'ensembles d'indices. Les indices de Λ_u précisent quels nœuds voient leur déplacements imposés, tandis que Λ_p précise quels nœuds ont une pression imposées. Ce couple est nommé BOUNDARY VALUE PROBLEM, BVP.

Les déplacements libres $\mathbf{u}_j$, $j \in \Lambda_p$ et les pressions libres $\mathbf{p}_j$, $j \in \Lambda_u$ sont regroupés dans $\mathbf{v}$, tandis que les déplacements contraints $\mathbf{u}_j$, $j \in \Lambda_u$ et pressions contraintes $\mathbf{p}_j$, $j \in \Lambda_p$ sont stockés dans $\bar{\mathbf{v}}$.

$$\mathbf{v} = \begin{cases} u_j,\ j \in \Lambda_p \\ p_j,\ j \in \Lambda_u \end{cases} \quad \bar{\mathbf{v}} = \begin{cases} u_j,\ j \in \Lambda_u \\ p_j,\ j \in \Lambda_p \end{cases} \quad \Lambda_u \cap \Lambda_p = \emptyset \tag{2.35}$$

Ce qui permet de construire le système inversible :

$$\mathbf{G}\mathbf{v} + \bar{\mathbf{G}}\bar{\mathbf{v}} = \mathbf{b} \tag{2.36}$$

$$\mathbf{G}\mathbf{v} = \mathbf{z} \tag{2.37}$$

Avec $\mathbf{G}$ contenant les colonnes de $\mathbf{A}$ correspondant aux nœuds $j \in \Lambda_p$ et les colonnes de $-\mathbf{B}$ correspondant aux nœuds $j \in \Lambda_u$, et

$$\mathbf{z} = \bar{\mathbf{G}}\bar{\mathbf{v}} + \mathbf{b} \tag{2.38}$$

Où $\bar{\mathbf{G}}$ contient les colonnes de $\mathbf{A}$ correspondant aux nœuds $j \in \Lambda_u$ et les colonnes de $-\mathbf{B}$ correspondant aux nœuds $j \in \Lambda_p$.

Remarques Ainsi, on peut calculer la déformation de la surface d'un objet soumis à des forces externes et à des contraintes en surfaces. Cependant, même s'il est possible de moduler l'amplitude des contraintes, il est impossible de les déplacer, ce qui restreint drastiquement les possibilités dans le cadre de simulations interactives. Tout changement dans la position ou la nature des contraintes nécessite de reconstruire le système 2.36. De plus, étant donné la structure des matrices obtenues, même si leur dimension est inférieure à celle qu'on aurait avec les FEM, ces derniers s'avèrent souvent plus efficaces car moins gourmand en mémoire, et plus facilement optimisables en raison du motif des matrices creuses générées.

Il est important de noter que les modèles BEM ne peuvent pas simuler des matériaux hétérogènes. Seuls des matériaux homogènes isotropes peuvent être modélisés. De plus, ces modèles ne sont valables que pour de petits déplacements puisqu'on ne peut modéliser que des déformations linéaires. Enfin, il n'est pas possible de simuler la dynamique des déformations car les fonctions de GREEN de ce problème ne sont pas connues.

2.4.5 Les modèles Élements Finis

Les modèles éléments finis sont la réponse classique au problème de la résolution d'équation différentielles elliptiques pour des objets quelconques. Ils permettent de discrétiser l'équation 2.25 de manière à obtenir un système qui puisse être résolu numériquement.

Pour cela, la forme faible de l'équation 2.25 est dérivée. C'est à dire qu'en utilisant les moyennes pondérées, le théorème des travaux virtuels, ou une analyse variationnelle, on passe d'une équation différentielle, appelée forme forte, avec des conditions aux limites à une intégrale, appelée forme faible, sur un domaine volumique Ω. Pour cela, on utilise une fonction de test, ou fonction de pondération $\boldsymbol{v}$. $\boldsymbol{v}$ doit être *compatible*, ou *cinématiquement admissible*, ce qui signifie qu'elle doit respecter les conditions aux limites essentielles, ou conditions de DIRICHLET.

$$\int_\Omega \boldsymbol{v} \cdot \left(\rho \cdot \frac{\partial^2 \mathbf{u}}{\partial t^2} + \nabla \cdot \sigma + b \cdot \frac{\partial \mathbf{u}}{\partial t} + f \right) \cdot d\Omega = 0, \forall v \tag{2.39}$$

Ensuite, la dérivation portant sur $\boldsymbol{\sigma}$ est transférée sur $\boldsymbol{v}$ en utilisant le théorème de la divergence.

L'équation obtenue est alors discrétisée de manière à pouvoir être intégrée numériquement. Pour cela, on utilise des approximations $\boldsymbol{u}_h$ et $\boldsymbol{v}_h$ de $\boldsymbol{u}$ et $\boldsymbol{v}$ qui soient des combinaisons linéaires de fonctions de bases connues et facilement intégrables.

L'élément finis

Pour construire ces fonctions d'approximation il est possible de s'appuyer sur une discrétisation Ω_h du domaine Ω. Cette discrétisation est généralement obtenue à l'aide d'une partition $\{T_k\}_{1<k<nb_{el}}$ du domaine en tétraèdres T_k[4]. C'est à dire qu'on découpe le domaine Ω en un ensemble de cellules disjointes, généralement des tétraèdres. Ces tétraèdres sont ensuite munis de fonctions d'interpolation $\theta_{1,2,3,4}$ qui permettront de construire $\boldsymbol{u}_h$ et $\boldsymbol{v}_h$.

L'élément finis associé au tétraèdre T peut alors être définit comme le couple $\{T, \Sigma\}$, où Σ est un ensemble de 4 fonctions d'interpolation $\theta_{1,2,3,4}$ associées avec chacun des sommets $\mathbf{x}_{1,2,3,4}$ du tétraèdre.

Interpolateurs local et global

À l'aide d'un élément fini $\{T, \Sigma\}$ il est possible de définir un interpolateur local $\mathcal{I}_L$ qui va permettre de calculer une grandeur physique à l'intérieur du tétraèdre T. Il est défini comme suit :

$$\mathcal{I}_l : v \rightarrow \sum_{i=1}^{4} v(x_i)\theta_i \tag{2.40}$$

De manière similaire on peut définir l'interpolateur global $\mathcal{I}_h$ sur l'ensemble du domaine Ω comme :

$$\mathcal{I}_h : v \rightarrow \sum_{T\in\{T_k\}} \sum_{i=1}^{4} v(x_i)\theta_i \tag{2.41}$$

Interpolation du déplacement

L'interpolateur peut maintenant être utilisé pour construire une approximation u_h du déplacement u :

$$\mathbf{u}_h = \mathcal{I}_h\mathbf{u} = \sum_{T\in\{T_k\}} \sum_{i=1}^{4} \mathbf{u}(\mathbf{x}_i)\theta_i^T \tag{2.42}$$

Cette approximation, dont les fonctions de base $\theta^T_{1,2,3,4}$ sont définies pour le support compact constitué par le tétraèdre T, permet d'utiliser la linéarité de l'intégrale pour faciliter l'intégration de l'équation 2.39. L'intégration est alors faite tétraèdre par tétraèdre.

Interpolation de la géométrie

Nous sommes donc parvenu à découper l'intégrale 2.39 sur le domaine Ω en une somme d'intégrales sur Ω_h. Chaque terme de cette somme doit donc être intégré sur le volume défini par le tétraèdre T. Pour simplifier plus encore les calculs, il est intéressant d'effectuer un changement de variable pour intégrer chaque terme de l'intégrale non plus sur T, mais sur un tétraèdre de référence $\hat{T}$. $\hat{T}$ est décrit dans la Figure 2.10.

[4] Cette partition pourrait aussi être obtenue avec un autre type de volume, comme un cube par exemple

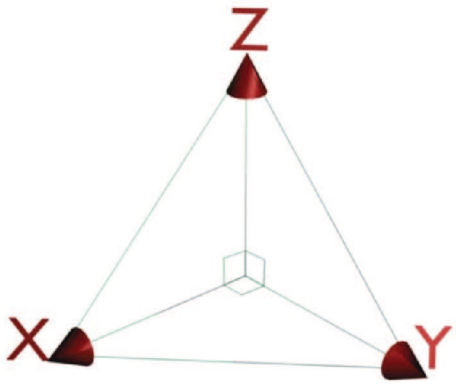


FIG. 2.10 – le tétraèdre de référence $\hat{T}$.

Pour cela, on réutilise l'interpolateur global défini en 2.41 pour interpoler la géométrie d'un tétraèdre T_k depuis $\hat{T}$:

$$\mathbf{x}^h = \mathcal{I}\mathbf{x} = \sum_{T \in \{T_k\}} \sum_{i=1}^{4} \mathbf{x}(\mathbf{x}_i)\theta_i^T \tag{2.43}$$

Discrétisation de l'équation

L'équation 2.39 peut donc finalement s'écrire sous la forme :

$$\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{B}\dot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{x} = \boldsymbol{f}_{int}(\boldsymbol{x}) \tag{2.44}$$

2.5 Modèles physiques et optimisés pour la simulation interactive

Les méthodes décrites ci-dessus assurent le calcul de déformées relativement *physiques*. Cependant, elles souffrent de diverses limitations, comme l'impossibilité de simuler de grands déplacements ou un temps de calcul prohibitif pour des objets complexes.

Le but de cette section est donc de voir quelles optimisations peuvent être employées pour pallier à ces restrictions, et quelles sont les limitations introduites par ces optimisations.

2.5.1 Les modèles basés sur les fonctions de Green et les Capacitance Matrix Algorithmes

Comme nous l'avons ci-dessus, bien que réduisant la complexité des modèles simulés d'un ordre de grandeur, les méthodes ÉLEMENTS FRONTIÈRES ne sont finalement pas tellement plus performantes, et ce en raison de la structure du système matriciel généré. De plus, Le stockage des fonctions de Green, sous forme matricielle, requière une quantité de mémoire qui s'accroît rapidement avec la complexité des modèles simulés. Il n'est donc pas possible de déformer des objets au maillage important. Enfin, et c'est le point le plus bloquant, elles souffrent de l'impossibilité de changer dynamiquement la nature des contraintes, ce qui empêche par exemple de gérer des contacts de manière interactive.

Pour contourner ces limitations Doug James propose dans sa thèse deux solutions, dans le cas de l'élastostatique, pour éviter ces problèmes :

1. Il propose d'utiliser la formule de Sherman-Morrison-Woodbury pour mettre à jour les fonctions de Green rapidement, et permettre ainsi de changer les contraintes.
2. Il montre que l'utilisation d'un maillage hiérarchique couplé à une transformée en ondelette de seconde génération permet de réduire drastiquement la mémoire consommée.

Pour cela, il propose de calculer hors-ligne les fonctions de Green $\mathbf{\Xi}$ discrètes associées à un BVP de la façon suivante :

$$\mathbf{\Xi} = -\mathbf{G}^{-1}\bar{\mathbf{G}} \tag{2.45}$$

De telle sorte qu'on puisse calculer le $\mathbf{v}$ de l'équation 2.35 directement depuis $\bar{\mathbf{v}}$:

$$\mathbf{v} = \mathbf{\Xi}\bar{\mathbf{v}}, \text{ ou} \tag{2.46}$$

$$\mathbf{v} = \mathbf{\Xi}\bar{\mathbf{v}} + \mathbf{G}^{-1}\mathbf{b} \tag{2.47}$$

L'avantage de cette formulation réside dans la possibilité de mettre à jour la matrice $\mathbf{\Xi}$ en utilisant des algorithmes de type Capacitance Matrix Algoritmes, CMA.

Il est donc possible de précalculer les fonctions de Green associées à cette équation différentielle pour plusieurs contraintes, et mieux encore, il est possible de les mettre rapidement à jour. Ainsi, si $S = S_i$ contient l'ensemble des indices des nœuds dont on inverse la nature de la contrainte, i.e. les déplacements deviennent des tractions et inversement, et si $\mathbf{E}$ est la matrice permettant de faire les permutations associées, alors :

$$\mathbf{G}_{new} = \mathbf{G} + (\bar{\mathbf{G}} - \mathbf{G})\mathbf{E}\mathbf{E}^T \tag{2.48}$$

$$\bar{\mathbf{G}}_{new} = \bar{\mathbf{G}} + (\mathbf{G} - \bar{\mathbf{G}})\mathbf{E}\mathbf{E}^T \tag{2.49}$$

James *et al.* ont montré dans [JP99] qu'il est possible de rapidement mettre à jour la matrice des fonctions de Green $\mathbf{\Xi}$ selon :

$$\mathbf{\Xi}_{new} = \mathbf{\Xi} + (\mathbf{\Xi} + \mathbf{I})\mathbf{E}\mathbf{C}^{-1}\mathbf{E}^T(\mathbf{\Xi} - \mathbf{I}) \tag{2.50}$$

Avec $\mathbf{C} = -\mathbf{E}^T\mathbf{\Xi}\mathbf{E}$.

Il est donc possible de changer dynamiquement et rapidement la nature des contraintes appliquées à un corps, ce qui autorise à gérer des contacts entre ce corps et son environnement. Cependant, la matrice $\mathbf{\Xi}$ reste pleine et son stockage devient prohibitif dès lors que le système simulé devient trop complexe.

C'est pourquoi James propose dans [JP03] de stocker les $\mathbf{\Xi}$ de manières hiérarchiques en utilisant des transformées en ondelettes. A terme, ce type de modèle peut aussi être utilisé pour simuler des chaînes de corps déformables [JP02].

Remarque En dépit ce ces améliorations, l'utilisation de fonctions de Green et de CMA est assez limitée. La principale limitation découle de la limitation à des petits déplacements du fait de la linéarité du modèle. Ensuite, il n'est pas possible de simuler la dynamique de la déformée. Seule la statique est simulable. Enfin, les matériaux doivent être homogènes, ce qui n'est pas toujours le cas.

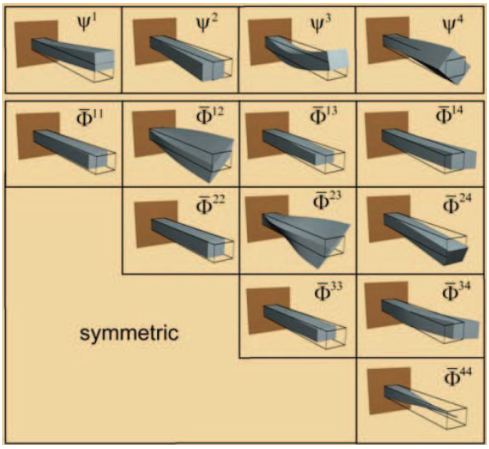


FIG. 2.11 – Les principaux modes propres d'une poutre encastrée et leurs dérivées. Issue de [BJ05].

2.5.2 Réduction de modèle

Une autre piste qui a été explorée pour réduire les temps de calcul durant la simulation est la réduction du nombre de degré de liberté du modèle. On sait par exemple, grâce à l'analyse modale, qu'un objet va se déformer suivant des modes qui lui sont propres, et qu'un ensemble réduit de ces modes sera particulièrement sollicités, et concentrera une bonne partie de l'énergie de déformation.

L'idée principale consiste donc à exprimer le déplacement $\mathbf{u}$ avec un nombre de paramètres $\mathbf{q}$ réduit :

$$\mathbf{u} = \mathbf{U}\mathbf{q}, \text{ avec } dim(\mathbf{q}) < dim(\mathbf{u}) \tag{2.51}$$

où $\mathbf{U}$ contient les modes de déformation ν_i.

JERNEJ BARBIČ et JAMES ont proposé cette méthode dans [JP04].

Extraction des modes de déformation

Il existe deux voies pour déterminer les modes de déformation d'un objet :

1. De manière analytique : à l'aide d'un modèle formel de déformation, on tente par voie mathématique de calculer les modes propres.
2. De manière statistique : l'objet à simuler n'a pas fait l'objet d'une modélisation, mais cet objet existe. L'objet est soumis à une batterie de tests et une analyse statistique révèle ses modes de déformations principaux.

Extraction par le calcul Il est déjà possible depuis longtemps d'extraire les modes de déformation d'un objet lorsque celui-ci ne se déforme que très peu autour de sa configuration d'origine. C'est ce que font les simulateurs basés sur l'Analyse Modale Linéaire qui calculent les modes propres, ou modes de déformation $\mathbf{\Psi}_i$ en résolvant :

$$\mathbf{K}\left(\mathbf{0}^{3n}\right)\mathbf{\Psi}_i = \lambda \mathbf{M}\mathbf{\Psi}_i \tag{2.52}$$

Cependant, cette méthode n'est plus appliquable dans le cas grands déplacements où le déplacement n'est plus linéaire.

Jernej Barbič propose dans [BJ05] d'étendre cette technique à la détermination de modes de déformation non-linéaire. Il montre que l'on peut construire ces modes pour une configuration déformée facilement. Pour cela, il évalue la dérivée directionnelle de $\Psi(\mathbf{x}_0)$, i.e. les modes propres autour de la position $\mathbf{x}_0$, à l'origine, et dans les directions $\mathbf{u_0}(\mathbf{p}) = \sum_i p_i \mathbf{\Psi}_i$, où $\boldsymbol{p}$ le vecteur des *facteurs de participation modale* :

$$\Phi^{ij} = \frac{\partial}{\partial p_j}\left(\Psi^i\left(\sum_l \Psi^l p_l\right)\right) \tag{2.53}$$

Et étend ainsi les modes de déformations à des configurations quelconques $\mathbf{u}$ à l'aide de la formule suivante :

$$\mathbf{u}(\mathbf{p}) = \sum_i \Psi_i p_i + \frac{1}{2}\Phi^{ij} p_i p_j + O(\mathbf{p}^3) \tag{2.54}$$

L'orthonormalisation de l'espace obtenu étant coûteuse, une dernière Analyse en Composante Principale pondérée par la masse, ou mass-PCA, est appliquée sur ces modes.

Parallèlement, il établit aussi que lorsqu'on utilise le tenseur de Green comme tenseur de déformations, les forces internes sont simplement des polynômes cubiques dont les coefficient peuvent être calculés hors ligne. La matrice de raideur tangente étant la jacobienne des forces internes, son calcul est direct. Cela permet de simuler rapidement la déformation dynamique géométriquement non-linéaire de matériaux linéaires.

Une autre approche envisageable pour réduire de manière analytique la dimension d'un système repose sur le transfert et la résolution de ce système dans le domaine fréquentiel, en ne conservant que les basses fréquences. C'est ce que proposent Rong dans [RCG08] pour des surfaces déformables et Martin dans [MKB+08] pour des volumes. Pour cela, il calcule les bases harmoniques $(\boldsymbol{H}^k, \lambda_k)$ solutions de :

$$-\Delta \boldsymbol{H}^k = \lambda_h \boldsymbol{H}^k \tag{2.55}$$

pour $k \in [|0, n-1|]$, où n est le nombre de nœuds d'un maillage surfacique. Ensuite, en ne conservant que les m bases harmoniques associées avec les plus basses fréquences, il construit la matrice $\boldsymbol{H} = [\boldsymbol{H}^0, \cdots, \boldsymbol{H}^{m-1}]$. Cette matrice et sa transposée peuvent ensuite être utilisées pour respectivement post et pré-multiplier les matrices du système et réduire la dimension du problème de n à m, avec généralement $m \ll n$.

Choi et Ko proposent dans [CK05] une méthode assez similaire à celle de James, sauf qu'ils utilisent un modèle corotationnel nodal pour autoriser des déformations géométriquement non-linéaires. Cependant, cette méthode ne permet pas de mouvements libres ; l'objet doit nécessairement être encastré quelque part.

Extraction statistique Lorsque l'on ne dispose pas d'un modèle théorique de déformation, il est possible de construire l'espace des déformations à l'aide d'une analyse statistique. Il suffit d'avoir à disposition un ensemble de déplacements caractéristiques des déformations acceptables par l'objet étudié.

Plusieurs possibilités sont envisageables pour construire cette base de déplacements. Si l'on dispose de l'objet à déformer, et qu'il peut être soumis à une batterie de tests, alors on le déforme

FIG. 2.12 – Découpage en sous-domaines. Issu de [HLB+06]

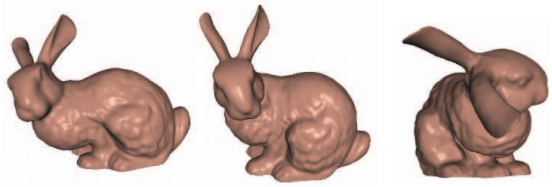


FIG. 2.13 – Un exemple de déformation. Issu de [HLB+06]

en notant les déplacements obtenus. Si l'on ne dispose pas de l'objet ou s'il n'est pas possible de l'intrumentaliser, alors il est possible de le simuler de manière interactive avec un simulateur linéaire interactif. Les efforts appliqués lors de cette simulation interactive sont mémorisés. Ils sont ensuite fournis en entrée à un simulateur non linéaire qui calculera les déplacements réels de l'objet. JAMES détaille cette approche dans [BJ05].

Une fois les déplacements correspondants à un cas des déformations réalistes obtenus, ils sont soumis à une Analyse en Composante Principale de type mass-PCA, et les modes de déformations ν en sont déduits.

Remarques Cette méthode est intéressante par divers aspects. Premièrement, la réduction drastique du nombre de paramètres accélère considérablement la simulation. Deuxièmement, la possibilité de simuler des objets sans même avoir à les modéliser est intéressante et nous rapproche des méthodes par apprentissage.

Cependant, le fait même de réduire le modèle en sélectionnant uniquement certains modes de déformation limite la vraisemblance de la simulation. Par exemple, les modes de déformations généralement retenus correspondent à des basses fréquences. Il est donc impossible de simuler des déformations locales avec ce genre de méthodes sans ajouter beaucoup de modes et perdre le bénéfice de la réduction de modèle. Autre inconvénient, les modes de déformation sont calculés pour des conditions aux limites précises qui ne peuvent être modifiées en ligne.

2.5.3 La Décomposition de Domaine

Lorsque des objets se déforment, même dans de grandes proportions, les déformations restent souvent de faible amplitude. Cette observation ouvre la voie à l'utilisation de méthode de type DOMAIN DECOMPOSITION METHOD (DDM), qui consistent à découper le domaine étudié en sous-domaines (Voir figure 2.12). La solution au problème est alors calculée suivant un modèle simplifié au niveau de chaque sous-domaine. Un solveur se charge ensuite d'interfacer ces solutions pour fournir une solution globale.

HUANG a adapté dans [HLB+06] les DDM à la simulation interactive de corps déformables. Pour cela, il découpe le domaine Ω en sous domaine disjoints Ω_i, et pour chacun des domaines, il résoud :

$$\boldsymbol{R}_i \left(\boldsymbol{M}_i + h\boldsymbol{D}_i + h^2 K_i\right) \boldsymbol{R}_i^T \Delta\dot{\boldsymbol{q}}_i = h\left(\boldsymbol{f}_{ext,i} - \boldsymbol{D}_i\dot{\boldsymbol{q}}_i - \boldsymbol{K}_i(\boldsymbol{q}_i + h\dot{\boldsymbol{q}}_i)\right) \quad (2.56)$$

$$\boldsymbol{H}_i\Delta\dot{\boldsymbol{q}}_i = \boldsymbol{f} \quad (2.57)$$

où R_i est une estimation de la rotation globale du sous-domaine Ω_i et ou $\boldsymbol{q}_i$ contient les degrés de liberté du système. Pour chaque domaine Ω_i, les matrices $\boldsymbol{M}_i$, $\boldsymbol{D}_i$ et $\boldsymbol{K}$ sont constantes. L'inverse de $\boldsymbol{H}_i$ peut donc être en partie précalculée. L'équation 2.56 peut donc être résolue directement, sans faire appelle à un solveur. Seules les rotations doivent être prises en compte durant la simulation.

A ces équation, HUANG ajoute une contrainte portant sur les déplacements aux interfaces des différents domaines. En ces points là, les déplacement doivent impérativement être égaux. Cette contrainte est assurée par des multiplicateurs de LAGRANGE $\boldsymbol{\lambda}$:

$$\begin{pmatrix} \boldsymbol{H} & \boldsymbol{B}^T \\ \boldsymbol{B} & 0 \end{pmatrix} \begin{pmatrix} \Delta\dot{\boldsymbol{q}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{pmatrix} \text{ où } \boldsymbol{H} = \begin{pmatrix} \boldsymbol{H}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \boldsymbol{H}_n \end{pmatrix} \tag{2.58}$$

et $\boldsymbol{B}$ assure le respect de la contrainte.

En procédant ainsi, le temps de résolution de l'équation 2.56 est théoriquement réduit puisqu'il n'est plus nécessaire d'avoir recourt à un solveur, l'inverse des matrices $\boldsymbol{H}_i$ pouvant être en partie précalculée. Cependant, les matrices $\boldsymbol{H}_i$ sont creuses, tandis que les matrices inverses sont pleines. Du coup, le gain de temps n'est pas significatif. C'est pourquoi HUANG utilise une Analyse en Composante Principale de type cluster pour réduire la dimension de ces matrices inverses, ce qui lui permet d'obtenir des simulation interactives.

Bien qu'efficaces, ces méthodes souffrent d'une limitation majeure : le découpage en sous-domaines doit garantir qu'effectivement ces sous-domaines subissent de petites déformations. Or il est difficile d'automatiser ce genre de découpage.

Remarque Cette méthode est très proche d'une méthode élément finis corotationnelle, dans laquelle on prend en compte les rotations de chaque éléments du maillage. Cependant, du fait de l'assemblage des matrices élémentaires, il est impossible de pré-inverser, même en partie, le membre gauche du schéma d'intégration. Les DDM rendent cela possible en dédoublant les nœuds communs, et utilisant les multiplicateurs de LAGRANGE pour assurer l'égalité en déplacement.

2.5.4 Le modèle corotationnel

L'idéal pour nos simulations serait de pouvoir utiliser un modèle ÉLÉMENT FINIS de manière interactive tout en autorisant de larges déformations. C'est justement ce que proposent les modèles ELÉMENT FINIS COROTATIONNELS, à condition que les contraintes restent faibles dans le matériau.

Ce type de modèle pare à la non-invariance du tenseur linéaire de CAUCHY en prenant en compte les rotations locales de l'objet déformé. Nous présentons ces modèles dans le cas où les rotations sont évaluées au niveau des éléments du maillage, dans la mesure où cela accroît la stabilité. Néanmoins, il est possible d'évaluer ces rotations au nœuds comme présenté dans [MDM+02]. Nous y reviendrons dans le chapitre 4.

Invariance aux rotations des Tenseur de Green et Tenseur de Cauchy

Comme vu dans la sous-section 2.3.1 sur la théorie de l'élasticité, le tenseur de GREEN ϵ_G à l'expression suivante :

$$\epsilon_G = \frac{1}{2}\left(\bigtriangledown \boldsymbol{u}^T + \bigtriangledown \boldsymbol{u} + \bigtriangledown \boldsymbol{u}^T \bigtriangledown \boldsymbol{u}\right) \tag{2.59}$$

Ce qui peux aussi s'écrire sous la forme :

$$\epsilon_G = \frac{1}{2}\left(\bigtriangledown \boldsymbol{\Phi}^T \bigtriangledown \boldsymbol{\Phi} - \boldsymbol{I}\right) \tag{2.60}$$

où $\boldsymbol{\Phi}(x) = \boldsymbol{x} + \boldsymbol{u}$ est la fonction qui lie la position au repos $\boldsymbol{x}_0$ à la position courante $\boldsymbol{x}$.

Le tenseur de CAUCHY ϵ_C est la linéarisation du tenseur de GREEN et s'écrit donc :

$$\epsilon_C = \frac{1}{2}\left(\bigtriangledown \boldsymbol{u}^T + \bigtriangledown \boldsymbol{u}\right) \tag{2.61}$$

Si l'on applique une rotation $\boldsymbol{R}$ au tenseur de GREEN, on obtient :

$$\epsilon_G^R = \frac{1}{2}\left(\bigtriangledown (\boldsymbol{R}\boldsymbol{\Phi})^T \bigtriangledown (\boldsymbol{R}\boldsymbol{\Phi}) - \boldsymbol{I}\right) \tag{2.62}$$

$$\epsilon_G^R = \frac{1}{2}\left(\bigtriangledown \Phi^T \boldsymbol{R}^T \boldsymbol{R} \bigtriangledown \boldsymbol{\Phi} - \boldsymbol{I}\right) \tag{2.63}$$

$$\epsilon_G^R = \frac{1}{2}\left(\bigtriangledown \Phi^T \bigtriangledown \boldsymbol{\Phi} - \boldsymbol{I}\right) = \epsilon_G \tag{2.64}$$

Le tenseur de GREEN est donc invariant aux rotations. Ce n'est pas le cas du tenseur de CAUCHY, qui lui est affecté par les rotations, et qui introduit des forces fantômes conduisant à une déformée aberrante.

Pour nos applications, puisque l'on veut simuler de grande déformée, l'utilisation du tenseur de GREEN semble donc nécessaire. Cependant, ce tenseur étant non-linéaire, la résolution des équations de la dynamique associée requiert l'utilisation d'un solveur non-linéaire de type NEWTON-RAPHSON si l'intégration est implicite. Malheureusement, ce type de solveur est trop lent pour être utilisé dans un contexte interactif.

Abstraction des rotations

L'idée sous-jacente aux modèles corotationnels consiste simplement à évaluer à chaque pas de temps la rotation de chaque élément ou nœud. Cela permet de s'abstraire de la rotation de l'objet en ramenant l'élément dans son orientation d'origine. Les calculs de déformation sont ensuite faits dans ce repère, en utilisant les matrices $\boldsymbol{M}$, $\boldsymbol{B}$ et $\boldsymbol{K}$ de l'équation 2.44 mises à jour à l'aide des rotations comme suit :

Pour chaque élément i, la matrice de rotation associée R_i est estimée, et la matrice de raideur élémentaire $\boldsymbol{K}_i^0$ est mise à jour ainsi : $\boldsymbol{K}_i = \boldsymbol{R}^T \boldsymbol{K}_i^0 \boldsymbol{R}$. Il est fait de même pour les matrices de masse $\boldsymbol{M}_i$ et d'amortissement $\boldsymbol{B}_i$. Les matrices globales $\boldsymbol{M}$, $\boldsymbol{B}$ et $\boldsymbol{K}$ sont ensuite assemblées. La déformée peut maintenant être calculée en utilisant un solveur linéaire classique comme, par exemple, un GRADIENT CONJUGUÉ.

L'algorithme 2 détaille toutes les étapes du calcul.

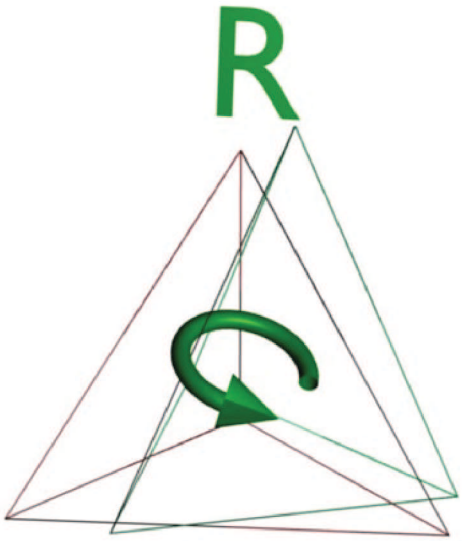


FIG. 2.14 – Abstraction des rotations.

Algorithm 2 Calcul de la déformée avec un modèle corotationnel

1: **while** 1 **do**
2: Estimer les rotations $\boldsymbol{R}_i$ pour chaque élément
3: Mettre à jour les matrices élémentaires $\boldsymbol{M}_i$, $\boldsymbol{B}_i$ et $\boldsymbol{K}_i$
4: Assembler les matrices globales $\boldsymbol{M}$, $\boldsymbol{B}$ et $\boldsymbol{K}$
5: Calculer le déplacement en utilisant un solveur linéaire classique
6: **end while**

Remarque Dans la pratique, les matrices globales $\boldsymbol{M}$, $\boldsymbol{B}$ et $\boldsymbol{K}$ ne sont pas assemblées. Il est plus efficace de calculer les forces éléments par éléments en tenant compte des rotations.

Estimation des rotations

Les rotations peuvent être estimées de différentes manières. Müller et al. proposent dans [MDM+02] de se baser sur un calcul géométrique pour estimer les rotations. Bien que détaillée pour des modèles corotationnels nodaux, leur méthode peut être appliquée directement au cas élémentaire. Pour cela, il faut calculer pour chaque élément la rotation permettant de passer du repère d'origine au repère courant.

Müller utilise dans [MG04] la matrice de transformation homogène entre la configuration de référence et la configuration courante d'un tétraèdre pour déterminer les $\boldsymbol{R}_i$.

Etzmuss, qui fut le premier à proposer dans la communauté Computer Graphics une approche basée sur des éléments, utilise une décomposition polaire de $\bigtriangledown\Phi$ pour en extraire la composante en rotation. De manière similaire, Hauth dans [HS04] propose de s'appuyer sur une décomposition QR de $\bigtriangledown\boldsymbol{\Phi}$. Ces deux méthodes sont plus coûteuses en temps de calcul mais s'avèrent plus stables dans la pratique. Cette contrainte peut être réduite en orthonormant $\bigtriangledown\boldsymbol{\Phi}$ par un procédé de Graham Schmidt, et ce au détriment de la précision du résultat. C'est ce que fait OpenTissue[5].

2.6 Conclusion

Ce chapitre a été l'occasion de brosser rapidement le portrait de l'ensemble des méthodes régulièrement employées par la communauté Computer Graphics. A travers une logique de présentation suivant une complexité et une modélisation plus proche de la physique croissantes, nous avons abordé dans un premier temps les méthodes rapides, assurant une déformation *visuellement* satisfaisante mais sans fondement physique.

Ensuite, conscient de la nécessité d'ancrer les résultats de nos simulations dans une réalité physique, nous nous sommes tournés vers la Mécanique des Milieux Continus et la Théorie de l'Élasticité pour modéliser cette réalité physique, et avons présenté les méthodes s'appuyant sur ces théories pour déformer des objets virtuels.

Enfin, ayant jugé de l'intérêt des méthodes physiques, mais ayant constaté que leur complexité en temps de calcul ne permettait un usage interactif que pour de petits modèles, ou pour des objets se déformant peu, nous avons étudié quels améliorations la communauté proposait pour accélérer ces méthodes, en notant quelles limitations ces améliorations apportaient.

A l'aune de ces informations, nous pouvons motiver notre choix pour le modèle corotationnel élémentaire, qui basé sur les Éléments Finis et la Théorie de l'Élasticité, est en accord avec la physique pour peu qu'il soit utilisé dans le cadre des grands déplacements et des petites contraintes. C'est de toutes les méthodes physiques bénéficiant de simplification en vue d'une accélération la meilleure suivant nos critères, puisqu'elle permet de simuler des grands déplacements, aussi bien ceux de basse fréquence que ceux de haute fréquence ; elle ne souffre aussi d'aucune limitation sur la géométrie de l'objet à déformer et permet de simuler des matériaux hétérogènes. Nous utiliserons donc cette méthode dans la suite de cette thèse.

[5] www.opentissue.org

Chapitre 3

Solveurs Multigrilles et Ondelettes

Sommaire

3.1 Introduction

L'étape la plus consommatrice en temps de calcul dans les simulations de corps déformables se situe dans l'étape d'intégration temporelle. Cette étape nécessite en effet la résolution d'un système d'équations de grande dimension. Dans ce chapitre, nous proposons plusieurs voies pour améliorer cette étape. Nous offrons notamment d'améliorer et la performance et la précision des solveurs multigrilles en accélérant le calcul des matrices intermédiaires et en utilisant un mailleur hiérarchique. Nous introduisons aussi l'utilisation de transformées en ondelettes 3D pour les solveurs hiérarchiques dans le cadre d'un modèle éléments finis.

Ce chapitre s'ouvre donc, après qu'ait été balayée dans le chapitre précédent la majorité des modèles et méthodes utilisés pour décrire la déformation d'un solide soumis à des efforts ou des contraintes, sur les moyens permettant d'intégrer ces modèles dans le temps.

Pour cela, nous précisons dans un premier temps les deux types de schémas d'intégration les plus courants : le schéma de type *explicite* et le schéma de type *implicte*. Ces deux types de schémas, construits à l'aide de la méthode des différences finies, permettent d'intégrer des équations dans le temps, et se distinguent par des propriétés différentes que nous préciserons.

Une fois le schéma d'intégration construit, il faut résoudre le système d'équations qu'il forme. Pour assurer cette résolution, il faut avoir recours à des solveurs, dont les principaux sont de type *directs, itératifs, multigrilles, hiérarchiques ou adaptatifs.*

À l'issue de ces deux étapes, *i.e.* après avoir choisi le schéma d'intégration et le solveur idoine, la déformation d'un modèle peut être simulée. Cependant, une fois encore, il apparaît que le choix du solveur, n'est pas immédiat. En effet, au regard des caractéristiques propres à chaque type de solveurs, ceux *multigrilles* semblent être un bon choix, bien qu'ils ils souffrent de deux limitations : ils ne sont pas très adaptés à des modèles représentés par des matrices creuses, or dans le cas des modèles éléments finis, les matrices utilisées sont justement creuses. De plus, du fait de l'utilisation de schémas de subdivision, ils ne permettent pas d'approximer des géométries complexes.

A ces deux limitations nous apportons une solution en proposant une construction rapide des matrices aux différentes résolutions, et en proposant l'idée d'un mailleur hiérarchique. Ces deux idées on fait l'objet d'un publication à l'International Symposium on Visual Computing [SDG07].

Enfin, la nature même des solveurs multigrilles, basée sur l'utilsation de multiples résolutions, amène rapidement à penser aux ondelettes. Aussi nous montrons comment lier solveurs hiérarchiques et transformées en ondelettes. Notamment, nous expliquons comment utiliser le Lifting Scheme pour construire des opérateurs de prolongation et restriction qui soient des transformées en ondelettes. Nous établissons ainsi que les opérateurs de prolongation et de restriction barycentriques peuvent être vus comme une transformée en ondelettes constructible à l'aide du Lifting Scheme.

Le plan suivi est donc le suivant : la section 3.2 présente les intégrateurs de types *explicite* et *implicite*, la section 3.3 détaille les différents type de solveurs permettant de résoudre les systèmes posés par ces intégrateurs. Viennent ensuite nos contributions, avec dans la section 3.4 notre méthode de construction rapide des matrices intermédiaires nécessaires aux solveurs multigrilles suivi de notre mailleur hiérarchique. La section 3.5 contient, elle, notre contribution sur l'utilisation de transformées en ondelettes dans les solveurs hiérarchiques.

3.2 Discrétisation et Intégration Temporelle

Dans les modèles présentés précédemment, certains ne modélisent qu'un comportement statique des matériaux, c'est à dire qu'ils sont toujours considérés comme étant à l'équilibre. À chaque pas de la simulation, la somme des forces est nécessairement nulle. Les états transitoires, dans lesquels l'objet n'est pas en équilibre ne sont pas considérés. D'autres modèles, par contre, cherchent à simuler aussi ces états transitoires. Dans ce cas, ils voient leur déformation décrite par la seconde loi de Newton, et évoluent dynamiquement dans le temps.

Ce type de modèle requiert l'intégration des équations de la dynamique dans le temps. Pour cela, un schéma d'intégration temporelle est construit à l'aide de la méthode des différences finies. Si ce schéma a pour inconnue l'accélération, et que les positions et vitesses sont extrapolées à

l'aide des valeurs aux pas de temps ou aux itérations précédents, alors ce schéma est qualifié d'*explicite*. À l'opposé, si les positions et vitesses au pas de temps courant sont les inconnues, alors ce schéma est nommé *implicite*. Le choix d'un schéma *explicite* ou *implicite* impacte sur la stabilité de la simulation, mais aussi sur le temps de calcul.

3.2.1 Les schémas de type explicite

Les schémas explicites sont construits simplement en s'appuyant sur les données issus des calculs au pas de temps ou itérations précédents.

Construction

Soit en partant de l'équation de la dynamique :

$$\boldsymbol{M}\ddot{\boldsymbol{x}} = \boldsymbol{F}_{ext} - \boldsymbol{F}_{int}(\boldsymbol{x}, \dot{\boldsymbol{x}}) \tag{3.1}$$

Avec $\boldsymbol{M}$ la masse du système, $\boldsymbol{x}$ et $\boldsymbol{v}$ ses position et vitesse, et $\boldsymbol{F}_{ext}$ et F_{ext} les forces internes et externes.

Il est alors possible d'écrire simplement, en posant $\boldsymbol{x}_i - \boldsymbol{x}_{i-1} = \Delta\boldsymbol{x} = h\dot{\boldsymbol{x}}_{i-1}$ et $\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_{i-1} = \Delta\dot{\boldsymbol{x}} = h\ddot{\boldsymbol{x}}_{i-1}$:

$$\begin{pmatrix} \Delta\boldsymbol{x} \\ \Delta\dot{\boldsymbol{x}} \end{pmatrix} = h \begin{pmatrix} \dot{\boldsymbol{x}}_{i-1} \\ \boldsymbol{M}^{-1}(\boldsymbol{F}_{ext} - \boldsymbol{F}_{int}(\boldsymbol{x}_{i-1}, \dot{\boldsymbol{x}}_{i-1})) \end{pmatrix} \tag{3.2}$$

Propriétés

Le calcul d'un tel schéma, dit d'Euler explicite, est immédiat et ne nécessite pas la résolution d'un système si la matrice de masse à fait l'objet d'un *mass lumping*, comme c'est généralement le cas. Il est donc particulièrement peu gourmand en temps de calcul. Par contre, ce type de schéma est peu stable, et nécessite des pas de temps d'autant plus courts que le système à résoudre est mal conditionné. Le gain en temps de calcul peut parfois être rendu négligeable par la nécessité de calculer de plus nombreux pas de temps. Autre limitation, plus les éléments du maillage sont petits, plus le pas de temps à utiliser doit être petit, ce qui problématique avec des méthodes hiérarchiques qui affinent le maillage.

Autre avantage de ces méthodes, elles sont aisément parallèlisables, et tirent donc facilement parti des architectures GPU de type CUDA comme en atteste l'article de Comas *et al.* [CTA+08]. Cette rapidité permet aussi d'utiliser des modèles déformables plus avancés, gérant l'anisotropie ou la visco-élasticité.

Les articles suivant utilisent des schémas d'intégration dérivés des schémas explicites : [MHTG05, THMG04, Mil88, NAM+06]

3.2.2 Les schémas de type implicite

Construction

Les schémas de type implicite semblent assez proches des schémas de type explicite, dans la mesure où l'on définit $\Delta\boldsymbol{x}$ et $\Delta\dot{\boldsymbol{x}}$ d'une façon relativement similaire dans le cas d'un d'Euler

implicite :

$$\begin{pmatrix} \Delta \boldsymbol{x} \\ \Delta \dot{\boldsymbol{x}} \end{pmatrix} = h \begin{pmatrix} \dot{\boldsymbol{x}}_{i-1} + \Delta \dot{\boldsymbol{x}} \\ \boldsymbol{M}^{-1}(\boldsymbol{F}_{ext} - \boldsymbol{F}_{int}(\boldsymbol{x}_{i-1} + \Delta \boldsymbol{x}, \dot{\boldsymbol{x}}_{i-1} + \Delta \dot{\boldsymbol{x}})) \end{pmatrix} \tag{3.3}$$

La différence principale tient en ce que l'on prend en compte le pas courant pour calculer les position et vitesse au pas courant, ce qui nécessite la résolution d'un système.

En utilisant un développement de TAYLOR, il est possible de linéariser :

$$F_{int}(\boldsymbol{x}_{i-1} + \Delta \boldsymbol{x}, \dot{\boldsymbol{x}}_{i-1} + \Delta \dot{\boldsymbol{x}}) = \boldsymbol{F}_{int}(\boldsymbol{x}_{i-1}, \dot{\boldsymbol{x}}_{i-1}) + \frac{\partial \boldsymbol{F}_{int}}{\partial \boldsymbol{x}} \Delta \boldsymbol{x} + \frac{\partial \boldsymbol{F}_{int}}{\partial \dot{\boldsymbol{x}}} \Delta \dot{\boldsymbol{x}} \tag{3.4}$$

Ce qui permet d'écrire :

$$\Delta \dot{\boldsymbol{x}} = h \left[\boldsymbol{M}^{-1} \left(\boldsymbol{F}_{ext} - \boldsymbol{F}_{int}(\boldsymbol{x}_{i-1}, \dot{\boldsymbol{x}}_{i-1}) - \frac{\partial \boldsymbol{F}_{int}}{\partial \boldsymbol{x}} \Delta \boldsymbol{x} - \frac{\partial \boldsymbol{F}_{int}}{\partial \dot{\boldsymbol{x}}} \Delta \dot{\boldsymbol{x}} \right) \right] \tag{3.5}$$

$$= h \left[\boldsymbol{M}^{-1} \left(\boldsymbol{F}_{ext} - \boldsymbol{F}_{int}(\boldsymbol{x}_{i-1}, \dot{\boldsymbol{x}}_{i-1}) - \frac{\partial \boldsymbol{F}_{int}}{\partial \boldsymbol{x}} h(\dot{\boldsymbol{x}}_{i-1} + \Delta \dot{\boldsymbol{x}}) - \frac{\partial \boldsymbol{F}_{int}}{\partial \dot{\boldsymbol{x}}} \Delta \dot{\boldsymbol{x}} \right) \right] \tag{3.6}$$

dont on tire :

$$(\boldsymbol{I} + h\boldsymbol{M}^{-1} \frac{\partial \boldsymbol{F}_{int}}{\partial \dot{\boldsymbol{x}}} + h^2 \boldsymbol{M}^{-1} \frac{\partial \boldsymbol{F}_{int}}{\partial \boldsymbol{x}}) \Delta \dot{\boldsymbol{x}} = h \left[\boldsymbol{M}^{-1} \left(\boldsymbol{F}_{ext} - \boldsymbol{F}_{int}(\boldsymbol{x}_{i-1}, \dot{\boldsymbol{x}}_{i-1}) - \frac{\partial \boldsymbol{F}_{int}}{\partial \boldsymbol{x}} h \dot{\boldsymbol{x}}_{i-1} \right) \right] \tag{3.7}$$

La résolution de ce système permet donc de calculer $\Delta \dot{\boldsymbol{x}}$ dont on peut ensuite déduire $\Delta \boldsymbol{x}$ et par conséquent $\boldsymbol{x}_i$.

Propriétés

La différence majeure entre les schémas explicites et implicites est l'utilisation par les schémas implicites des dérivées $\frac{\partial \boldsymbol{F}_{int}}{\partial \boldsymbol{x}}$ et $\frac{\partial \boldsymbol{F}_{int}}{\partial \dot{\boldsymbol{x}}}$ qui permettent d'assurer une certaine cohérence à l'intégration. Notamment, dans le cas d'un schéma d'EULER implicite, il est garanti que partant de $(\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i)$, et en utilisant un schéma EULER explicite avec pour correction $-h(\dot{\boldsymbol{x}}_i, \boldsymbol{M}^{-1}\boldsymbol{F}_{int}(\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i))$, c'est à dire en revenant en arrière, on retombe bien sur $(\boldsymbol{x}_{i-1}, \dot{\boldsymbol{x}}_{i-1})$; ce que les schémas explicites ne garantissent pas [BW98].

Cela confère à ce type de schéma d'intégration une stabilité qui est, selon [NAM+06], inconditionnelle. Contrairement aux schémas explicites, ils permettent de simuler des systèmes ayant un mauvais conditionnement[1]. Par contre, ils nécessitent de résoudre un système non linéaire. Ce dernier peut être linéarisé et résolu à l'aide d'un NEWTON-RAPHSON, ce qui est coûteux. Pour de petits systèmes la résolution peut se faire de manière directe, tandis que pour de grands systèmes la résolution se fait généralement avec un GRADIENT CONJUGUÉ pour peu que le système soit défini positif.

En raisons de ces qualités, de nombreux articles [BW98, HLB+06, CMT02, MLM06, TT94, TBHF03, GW06] utilisent des schémas implicites.

[1] On parle de système *stiff*, ou *raides*.

3.3 Les solveurs en ligne

Une fois le schéma d'intégration déterminé, si ce dernier est implicite, ou s'il est explicite avec une matrice de masse non diagonale, il faut choisir un solveur.

Le choix de ce solveur est particulièrement important, car il va conditionner la fréquence de la simulation ainsi que sa précision. De nombreux paramètres sont alors à prendre en compte, tels que le modèle utilisé, sa complexité, le format de stockage des données, *et cætera*.

Nous décrivons donc dans la suite les grands types de solveurs souvent rencontrés en insistant sur le cadre de leur utilisation. Plus de détails à leur sujet peuvent être trouvés dans [GvL96].

3.3.1 Résolution par solveurs directs

Les solveurs directs cherchent à résoudre le système $\boldsymbol{Ax} = \boldsymbol{b}$ en une seule passe. Ils reposent le plus souvent sur la décomposition de la matrice $\boldsymbol{A}$ en un produit de matrices plus simples. C'est le cas de la factorisation LU qui décompose $\boldsymbol{A}$ comme suit :

$$\boldsymbol{PAQ} = \boldsymbol{LU} \tag{3.8}$$

où $\boldsymbol{L}$ est une matrice triangulaire inférieure, et où $\boldsymbol{U}$ est une matrice triangulaire supérieure. Les matrices $\boldsymbol{P}$ et $\boldsymbol{Q}$ sont simplement des matrices de permutations qui évitent des divisions par zéro ou des chiffres trop petits. Une fois la factorisation faite, il suffit de résoudre :

1. $\boldsymbol{Lx}_0 = \boldsymbol{Pb}$
2. $\boldsymbol{Ux}_1 = \boldsymbol{x}_0$
3. $\boldsymbol{x} = \boldsymbol{Qx}_1$

$\boldsymbol{L}$ et $\boldsymbol{U}$ étant triangulaires, les étapes 1 et 2 se font rapidement. L'étape 3 est encore plus rapide car il s'agit d'un simple permutation.

Dans la pratique, ce type de solveur n'est pas très employé car la matrice $\boldsymbol{A}$ peut être amenée à changer au cours de la simulation. Sa re-factorisation est trop coûteuse pour être faite en temps réel. De plus ces méthodes ne tirent pas partie de la structure creuse des matrices issues de modèle Élements Finis.

Dans des cas précis, comme pour des objets déformables présentant une structure linéique, avec une bonne numérotation des nœuds il est possible d'obtenir des matrices tridiagonales qui peuvent bénéficier d'une résolution immédiate. Il existe aussi des bibliothèques de calcul, comme superLU, qui essaient de résoudre de manière directe de gros systèmes. Au Labri, Hénon propose aussi des méthodes pour parallèliser ces problèmes.

3.3.2 Résolution par solveurs itératifs

Contrairement aux solveurs directs, les solveurs itératifs cherchent à résoudre progressivement l'équation $\boldsymbol{Ax} = \boldsymbol{b}$, en proposant à chaque itération une correction approchant la solution exacte. Ce principe même, en plus de leurs performances, rendent ces solveurs particulièrement attrayant pour les simulations interactives car ils apportent une grande souplesse. Ils permettent d'établir un compromis entre précision du résultat et temps de calcul nécessaire en jouant simplement sur le nombre d'itérations. À cela il faut encore ajouter l'avantage que tirent ces méthodes des matrices creuses.

Le Gradient Conjugué

Formulation quadratique Soit $\boldsymbol{A}$ une matrice définie positive symétrique. Plutôt que de résoudre l'équation linéaire $\Phi'(\boldsymbol{x}) = \boldsymbol{Ax} - \boldsymbol{b} = \boldsymbol{0}$, nous cherchons à minimiser la fonction quadratique $\Phi(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Ax} - \boldsymbol{x}^T\boldsymbol{b}$, ce qui est équivalent et correspond à la minimisation de l'énergie du système.

Pour minimiser, il est possible dans un premier temps de se déplacer dans la direction inverse du gradient. Or, ce gradient, $\bigtriangledown\Phi$ est égal à $\bigtriangledown\Phi = \boldsymbol{Ax} - \boldsymbol{b}$. C'est donc le résidu $\boldsymbol{r}$ de $\boldsymbol{Ax} = \boldsymbol{b}$.

Maintenant que la direction dans laquelle se déplacer est connue, il faut déterminer de combien se déplacer. Pour cela, il faut trouver le scalaire $\alpha > 0$ qui minimise $\Phi(\boldsymbol{x} + \alpha\boldsymbol{r})$. Par le calcul il est prouvé que :

$$\alpha = \frac{\boldsymbol{r}^T\boldsymbol{r}}{\boldsymbol{r}^T\boldsymbol{Ar}} \tag{3.9}$$

permet la plus grande minimisation.

Partant d'une valeur initiale $\boldsymbol{x}_0$, Φ peut donc être minimisée, et $\Phi'(\boldsymbol{x}) = 0$ résolue en suivant l'algorithme 3.

Algorithm 3 STEEPEST DESCENT

1: **while** $\boldsymbol{r}_k \neq 0$ **do**
2: $k = k + 1$
3: $\alpha_k = \frac{\boldsymbol{r}_{k-1}^T\boldsymbol{r}_{k-1}}{\boldsymbol{r}_{k-1}^T\boldsymbol{Ar}_{k-1}}$
4: $\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \alpha_k\boldsymbol{r}_{k-1}$
5: $r_k = \boldsymbol{b} - \boldsymbol{Ax}_k$
6: **end while**

Cette méthode appelée la STEEPEST DESCENT, ou descente du gradient, souffre cependant d'un taux de convergence extrêmement lent si la matrice $\boldsymbol{A}$ est mal conditionnée.

Gradient Conjugué Pour améliorer la vitesse de convergence de la STEEPEST DESCENT, il a été proposé de se déplacer non plus suivant le gradient de Φ, mais suivant les directions successives $\{\boldsymbol{p}_1, \cdots, \boldsymbol{p}_n\}$. Ces $\boldsymbol{p}_k$ sont évidemment choisis pour assurer à chaque fois la plus grande minimisation. Le coefficient α se calcule alors comme suit :

$$\alpha_k = \frac{\boldsymbol{r}_{k-1}^T\boldsymbol{r}_{k-1}}{\boldsymbol{p}_k^T\boldsymbol{Ap}_k} \tag{3.10}$$

Les $\boldsymbol{p}_k$ sont donnés par :

$$\boldsymbol{p}_0 = \boldsymbol{r}_0 \tag{3.11}$$

$$\beta_k = -\frac{\boldsymbol{p}_{k-1}^T\boldsymbol{Ar}_{k-1}}{\boldsymbol{p}_{k-1}^T\boldsymbol{Ap}_{k-1}} \tag{3.12}$$

$$\boldsymbol{p}_k = \boldsymbol{r}_{k-1} + \beta_k\boldsymbol{p}_{k-1} \tag{3.13}$$

L'algorithme 4 obtenu pour résoudre $\boldsymbol{Ax} - \boldsymbol{b} = 0$ est alors très similaire à l'algorithme 3 :

Algorithm 4 GRADIENT CONJUGÉ

while $\boldsymbol{r}_k \neq 0$ **do**
  $k = k + 1$
  if $k = 1$ **then**
    $\boldsymbol{p}_1 = \boldsymbol{r}_0$
  else
    $\alpha_k = \frac{\boldsymbol{r}_{k-1}^T \boldsymbol{r}_{k-1}}{\boldsymbol{r}_{k-1}^T \boldsymbol{A} \boldsymbol{r}_{k-1}}$
  end if
  $\alpha_k = \frac{\boldsymbol{r}_{k-1}^T \boldsymbol{r}_{k-1}}{\boldsymbol{p}_{k-1}^T \boldsymbol{A} \boldsymbol{p}_{k-1}}$
  $\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \alpha_k \boldsymbol{r}_{k-1}$
  $\boldsymbol{r}_k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k$
end while

Pre-conditionnement Malgré cette amélioration, la méthode du GRADIENT CONJUGUÉ voit toujours sa vitesse de convergence dépendre du conditionnement de la matrice $\boldsymbol{A}$. C'est pourquoi il a été proposé de résoudre à la place du système $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ le système :

$$\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{P}^{-1}\boldsymbol{b} \tag{3.14}$$

où la matrice $\boldsymbol{P}^{-1}$ doit être proche de $\boldsymbol{A}^{-1}$ de telle sorte que le produit $\boldsymbol{P}^{-1}\boldsymbol{A}$ soit proche de l'identité, ce qui permet d'améliorer le conditionnement de la matrice[2]. Par contre, le surcoût de l'utilisation de la matrice $\boldsymbol{P}^{-1}$ doit impacter au minimum le gain dû au meilleur conditionnement.

Le choix d'un bon pré-condtionnneur est donc délicat. Le SYMMETRIC GAUSS-SEIDEL semble être un bon choix, car il tire partie de la symétrie de la matrice A et est construit de telle manière qu'il ne requière pas de stocker d'information complémentaire. Ce dernier se contruit comme suit :

$$\boldsymbol{P} = (\boldsymbol{D} - \boldsymbol{L})\boldsymbol{D}^{-1}(\boldsymbol{D} - \boldsymbol{L})^T \tag{3.15}$$

avec $\boldsymbol{L}$ la partie strictement inférieure de $\boldsymbol{A}$, et $\boldsymbol{D}$ sa partie diagonale. Les calculs nécessaires à son application peuvent être effectués rapidement car il n'est pas nécessaire de l'inverser. Il est plus judicieux de chercher $\boldsymbol{y}$ tel que $\boldsymbol{P}\boldsymbol{y} = \boldsymbol{c}$ plutôt que de calculer $\boldsymbol{y} = \boldsymbol{P}^{-1}\boldsymbol{c}$. Les étapes nécessaires sont les suivantes :

1. On cherche $\boldsymbol{y}_0$ tel que $(\boldsymbol{D} - \boldsymbol{L})\boldsymbol{y}_0 = b$
2. Puis $\boldsymbol{y}_1$ tel que $\boldsymbol{D}^{-1}\boldsymbol{y}_1 = \boldsymbol{y}_0$
3. Et enfin $\boldsymbol{y}$ tel que $(\boldsymbol{D} - \boldsymbol{L})^T\boldsymbol{y} = \boldsymbol{y}_1$

L'étape 2 est résolue très rapidement en multipliant par les valeurs diagonales de $\boldsymbol{A}$. Les étapes 1 et 3, quand à elles, peuvent aussi être résolues rapidement puisque $(\boldsymbol{D} - \boldsymbol{L})$ est triangulaire.

Les pré-conditionneurs permettent donc d'accélérer les calculs, du moins tant qu'ils sont implémentés sur CPU. Leur déploiement sur GPU n'est par contre généralement pas trivial du fait même de leur structure. Nous verrons dans la chapitre 4 d'autres exemples de pré-conditionneur.

Implémentation sur GPU du Gradient Conjugué Dans la cas où la matrice $\boldsymbol{A}$ a été assemblée à partir de matrices élémentaires issue de la méthode des ÉLEMENT FINIS, le GRADIENT CONJUGUÉ peut-être facilement implémenté sur GPU. Les opérations les plus coûteuses

[2] Le conditionnement de la matrice identité est 1.0.

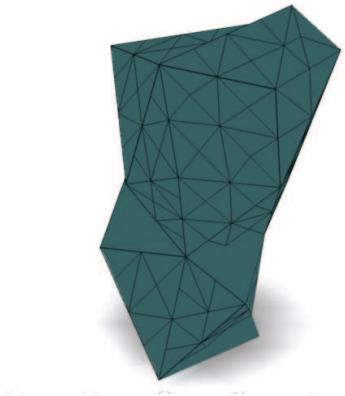


FIG. 3.1 – Discrétisation spatiale grossière.

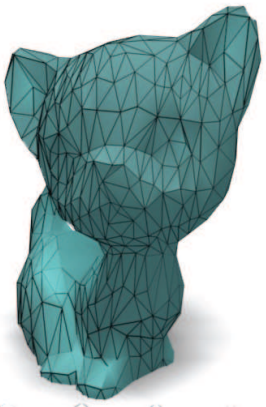


FIG. 3.2 – Discrétisation spatiale fine.

sont celles impliquant les produits matrice-vecteur. Or, lorsque la matrice $\boldsymbol{A}$ est assemblée à l'aide de matrices élémentaires, il est possible de construire le produit $\boldsymbol{Ax}$ élément e par élément comme suit :

1. Pour chaque matrice élémentaire $\boldsymbol{A}_e$, on calcule en parallèle le produit $\boldsymbol{b}_e = \boldsymbol{A}_e\boldsymbol{x}_e$, où $\boldsymbol{x}_e$ contient les $\boldsymbol{x}_i$ associés avec les nœuds de l'élément.
2. Le vecteur $\boldsymbol{x}$ est alors construit en accumulant les différents $\boldsymbol{x}_e$.

C'est la méthode implémentée par JÉRÉMIE ALLARD dans SOFA[3], et qui permet de simuler par exemple des maillages de 22000 tétraèdres à 40 fps avec une carte Nvidia GeForce 8800.

3.3.3 Résolution par solveurs multigrilles

Les solveurs hiérarchiques sont une autre classe de solveurs itératifs bien adaptée à la résolution de problème ÉLEMENT FINIS. Ils sont basés sur des algorithmes de relaxation comme GAUSS-SEIDEL ou JACOBI et l'utilisation de différentes résolutions. Leur développement tire son origine des observations suivantes :

- Les solveurs de type GAUSS-SEIDEL ou JACOBI sont efficaces mais convergent lentement,
- Sauf si la solution initiale est correcte.
- Ils ont un meilleur taux de convergence lorsqu'ils sont appliqués sur des données peu lisses.

Ces observations ont mené aux idées suivantes :

- Lorsque l'on utilise des modèles basés sur une discrétisation, une résolution grossière permet de donner une idée approximative de la solution à une résolution plus fine. Voir la figure 3.3. Pourquoi ne pas utiliser cette solution grossière comme solution initiale ?
- Les résidus d'une équation sont généralement non lisses. Pourquoi ne pas appliquer ces schémas de relaxation sur les résidus plutôt que sur l'inconnue du système ?

Pour mettre en application ces idées, il faut dans un premier temps définir un moyen permettant de passer d'une résolution à une autre. C'est le rôle des opérateurs de restriction $\boldsymbol{R}$ et de prolongation $\boldsymbol{P}$. Ainsi, si $\boldsymbol{x}_i$ est une inconnue définie à la résolution fine i, alors il est possible de projeter cette inconnue à la résolution $i-1$ comme suit :

$$\tilde{\boldsymbol{x}}_{i-1} = \boldsymbol{R}\boldsymbol{x}_i \tag{3.16}$$

[3] www.sofa-framework.org

De manière similaire, si $\boldsymbol{x}_{i-1}$ est une inconnue définie à une résolution $i-1$, alors elle s'interpole à la résolution i comme suit :

$$\tilde{\boldsymbol{x}}_i = \boldsymbol{P}\boldsymbol{x}_{i-1} \tag{3.17}$$

Ensuite, pour tirer partie de la propension des schémas de relaxation à mieux réduire l'erreur sur les hautes fréquences, la relaxation est appliquée sur les résidus en suivant :

1. Calcul du résidu : $r_i = A_i x_i^0 - b_i$
2. Transfert du résidu à la résolution grossière : $r_{i-1} = Rr_i$.
3. Recherche de v_{i-1} tel que $A_{i-1}v_{i-1} = r_{i-1}$
4. Transfert de la correction à la résolution fine : $v_i = Pv_{i-1}$.
5. Application de la correction : $x_i = x_i^0 + v_i$. (x_i est non-lisse puisque le résidu r_i ne l'était pas)
6. Relaxation de $A_i x_i = b_i$ avec α Gauss-Seidel itérations.

Le cycle en V bien connu peut ensuite en être appliqué comme montré dans l'algorithme 5.

Algorithm 5 le cycle en V

Require: A_i, b_i, x_i^0, R, P
Ensure: x^i tel que $A_i x^i = b_i$
if Si la résolution est la plus grossière **then**
  Résoudre $A_i x_i = b_i$. //On est à la résolution la plus grossière. La solution est calculée avec un solveur direct.
else
  Calculer la matrice grossière $A_{i-1} = RA_iP$.
  for j in $[0, maxIter]$ **do**
    Relaxer α_1 fois $A_i x_i^0 = b_i$ avec un Gauss Seidel. //Début du cycle en V
    Calculer le résidu : $r_i = A_i x_i^0 - b_i$
    Transférer le résidu à la résolution grossière : $r_{i-1} = Rr_i$.
    Résoudre Récursivement $A_{i-1}v_{i-1} = r_{i-1}$ avec le cycle en V. //Point du cycle en V
    Mettre à jour x_i : $x_i = x_i + Pv_{i-1}$.
    Relaxer α_2 fois $A_i x_i = b_i$ avec α Gauss Seidel. //fin du cycle en V
  end for
end if

Il existe deux types de hiérarchies de grilles. Soit les grilles sont totalement indépendantes d'une résolution à une autre, soit il existe un lien topologique en elles. Dans le premier cas, il s'agit de hiérarchies de grilles quelconques, tandis que dans l'autre cas, il s'agit de hiérarchies de grilles imbriquées.

Multigrille Volumique Quelconque Dans le premier cas, lorsque des maillages tétraèdriques sont utilisés, comme c'est le cas dans la figure 3.3, l'opérateur de restriction est construit en utilisant une interpolation barycentrique [GW06]. La grille fine est plongée dans la grille grossière, et les coordonnées barycentriques des nœuds de la résolution fines sont calculées dans les tétraèdres qui les contiennent. Les coordonnées ainsi obtenues servent de poids pour l'interpolation et permettent de construire $\boldsymbol{R}$.

L'opérateur de prolongation est alors défini comme $\boldsymbol{P} = \boldsymbol{R}^T$.

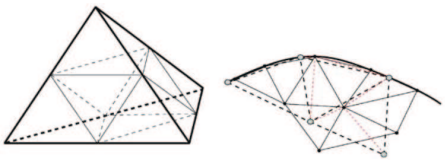


FIG. 3.3 – Interpolation Barycentrique. Issu de [GW06].

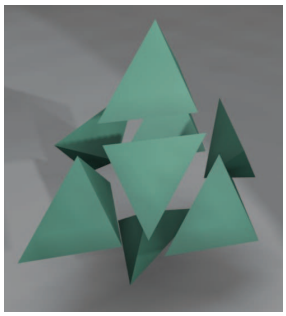


FIG. 3.4 – Schéma de subdivision tétraèdrique.

L'avantage de ce type de méthode est qu'il n'est pas nécessaire d'avoir de lien topologique entre les grilles aux différentes résolutions. Il n'y a donc pas de limitation à la simplification entre deux maillages successifs. Par contre, l'utilisation de l'interpolation barycentrique, nécessaire pour construire les matrices aux diverses résolutions, est coûteuse en temps de calcul.

Multigrille Volumique Imbriquée Dans ce cas, les nœuds d'un maillage à une résolution $i+1$ sont connectés à ceux d'une résolution i. Ces hiérarchies de maillages sont donc généralement créées avec des schémas de subdivision, comme celui de la figure 3.4, qui permettent d'assurer un lien topologique entre l'ancien et le nouveau maillage.

Ce genre de hiérarchie permet une construction rapide des matrices $\boldsymbol{A}_i$ aux différentes résolutions. Par contre, l'utilisation de schémas de subdivision restreint leur utilisation à des géométries sans trop de détails ou ayant des frontières lisses. Par exemple, si le schéma de subdivision de la figure 3.4 est utilisé, malgré l'augmentation du nombre de tétraèdres, l'objet maillé finement aura toujours la même surface que l'objet à la résolution grossière.

À ces deux limitations, la lenteur de construction des matrices $\boldsymbol{A}_i$ pour les hiérarchies quelconques, et l'impossibilité d'approximer des surfaces complexes avec des hiérarchies imbriquées, nous avons obvié à l'aide des méthodes qui seront présentées dans la section 3.4.

GPU Jusqu'à présent, les solveurs multigrilles ont rarement fait l'objet d'un port sur GPU en raison de leur structure qui les rend peu adaptés à une approche parallèle. En particulier, les schémas de relaxation de type GAUSS-SEIDEL se parallélisent mal.

De nombreux papiers utilisent des solveurs multigrilles. On peut citer notamment [SYBF06, GW06].

3.3.4 Résolution par solveurs adaptatifs

Les solveurs hiérarchiques que nous venons de présenter utilisent des résolutions grossières pour accélérer la résolution d'un problème à une résolution fine, avec pour objectif un gain de temps substantiel.

Les solveurs adaptatifs prennent en quelque sorte le parti inverse. Ils cherchent à résoudre un problème à une résolution grossière, en désirant affiner localement la solution, et ce au prix d'un

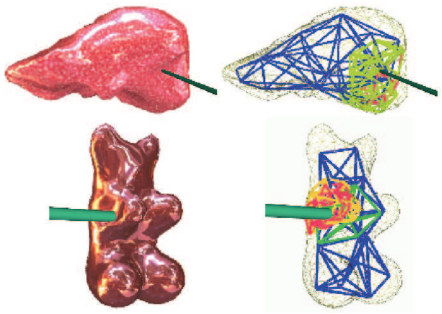


FIG. 3.5 – Adaptation locale du maillage. Issu de [DDCB01].

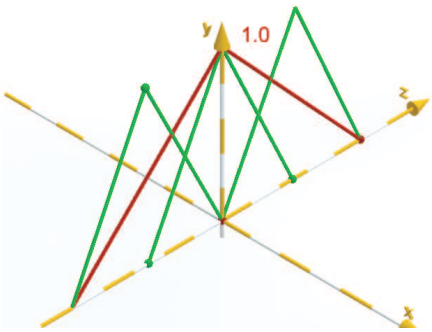


FIG. 3.6 – La fonction mère en rouge est remplacée par les fonctions filles en rouge.

léger surcoût. Pour cela, ils font appel à une partie des résolutions plus fines. Voir figure 3.5. Ce principe permet ainsi d'optimiser au mieux le maillage utilisé pour une simulation.

Cet objectif peut être atteint simplement en construisant, dans le cadre des ÉLEMENTS FINIS, un maillage qui prenne en compte les spécificités d'une simulation. Par exemple, la surface de l'objet, ou une partie étant particulièrement déformée, peut faire l'objet d'une discrétisation plus fine tandis que l'intérieur de l'objet est discrétisé plus grossièrement.

Cependant cette méthode n'est pas dynamique dans le sens où la discrétisation de l'objet est fixée pour toute la simulation, or il est envisageable que les zones d'intérêt, nécessitant une résolution plus fine, changent. Nous allons donc voir comment les équations pour ces maillages adaptés dynamiquement peuvent être construites.

Adaptation du problème

De la même manière que dans le cas des solveurs hiérarchiques il fallait définir des opérateurs $\boldsymbol{P}$ et $\boldsymbol{R}$ pour passer d'une résolution à l'autre, dans le cas des solveurs adaptatifs, il va falloir définir localement un moyen pour construire les équations associées à un maillage adapté.

DEBUNNE *et al.* proposent dans [DDCB01] d'utiliser une interpolation barycentrique pour affiner localement les équations. L'utilisation d'une interpolation barycentrique permet d'utiliser des maillages sans lien d'une résolution à l'autre. Par contre, comme il est noté dans [GKS02], aucune étude mathématique de cette méthode n'a été faite.

GRINSPUN présente pour sa part dans [GKS02] une méthode d'affinage basée sur un schéma de construction de fonctions d'interpolation très proche des théories ondelettes. L'affinage de la solution se fait en remplaçant une fonction d'interpolation par un ensemble de fonctions filles dont la somme pondérée égale la fonction mère. Voir figure 3.6.

Détermination des zones à adapter

Une fois connue la manière selon laquelle sont affinés localement les maillages, if faut déterminer où les affiner. Cela se fait à l'aide d'un critère qui essaie d'estimer l'erreur locale due à l'utilisation d'une résolution particulière. Plusieurs critères différents peuvent être trouvés dans [DDCB01, GKS02]. Si ce critère dépasse un certain seuil, le maillage est affiné localement à l'aide

d'une résolution plus fine. Inversement, si le critère indique une très faible erreur, il est possible de passer à une résolution plus faible pour accélérer les calculs.

3.4 Mailleur hiérarchique et Construction rapide des matrices $\boldsymbol{A}_i$

Lorsque l'on simule des objets déformables avec un schéma d'intégration temporel de type implicite et une méthode Élements Finis, un des goulets d'étranglement est la résolution de l'équation 3.7. Celle-ci nécessite en effet la résolution d'un système de dimension $3n$ où n est le nombre de nœuds du maillage tétraèdrique.

Au regard du rapide aperçu des différents solveurs que nous venons de présenter, il semble que les solveurs hiérarchiques soit une bonne solution pour rapidement résoudre cette équation. Les éléments finis se prêtent bien à une approche multirésolution. Parler de multigrille dans ce contexte est naturel et aisément appréhendable. Cependant, comme nous l'avons détaillé dans le sous-section 3.3.3, l'utilisation de hiérarchies de grilles quelconques ou imbriquées posent toutes deux des problèmes. La première ne permet pas un construction rapide des matrices $\boldsymbol{A}_i$ nécessaires à chaque niveau de résolution i, tandis que la seconde ne permet pas d'approximer des géométries complexes.

Pour pallier à ces difficultés, nous avons proposé respectivement de tirer partie de la structure creuse des matrices $\boldsymbol{A}_i$, et d'utiliser un mailleur hiérarchique qui permette d'obtenir une hiérarchie de grilles imbriquées approximant une surface complexe.

3.4.1 Construction rapide des matrices $\boldsymbol{A}_i$

Ces matrices sont construites comme suit :

$$\boldsymbol{A}_{i-1} = \boldsymbol{R}\boldsymbol{A}_i\boldsymbol{P} \tag{3.18}$$

Pour accélérer la construction des matrices $\boldsymbol{A}_i$, nous tirons parti de leur structure creuse. Pour cela, nous utilisons le format de stockage bien connu Compressed Sparse Row, CSR.

Les formats de stockage CSR C'est un format très répandu et utilisé dans de nombreuses librairies de calcul matriciel comme la MKL ou Sparse Blas. Il utilise trois tableaux pour stocker les coefficients de la matrice :

1. Value : ce tableau contient les valeurs non nulles de la matrice stockées ligne par ligne. Sa dimension est égale au nombre de coefficients non nuls.
2. Columns : ce tableau contient pour chaque valeur du tableau Value l'indice de la colonne où est stockée cette valeur. Il est de même dimension que le tableau Value.
3. RowIdx : ce tableau contient pour chaque ligne l'indice auquel cette ligne commence à être stockée dans Value et Columns. Ce tableau a pour dimension le nombre de lignes $+1$ car il est nécessaire de savoir combien de valeur sont stockées, or cette valeur est stockée à la fin de ce tableau.

L'avantage de ce stockage, outre le gain de mémoire, réside dans la rapidité du produit matrice creuse - vecteur plein. En effet, en parcourant simultanément les tableaux Value et Columns de RowIdx$[i]$ à RowIdx$[i+1]$, la colonne et la valeur de la ligne i de la matrice à multiplier par l'élément du vecteur plein idoine sont immédiatement connues.

$$\boldsymbol{C} = \begin{pmatrix} 0 & {}^{0}1 & 0 & {}^{1}7 & 0 \\ 0 & 0 & {}^{2}4 & 0 & 0 \\ 0 & {}^{3}2 & 0 & {}^{4}3 & 0 \\ {}^{5}6 & 0 & 0 & 0 & {}^{6}4 \end{pmatrix} \begin{pmatrix} 0 & 0 & {}^{0}12 & {}^{1}3 \\ 0 & {}^{2}5 & 0 & 0 \\ {}^{3}1 & {}^{4}7 & 0 & 0 \\ 0 & 0 & {}^{5}1 & 0 \\ 0 & {}^{6}4 & 0 & {}^{7}7 \end{pmatrix}.$$

FIG. 3.7 – La construction de l'*intersection* pour les valeurs sur la diagonale c_{22}, c_{33}, c_{44}

.

Les trois matrices $\boldsymbol{R}$, $\boldsymbol{A}_i$, et $\boldsymbol{P}$, étant creuses, sont stockées à l'aide ce format. Le problème, c'est que le produit de matrices creuses est assez inefficace. C'est pourquoi nous avons introduit dans [SDG07] la méthode suivante.

Produit rapide entre matrices creuses L'idée derrière cette méthode est assez simple. Le profil de ces trois matrices est connu à l'avance, donc lors du produit ce sont toujours des coefficients aux mêmes positions qui sont utilisés. Il est donc possible de créer une structure de données stockant en quelque sorte les *intersections* entre le lignes et colonnes des matrices du produit, et permettant d'accélérer le calcul.

Prenons le produit de matrice suivant :

$$\boldsymbol{C} = \boldsymbol{AB} = \begin{pmatrix} 0 & 1 & 0 & 7 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 2 & 0 & 3 & 0 \\ 6 & 0 & 0 & 0 & 4 \end{pmatrix} \begin{pmatrix} 0 & 0 & 12 & 3 \\ 0 & 5 & 0 & 0 \\ 1 & 7 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 4 & 0 & 7 \end{pmatrix} \tag{3.19}$$

Ces deux matrices sont stockées sous forme CSR :
- $\text{VALUE}_1 = \{1\ 7\ 4\ 2\ 3\ 6\ 4\}$
- $\text{COLUMNS}_1 = \{1\ 3\ 2\ 1\ 3\ 0\ 4\}$
- $\text{ROWIDX}_1 = \{0\ 2\ 3\ 5\ 6\}$
- $\text{VALUE}_2 = \{12\ 3\ 5\ 1\ 7\ 1\ 4\ 7\}$
- $\text{COLUMNS}_2 = \{2\ 3\ 1\ 0\ 1\ 2\ 1\ 3\}$
- $\text{ROWIDX}_2 = \{0\ 2\ 3\ 5\ 6\ 7\}$

La structure stockant l'*intersection* de ces deux matrices serait alors :

$$\{c_{12} = \{(1,2)\}, c_{13} = \{(1,5)\}, \tag{3.20}$$
$$c_{21} = \{(2,3)\}, c_{22} = \{(2,4)\}, \tag{3.21}$$
$$c_{32} = \{(3,2)\}, c_{33} = \{(4,5)\}, \tag{3.22}$$
$$c_{42} = \{(6,6)\}, c_{43} = \{(6,0)\}, c_{44} = \{(5,1),(6,7)\}\} \tag{3.23}$$

Voir la figure 3.7.

Cette structure stocke pour chaque coefficient c_{ij} non nul une liste de couples. La première valeur de ces couples donne l'indice du coefficient dans la première matrice à multiplier par le coefficient de la seconde matrice dont l'indice est donné par la seconde valeur du couple.

Construction de l'*intersection* La construction de cette *intersection*, immédiate pour des matrices stockées sous forme pleine, l'est moins pour des formats de stockage CSR. En effet il faut parcourir la matrice de gauche $\boldsymbol{A}$ ligne par ligne tandis que la matrice de droite $\boldsymbol{B}$ doit être parcourue colonne par colonne. Or, avec le stockage CSR, si le parcours de la matrice de droite se fait dans le bon ordre, ce n'est pas le cas de la matrice de gauche, dont les valeurs sont balayées ligne par ligne. Pour effectuer le parcours dans le bon ordre, il faut transposer cette matrice de gauche, et ce rapidement si possible.

Heureusement, cette transposition peut être faite très efficacement dans le cas du stockage CSR.

Pour cela, il va falloir parcourir la structure CSR et stocker dans un tableau les indices globaux

$$toSort[i] = (\text{COLUMNS}_B[i] - 1) * nbRows_B + row$$

qui donneraient la position de l'élément $(row, \text{COLMUNS}_B[i])$ dans un tableau de dimension $nbRows_B * nbCols_B$ de la transposée de la matrice $\boldsymbol{B}$.

Lors de cette construction, les positions sont stockées dans le désordre, or compte tenu du sens de balayage du format CSR, i.e. ligne par ligne, ces positions devraient être croissantes. Nous trierons donc ensuite ce tableau.

Dans le même temps sera mémorisé le nouveau $\text{COLUMN}_{B^T}[i] = row$ qui stocke maintenant dans la forme transposée l'indice de ligne où est stockée la valeur $\text{VALUE}_B[i]$. Enfin, toujours dans la même étape, le nombre de valeurs non nulles stockées pour une colonne i est calculé. Ce nombre sera ensuite cumulé pour construire ROWINDEX_{B^T}. Ces calculs constituent les étapes 1 à 14 de l'algorithme 6.

Algorithm 6 Transposition de la matrice $\boldsymbol{B}$

Require: ROWINDEX_B, VALUES_B, COLUMNS_B
Ensure: Calcule ROWINDEX_{B^T}, COLUMNS_{B^T}, *perm*

1: $row = 1$
2: **for** $1 \geq i \geq$ nombre valeurs non nulles **do**
3: $toSort[i] = (\text{COLUMNS}_B[i] - 1) * nbRows + row$ //On stocke la position de chaque valeur de B^T
4: $\text{COLUMN}_{B^T}[i] = row$//On stocke l'indice de la colonne correspondante.
5: $\text{ROWINDEX}_{B^T}[\text{COLUMNS}[i]] + +$
6: **if** $i \geq \text{ROWINDEX}_B[row + 1]$ **then**
7: **repeat**
8: $row = row + 1$ //On passe a la ligne suivante.
9: **until** $row < nbRows_B$ and $\text{ROWINDEX}_B[row] = \text{ROWINDEX}_B[row+1]$//On fait ça jusqu'à avoir parcouru toute la matrice B.
10: **end if**
11: **end for**
12: **for** $2 \geq i \geq nbCols_B$ **do**
13: $\text{ROWINDEX}_{B^T}[i] += \text{ROWINDEX}_{B^T}[i-1]$ //On cumule le nombre de valeur par ligne de B^T pour construire ROWINDEX_{B^T}
14: **end for**
15: Trier par RADIXSORT *toSort* et construire *perm*.

Une fois le tableau *toSort* rempli, nous le trions à l'aide d'un algorithme de type *radixSort* qui profite de la nature entière des valeurs à trier. En même temps, nous construisons une table

de permutation *perm* qui nous permet de savoir où est stockée la donnée $\boldsymbol{B}^T_{ij}$ dans le tableaux VALUE_B de la matrice non transposée $\boldsymbol{B}$. Cela revient en fait à construire le tableau VALUE^T_B, sauf que pour des raisons d'optimisation il est préférable de construire *perm* et d'aller chercher les valeurs dans VALUE_B.

Nous avons alors les tableaux ROWINDEX_{B^T}, COLUMNS_{B^T} ainsi que la permutation *perm* qui donne $\text{VALUE}_{B^T}[i] = \text{VALUE}_B[perm[i]]$. Nous pouvons alors parcourir simultanément $\boldsymbol{A}$ et $\boldsymbol{B}$ pour déterminer leur intersection. Voir l'algorithme 7. Pour cela, nous parcourons en particulier COLUMNS_A et COLUMNS_{B^T} pour chaque ligne. S'ils ont des valeurs identiques pour une même ligne, alors il y a intersection. C'est ce que font les lignes 10 à 20 de l'algorithme 7. Ensuite, des lignes 23 à 29, nous construisons un tableau d'offset qui nous donne la position de la valeur c_{ij} dans la matrice pleine $\boldsymbol{C}$, ce qui permet de construire aussi, si besoin, cette matrice pleine rapidement.

Algorithm 7 Construction de l'intersection de $\boldsymbol{A}$ et $\boldsymbol{B}$ à l'aide de $\boldsymbol{A}^{csr}$ et $\boldsymbol{B}^{T,csr}$

Require: VALUES_A, ROWINDEX_A, COLUMNS_A, VALUES_B, ROWINDEX_B, COLUMNS_B
Ensure: Calcule l'*intersection* de $\boldsymbol{A}$ et $\boldsymbol{B}$.

1: Transpose $\boldsymbol{B}$
2: **for** $1 \geq r \geq nbRows_A$ **do**
3: $nbNonZeroRow = \text{ROWINDEX}_A[r+1] - \text{ROWINDEX}_A[r]$
4: $offset_A = \text{ROWINDEX}_A[r]$
5: **for** $1 \geq c \geq nbCols_B$ **do**
6: $offset_{B^T} = \text{ROWINDEX}_{B^T}[c]$
7: $nbNonZeroCol = \text{ROWINDEX}_{B^T}[c+1] - \text{ROWINDEX}_{B^T}[c]$
8: $itR = 0$
9: $itC = 0$
10: **while** $itC < nonZeroCol$ *and* $itR < nonZeroRow$ **do**
11: $col_A = \text{COLUMNS}[offset_A + itR]$
12: $col_{B^T} = \text{COLUMNS}_{B^T}[offset_{B^T} + itC]$
13: $doIntersect = !(col_A - col_{B^T})$ // *vrai* si il y a intersection
14: $isInfA = col_A < col_{B^T}$
15: **if** *doIntersect* **then**
16: Ajouter $([offset_A + itR, perm[offset_{B^T} + itC])$ à la liste des intersections.
17: **end if**
18: $itC += 1 * (doIntersect \text{ or } !isInf)$ // itC incrémenté si intersection ou si $col_A > col_{B^T}$
19: $itR += 1 * (doIntersect \text{ or } isInf)$ // itR incrémenté si intersection ou si $col_A < col_{B^T}$
20: **end while**
21: **end for**
22: **end for**
23: **for** $1 \geq r \geq nbRows_A$ **do**
24: **for** $1 \geq c \geq nbCols_B$ **do**
25: **if** il y a une intersection pour la cellule (r, c) **then**
26: Ajouter $offset = r + c * nbRows_A$ à la liste des offsets. $offset$ contient la position de C_{rc} dans la matrice C pleine.
27: **end if**
28: **end for**
29: **end for**
30:

Produit rapide Le produit se fait alors simplement en parcourant la liste des intersections, en additionnant pour chaque coefficient c_{ij} le produit des coefficients stockés aux indices spécifiés

par les couples. De cette manière la matrice $\boldsymbol{C}$ peut-être construite rapidement, comme le montre l'algorithme 8.

Algorithm 8 Construction de la matrice pleine $\boldsymbol{C}$.

Require: VALUES_A, VALUES_B
Ensure: Remplit VALUES_C
1: **for** $0 < i < nbNonZero_C$ **do**
2: **for** pour chaque intersection (a, b) de la valeur i **do**
3: $\boldsymbol{C}[offset_C[i]] += \text{VALUES}_A[a] * \text{VALUES}_B[b]$ //Grâce au tableau offset, les valeurs sont directement positionnées dans $\boldsymbol{C}$.
4: **end for**
5: **end for**

Il est envisageable d'améliorer encore cette méthode. En effet, si les matrices sont issues d'un assemblage d'éléments finis, la matrice résultant de ce produit est généralement assez creuse. Dans le cadre de notre exemple, $\boldsymbol{C}$ vaut :

$$\boldsymbol{C} = \begin{pmatrix} 0 & 5 & 7 & 0 \\ 4 & 28 & 0 & 0 \\ 0 & 10 & 3 & 0 \\ 0 & 16 & 72 & 46 \end{pmatrix} \tag{3.24}$$

et est donc presque à moitié vide.

De plus, quelques soient les valeurs dans les deux matrices du produit, la matrice $\boldsymbol{C}$ aura toujours le même profil, même s'il est possible qu'occasionnellement un zéro soit stocké. Il est donc intéressant de remplir directement cette structure creuse dans la même étape. Cela se fait facilement comme le montre l'algorithme 9, puisque le produit se fait dans le même ordre que le stockage des valeurs dans VALUES_C. La matrice $\boldsymbol{C}$ est donc obtenue directement dans sa forme CSR et en plus, l'affectation des valeurs se fait de façon contiguë en mémoire ce qui est optimal.

Algorithm 9 Construction de la matrice creuse $\boldsymbol{C}^{csr}$. Son profil doit déjà être connu.

Require: VALUES_A, VALUES_B
Ensure: Remplit VALUES_C
1: **for** $0 < i < nbNonZero_C$ **do**
2: **for** pour chaque intersection (a, b)de la valeur i **do**
3: $\text{VALUES}_C[i] += \text{VALUES}_A[a] * \text{VALUES}_B[b]$ // Dans le format CSR les valeurs sont stockées dans l'ordre du produit.
4: **end for**
5: **end for**

Ainsi, en utilisant cette méthode, il est possible de calculer rapidement le produit de deux matrices creuses, tout en obtenant aussi une matrice creuse, à condition bien sûr que les profils de ces matrices soient connus à l'avance, ce qui est le cas dans les solveurs hiérarchiques.

Application au calcul de $\boldsymbol{A}_{i-1} = \boldsymbol{R}\boldsymbol{A}_i\boldsymbol{P}$ Cette méthode peut évidemment être appliquée directement au calcul de $\boldsymbol{A}_{i-1}$. Pour cela, le calcul est décomposé en deux phases :

1. Phase de pré-calcul
   (a) La structure d'*intersection* pour le produit $\boldsymbol{A}_i\boldsymbol{P}$ est calculée. La matrice résultant de ce produit est elle aussi creuse. Le profil de cette matrice est calculée une fois pour toute.

(b) Cette structure est utilisée pour calculer la matrice intermédiaire $\boldsymbol{A}_{tmp} = \boldsymbol{A}_i\boldsymbol{P}$

(c) La structure d'*intersection* pour le produit $\boldsymbol{R}\boldsymbol{A}_{tmp}$ est calculée.

2. Phase en ligne

(a) La matrice creuse $\boldsymbol{A}_{tmp}$ est calculée en parcourant la première structure d'*intersection* et les matrices creuses $\boldsymbol{P}$ et $\boldsymbol{A}_i$

(b) La matrice creuse $\boldsymbol{A}_{i-1}$ est calculée en parcourant la seconde structure d'*intersection* et les matrices creuses $\boldsymbol{R}$ et $\boldsymbol{A}_{tmp}$

Performances Les performances obtenues par cette méthode sont données dans le tableau 3.1. Elles sont comparées à celles que l'on obtient en utilisant la librairie de calcul matriciel de INTEL, la MKL et celles qu'obtient GEORGII dans [GW06]. Bien que la MKL soit optimisée pour les opérations avec des matrices creuses et pleines, notre méthode est très nettement supérieure. Cela tient au fait que la MKL est incapable de multiplier des matrices creuses directement. Seul les produits matrice creuse - matrice pleine et matrice pleine - matrice pleine sont possibles.

#Tetra	#levels	MKL update (ms)	Georgii[GW06] update (ms)	optimized update (ms)
432	2	9		4
1148	2	151	~ 25	14
3456	2	464		37
7168	3	2377	~ 170	76

TAB. 3.1 – Benchmark - 2 Dual Core 2.0 Ghz

3.4.2 Mailleur hiérarchique

Comme précisé dans la sous-section 3.3.3, les solveurs hiérarchiques souffrent d'un autre problème lorsqu'ils sont utilisés avec des hiérarchies de maillages imbriqués. Ces hiérarchies sont créées en partant d'un maillage grossier qui est raffiné successivement en utilisant des schémas de subdivision. Le problème inhérent à cette méthode est qu'il n'y a aucune raison que la surface du maillage le plus fin, obtenu par les subdivisions successives, converge vers la surface de l'objet réel.

Du coup, les résultats obtenus sont valides pour un objet ayant pour frontière la surface du maillage le plus fin et non celle de l'objet simulé. Pour pallier à cet inconvénient, nous avons proposé dans [SDG07] l'idée de mailleur volumique hiérarchique. Le principe d'un tel mailleur consiste en la génération d'une hiérarchie de maillages entre lesquels il existe un lien topologique, et dont les surfaces convergent au fil des subdivisions vers une surface de référence.

Fonctionnement

Ce mailleur volumique utilise un schéma de subdivision du tétraèdre pour raffiner progressivement les maillages. Le but est de générer itérativement un maillage volumique dont la surface est toujours plus proche de la surface cible.

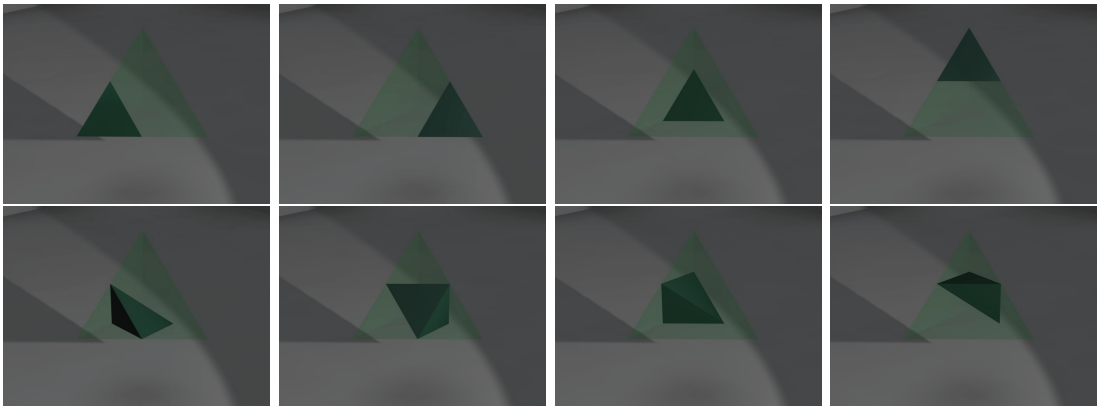


FIG. 3.8 – Schéma de subdivision du tétraèdre en 8.

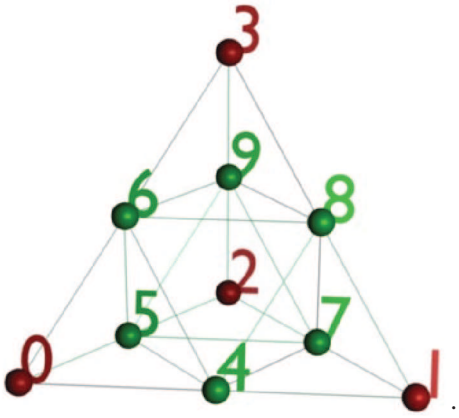


FIG. 3.9 – La numérotation des sommets du tétraèdre. Les nouveaux points sont en vert, les sommets sont en rouge.

Le point de départ est donc un maillage tétraèdrique grossier obtenu en maillant en tétraèdres une version décimée de la surface cible. La décimation a été faite dans notre cas avec le logiciel BLENDER[4]. Chaque tétraèdre est ensuite raffiné suivant le schéma de la figure 3.8, c'est à dire que chaque tétraèdre est découpé en huit sous-tétraèdres. Si les sommets du tétraèdre père, ainsi que les points ajoutés au centre de ses arêtes sont numérotés comme sur la figure 3.9, alors les sous-tétraèdres sont définis par les sommets suivants :

- $T_0 = \{0\ 4\ 5\ 6\}$
- $T_1 = \{1\ 8\ 7\ 4\}$
- $T_2 = \{2\ 7\ 9\ 5\}$
- $T_3 = \{3\ 6\ 9\ 8\}$
- $T_4 = \{4\ 5\ 6\ 7\}$
- $T_5 = \{4\ 6\ 7\ 8\}$
- $T_6 = \{5\ 6\ 7\ 9\}$
- $T_7 = \{6\ 7\ 8\ 9\}$

A noter que selon SCHAEFER *et al.* dans [SHW04], ce schéma de subdivision peut introduire un biais dans les résultats obtenus car il a tendance à privilégier une direction. Ils proposent donc un schéma de subdivision plus compliqué pour éviter cet écueil.

Enfin, après l'étape d'affinage par subdivision, nous cherchons à approcher au mieux la surface cible. Pour cela, chacun des points du nouveau maillage est déplacé suivant une heuristique pour coller au mieux à cette surface cible.

Heuristique

L'heuristique employée est très simple. Pour chaque point de la surface du maillage volumique raffiné, les trois triangles de la surface cible les plus proches sont déterminés. Sur chacun de ces triangles, le point le plus proche du point à déplacer est déterminé. Le barycentre de ces trois points devient alors la nouvelle position du point à déplacer.

L'algorithme complet est détaillé dans l'algorithme 10, et les résultats obtenus sont présentés dans la figure 3.10.

3.4.3 Conclusion

Nous avons présenté dans cette section deux améliorations pour les solveurs multigrilles. Notre algorithme permet de construire rapidement les matrices du système aux diverses résolutions en tirant parti de leur structure creuse. Il serait intéressant de voir dans quel mesure cet algorithme peut être adapté pour une architecture massivement parallèle comme le framework CUDA de NVIDIA.

Nous avons aussi introduit l'idée de mailleur hiérarchique, qui permet au fur et à mesure de l'affinage du maillage tétraèdrique de tendre vers une surface cible complexe. Cette méthode est pour l'instant très simple. Nous détaillons une piste d'amélioration dans les perspetives 3.6.

[4] www.blender.org

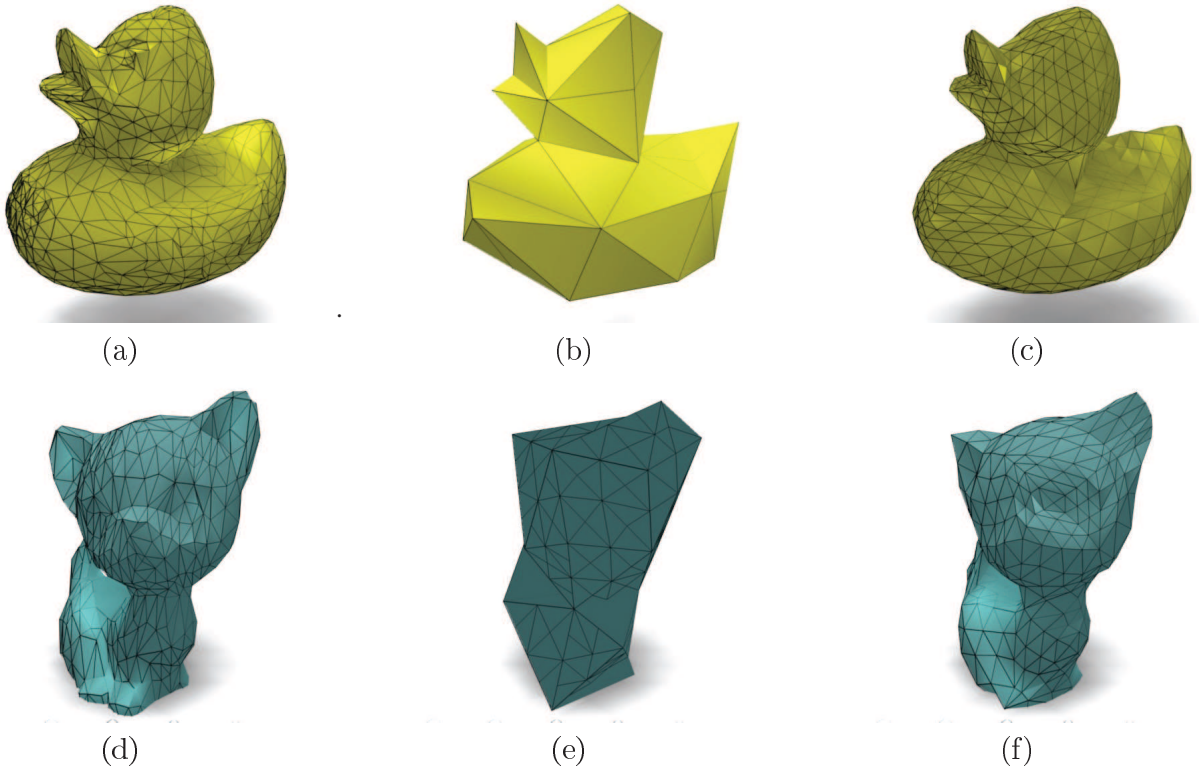


FIG. 3.10 – (a) : Maillage Initial ; (b) : Maillage Grossier ; (c) : Maillage Remaillé Hiérarchiquement (d) : Maillage Initial ; (e) : Maillage Grossier (f) : Maillage Remaillé Hiérarchiquement

Algorithm 10 Mailleur Hierarchique

Require: $\partial\Omega$, Ω_0, n_{level}
Ensure: $\partial\Omega_{n_{level}-1}$ soit une bonne approximation de $\partial\Omega$.

```
for i in [0, n_levels − 1] do
  for chaque tétraèdre T in Ω_i do
    Subdiviser T en T_0, ..., T_7.
    for chaque nouveaux nœuds n_new ajoutés do
      if deux nœuds parents sont sur ∂Ω_i then
        n_new est sur ∂Ω_{i+1}.
      end if
    end for
  end for
  créer Ω_{i+1} comme ⋃_{m=1}^{N_tet^{i+1}} T_m, où N_tet^{i+1} = 8 N_tet^i
  for chaque nœuds n_k dans ∂Ω_{i+1} do
    Trouver les 3 plus proches triangles t sur ∂Ω
    Trouver les points les plus proche de n_k sur chaque triangle.
    Déplacer n_k au baricentre de des trois points.
  end for
end for
```

3.5 Transformée en ondelettes volumique

Nous avons présenté dans la section 3.3 les solveurs multigrilles et adaptatifs. Ces solveurs utilisent différents niveaux de résolution pour accélérer la résolution de systèmes d'équations, ce qui appelle naturellement l'utilisation d'ondelettes.

Cependant, même si les ondelettes sont largement utilisées dans la communauté audio-visuelle, à travers des algorithmes de filtrage ou de compression, ou encore dans des modèles éléments finis 1D ou 2D[5], leur utilisation en 3D reste peu courante [BCCG06].

Dans les sections qui suivent nous présentons les ondelettes, et leur utilisation dans un contexte éléments finis volumiques.

3.5.1 Analyse Multirésolution

Une Analyse Multi-Résolution, AMR, de $\mathcal{L}_2(\mathbb{R})$, l'espace des fonctions de carré intégrable, est une suite d'espaces imbriqués V^i telle que l'union de ces V^i dans $\mathcal{L}_2(\mathbb{R})$ soit dense. Pour chaque V^i nous définissons un vecteur de fonctions de base $\mathbf{\Phi}^i(\boldsymbol{u})$:

$$\mathbf{\Phi}^i(\boldsymbol{u}) = \begin{bmatrix} \phi_1^i(\boldsymbol{u}) \\ \vdots \\ \phi_{n_i}^i(\boldsymbol{u}) \end{bmatrix} \tag{3.25}$$

où l'ensemble des $\{\phi_k^i\}_{k\in[|1,n_i|]}$ forme une base de V^i, et n_i est la dimension de V^i. Ces fonctions sont appelées *fonctions d'échelle*. Les espaces V_i étant imbriqués, il existe un matrice $\boldsymbol{H}^i$ telle que :

$$\mathbf{\Phi}^i = \boldsymbol{H}^i \mathbf{\Phi}^{i+1} \tag{3.26}$$

[5] Le format de compression d'image JPEG2000 utilise des ondelettes.

De ce fait, les *fonctions d'échelle* sont dites affinables. Il est possible de définir des espaces de détails W^i comment étant les compléments de V^i dans V^{i+1} :

$$V^i \oplus W^i = V^{i+1} \tag{3.27}$$

Ces espaces de détails W^i peuvent être vus comme des espaces contenant les informations nécessaires pour reconstruire une fonction f^{i+1} définie dans V^{i+1} à partir de sa projection f^i dans V^i. Pour chaque espace W^i, comme pour les espaces V^i, nous définissons un vecteur de fonctions de base $\boldsymbol{\Psi}^i$:

$$\boldsymbol{\Psi}^i(\boldsymbol{u}) = \begin{bmatrix} \psi^i_1(\boldsymbol{u}) \\ \vdots \\ \psi^i_{n_{i+1}-n_i}(\boldsymbol{u}) \end{bmatrix} \tag{3.28}$$

où l'ensemble des $\{\psi^i_k\}_{k\in[|1,n_{i+1}-n_i|]}$ est une base de W^i. Ces fonctions sont appelées *fonctions de détails*, ou *ondelettes*. Comme les W^i sont inclus dans les V^{i+1}, $\boldsymbol{\Psi}^i$ peut être construit en fonction de $\boldsymbol{\Phi}^{i+1}$ à l'aide de la matrice $\boldsymbol{G}^i$:

$$\boldsymbol{\Psi}^i = \boldsymbol{G}^i \boldsymbol{\Phi}^{i+1} \tag{3.29}$$

Lorsque les espaces W^i sont orthogonaux aux espaces V^i, l'Analyse Multi-Résolution est dite *orthogonale*. Dans ce cas, la meilleure approximationf^i d'une fonction $f \in \mathcal{L}_2(\mathbb{R})$ dans V^i est donnée par sa projection orthogonale dans V^i :

$$f^i = \sum_{k=1}^{k=n_i} < f, \phi^i_k > \phi^i_k \tag{3.30}$$

Cependant, la contrainte d'orthogonalité est très restrictive. La seule transformée connue qui soit symétrique, compacte, à valeur réelle et orthogonale est la transformée de Haar [JS94]. C'est pourquoi les ondelettes biorthogonales, moins restrictives, ont été introduites.

3.5.2 Ondelettes Biorthogonales

Dans le cas d'ondelettes biorthogonales, une *fonction d'échelle* duale $\tilde{\phi}^i_k$ est définie de telle façon que

$$< \phi^j_h, \tilde{\phi}^i_k >= \delta_{ij}\delta_{kh}$$

où le produit scalaire $<,>$ est :

$$< \phi^j_h, \tilde{\phi}^i_k >= \int \phi^j_h \tilde{\phi}^i_k$$

δ_{ij} est le symbole de Kronecker, qui vaut 1 si $i = j$, 0 sinon.

Des fonctions *ondelettes* duales sont définies de manière similaire. Ainsi, en plus de la première Analyse Multi-Résolution, une seconde est construite et nommée Analyse Multi-Résolution duale. Cette dernière est notée : $\tilde{V}^i$, $\tilde{W}^i$, $\tilde{\boldsymbol{\Phi}}^i$, $\tilde{\boldsymbol{\Psi}}^i$, $\tilde{\boldsymbol{H}}^i$ et $\tilde{\boldsymbol{G}}^i$.

Les quatre espaces V^i, W^i, $\tilde{V}^i$ et $\tilde{W}^i$ sont construits de telle sorte que $V^i \perp \tilde{W}^i$ et $\tilde{V}^i \perp W^i$. Par construction, les propriétés suivantes sont aussi valables :

$$\left\{ \begin{array}{l} \left[\begin{array}{c} \tilde{\boldsymbol{H}}^i \\ \tilde{\boldsymbol{G}}^i \end{array} \right] \left[\begin{array}{cc} \boldsymbol{H}^i_* & \boldsymbol{G}^i_* \end{array} \right] = \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right] \\ \left[\begin{array}{cc} \boldsymbol{H}^i_* & \boldsymbol{G}^i_* \end{array} \right] \left[\begin{array}{c} \tilde{\boldsymbol{H}}^i \\ \tilde{\boldsymbol{G}}^i \end{array} \right] = 1 \end{array} \right. \tag{3.31}$$

où le symbole $*$ dénote le transposée. D'où le fait que les filtres $\tilde{\boldsymbol{H}}^i$ et $\tilde{\boldsymbol{G}}^i$ permettent de calculer les fonctions d'échelles à la résolution $i+1$ comme suit :

$$\boldsymbol{\Phi}^{i+1} = \tilde{\boldsymbol{H}}^i_* \boldsymbol{\Phi}^i + \tilde{\boldsymbol{G}}^i_* \boldsymbol{\Psi}^i \tag{3.32}$$

Dans la pratique, ces outils sont utilisés pour projeter une fonction f dans V_i. Par exemple, si f est projetée dans V^{i-1} et W^{i-1}, alors les coefficients de la combinaison linéaire pour l'espace V^{i-1} sont donnés par $\boldsymbol{\lambda}^{i-1} = \tilde{\boldsymbol{H}}^{i-1} \boldsymbol{\lambda}_i$ et ceux dans W^{i-1} par $\boldsymbol{\gamma}^{i-1} = \tilde{\boldsymbol{G}}^{i-1} \boldsymbol{\lambda}_i$.

L'opération inverse, appelée reconstruction, est aussi possible :

$$\boldsymbol{\lambda}^i = \boldsymbol{H}^{i-1}_* \boldsymbol{\lambda}^{i-1} + \boldsymbol{G}^{i-1}_* \boldsymbol{\gamma}^{i-1} \tag{3.33}$$

De cette manière il est possible de décomposer itérativement un ensemble $\boldsymbol{\lambda}^n$ en un ensemble grossier $\boldsymbol{\lambda}^0$ et une suite d'ensembles de détails $\{\gamma^0, \cdots, \gamma^{n-1}\}$. C'est ce qu'on appelle la *transformée en ondelettes discrète*.

La difficulté avec les ondelettes, c'est le choix des *fonctions d'échelles* et de *détails*. Les propriétés de ces fonctions peuvent être diverses : continuité C^0, $C^1, \cdots$, conservation des moments, *et cætera*. C'est à l'utilisateur de choisir la propriété qu'il souhaite garantir.

Or il existe une alternative pour construire facilement des transformées en ondelettes discrètes en partant d'une transformée simple. C'est le LIFTING SCHEME, introduit par WIM SWELDENS dans [SS95, Swe98, FPS96].

Le théorème du LIFTING SCHEME affirme que si un ensemble de filtres $\{\boldsymbol{H}^i_{old}, \boldsymbol{G}^i_{old}, \tilde{\boldsymbol{H}}^i_{old}, \tilde{\boldsymbol{G}}^i_{old}\}$ satisfait l'équation 3.31, alors l'ensemble de nouveaux filtres $\{\boldsymbol{H}^i, \boldsymbol{G}^i, \tilde{\boldsymbol{H}}^i, \tilde{\boldsymbol{G}}^i\}$ construit selon l'équation 3.34 satisfait aussi cette équation.

$$\begin{array}{rcl} \boldsymbol{H}^i & = & \boldsymbol{H}^i_{old} \\ \tilde{\boldsymbol{H}}^i & = & \tilde{\boldsymbol{H}}^i_{old} + \boldsymbol{S}^i \tilde{\boldsymbol{G}}^i_{old} \\ \boldsymbol{G}^i & = & \boldsymbol{G}^i_{old} - \boldsymbol{S}^i_* \boldsymbol{H}^i_{old} \\ \tilde{\boldsymbol{G}}^i & = & \tilde{\boldsymbol{G}}^i_{old} \end{array} \tag{3.34}$$

où $\boldsymbol{S}^i$ est une matrice de dimension $n_i \times (n_{i-1} - n_i)$. Cette matrice $\boldsymbol{S}^i$ correspond en fait à une étape de lifting comme l'illustre la figure 3.11, et c'est elle qui ajoute des propriétés à la transformée en ondelette.

Si ces équations sont exprimées en terme de fonction de base, il apparaît :

$$\boldsymbol{\Psi}^i = \boldsymbol{\Psi}^i_{old} - \boldsymbol{S}^i_* \boldsymbol{\Phi}^i_{old} \tag{3.35}$$

Le LIFTING SCHEME ajoute donc des propriétés aux anciennes fonctions de base à l'aide de la matrice $\boldsymbol{S}^i$. Cette dernière doit donc être choisie avec discernement. L'impact de cette matrice sur les coordonnées s'écrit comme suit :

$$\begin{array}{rcl} \boldsymbol{\lambda}^i & = & \boldsymbol{\lambda}^i_{old} + \boldsymbol{S}^i\gamma^i_{old} \\ \boldsymbol{\gamma}^i & = & \boldsymbol{\gamma}^i_{old} \end{array} \tag{3.36}$$

Il existe un pendant dual au théorème du LIFTING SCHEME qui dit que pour un ensemble de filtres $\{\boldsymbol{H}^i_{old}, \boldsymbol{G}^i_{old}, \tilde{\boldsymbol{H}}^i_{old}, \tilde{\boldsymbol{G}}^i_{old}\}$ qui satisfait l'équation 3.31, alors l'ensemble de nouveaux filtres $\{\boldsymbol{H}^i, \boldsymbol{G}^i, \tilde{\boldsymbol{H}}^i, \tilde{\boldsymbol{G}}^i\}$ construit selon l'équation 3.37 satisfait aussi cette équation.

$$\begin{array}{rcl} \boldsymbol{H}^i & = & \boldsymbol{H}^i_{old} + \tilde{\boldsymbol{S}}^i\boldsymbol{G}^i_{old} \\ \tilde{\boldsymbol{H}}^i & = & \tilde{\boldsymbol{H}}^i_{old} \\ \boldsymbol{G}^i & = & \boldsymbol{G}^i_{old} \\ \tilde{\boldsymbol{G}}^i & = & \tilde{\boldsymbol{G}}^i_{old} - \tilde{\boldsymbol{S}}^i_*\tilde{\boldsymbol{H}}^i_{old} \end{array} \tag{3.37}$$

où $\boldsymbol{S}^i$ est une matrice de dimension $n_i \times (n_{i-1} - n_i)$ et :

$$\tilde{\boldsymbol{\Psi}}^i = \tilde{\boldsymbol{\Psi}}^i_{old} - \tilde{\boldsymbol{S}}^i_*\tilde{\boldsymbol{\Phi}}^i_{old} \tag{3.38}$$

L'impact de cette matrice sur les coordonnées s'écrit comme suit :

$$\begin{array}{rcl} \boldsymbol{\lambda}^i & = & \boldsymbol{\lambda}^i_{old} \\ \boldsymbol{\gamma}^i & = & \boldsymbol{\gamma}^i_{old} + \tilde{\boldsymbol{S}}^i\boldsymbol{\lambda}^i_{old} \end{array} \tag{3.39}$$

Ce qui signifie qu'il est aussi possible d'améliorer les propriétés des fonctions ondelettes duales.

Construction depuis la Lazy Wavelet Ainsi, en partant de la *Lazy Wavelet*[Swe98], et en l'enrichissant pour garantir telle ou telle propriété, il est possible de construire pas à pas une Analyse Multi-Résolution enrichie.

La *Lazy Wavelet* est la plus simple des transformées en ondelettes. Elle consiste simplement, en partant d'une fonction décrite par ses coefficients $\boldsymbol{\lambda}^i$, à séparer les échantillons pairs et impairs. Les pairs deviennent les coefficients $\boldsymbol{\lambda}^{i-1}$ de la résolution inférieure tandis que les impairs deviennent les $\boldsymbol{\gamma}^{i-1}$, soit :

$$\begin{array}{rcl} \boldsymbol{\lambda}^{i-1}_l & = & \boldsymbol{\lambda}^i_{2l} \\ \boldsymbol{\gamma}^{i-1}_l & = & \boldsymbol{\lambda}^i_{2l+1} \end{array} \tag{3.40}$$

Cette transformée est ensuite enrichie en alternant les étapes de DUAL LIFTING SCHEME et LIFING SCHEME. Ce schéma est appelé *cakewalk* et est représenté dans la figure 3.11.

La transformée inverse est obtenue simplement en parcourant ce schéma en sens inverse, comme montré dans la figure 3.11.

3.5.3 Le Lifting Scheme

Il est parfois fait référence à l'étape LIFTING SCHEME sous le nom de *prédiction*, tandis que l'étape DUAL LIFTING SCHEME est appelée *Mise à jour*. Dans le cas 1D, il est souvent fait usage

FIG. 3.11 – Décomposition (a) et Reconstruction (b) Cakewalks

d'une étape de *prédiction* qui utilise simplement deux échantillons λ^i_{2l} et λ^i_{2l+2} pour prédire la valeur de $\tilde{\lambda}_{2l+1} = \frac{\lambda_{2l}+\lambda_{2l+2}}{2}$. Les *détails* $\gamma^{i-1}_l = \lambda^i_{2l+1} - \tilde{\lambda}^i{}_{2l+1}$ ainsi obtenus sont alors plus petits, ce qui, dans certains cas, autorise une compression.

3.5.4 Ondelettes Volumiques

Lifting Scheme 3D

De nombreuses transformées en ondelettes ont été construites en 1D [Swe98] et en 2D [FPS96], mais peu en 3D [BCCG06]. Nous allons donc détailler dans les prochaines sections comment le LIFTING SCHEME peut-être utilisé pour construire une transformée en 3D, et quelles restrictions impose la 3D.

Schéma de subdivision

Une Analyse Multi-Résolution, et la transformée en ondelettes associée nécessitent un schéma de subdivision pour créer une résolution fine à partir d'une résolution grossière. Dans le cas 1D cela était fait en subdivisant chaque intervalle du support en deux.

Dans les cas 3D, notre support étant un tétraèdre, ce dernier est subdivisé en 8 tétraèdres en utilisant le schéma de subdivision de la figure 3.12, c'est à dire qu'un nœud de la nouvelle subdivision est ajouté au milieu de chaque segment. Par analogie avec le cas 1D, les coefficient ondelettes associés avec ces nouveaux nœuds sont notés γ^{i+1}_{2l+1}, tandis que les valeurs du signal associées aux nœuds parents après subdivision sont notées λ^{i+1}_{2l} et λ^{i+1}_{2l+2}. Voir la figure 3.13.

Lazy Wavelet

Une fois le schéma de subdivision créé, la construction de la *Lazy Wavelet* est immédiate. Comme dans le cas 1D, les valeurs associées aux nœuds *pairs*, c'est à dire les sommets des tétraèdres, deviennent les coefficients d'échelle λ^i_{2l} et λ^i_{2l+2}. Les valeurs associées aux nœuds *impairs*, c'est à dire les milieux des segments deviennent eux les coefficients de détails γ^i_{2l+1}.

Dual Lifting Step

Cette transformée constitue, comme dans le cas 1D, un point de départ pour la construction d'une transformée plus complexe. Là encore, il est possible d'appliquer directement l'étape de

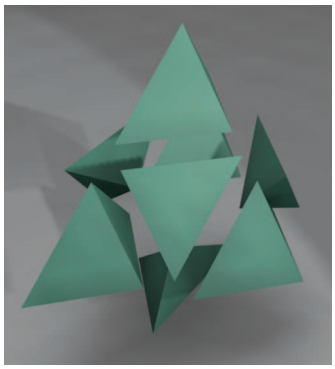


FIG. 3.12 – Schéma de subdivision tétraèdrique.

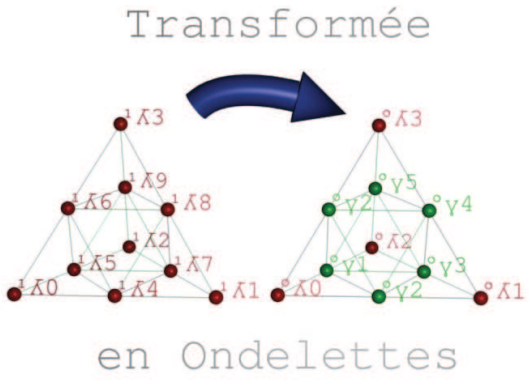


FIG. 3.13 – Transformée en ondelettes tétraèdrique.

LIFTING souvent rencontrée dans le cas 1D. C'est à dire que pour chaque point milieu, la valeur du signal est estimée avec cette approximation :

$$\tilde{\lambda}^{i+1}_{2l+1} = \frac{\lambda^{i+1}_{2l} + \lambda^{i+1}_{2l+2}}{2} \tag{3.41}$$

La valeur du signal à un point milieu est donc la moyenne du signal aux deux nœuds parents. Le coefficient ondelette peut alors s'écrire :

$$\gamma^{i}_{2l+1} = \lambda^{i+1}_{2l+1} - \frac{\lambda^{i+1}_{2l} + \lambda^{i+1}_{2l+2}}{2} \tag{3.42}$$

Transformée en Ondelettes Directe La transformée obtenue, constituée de la *Lazy Wavelet* à laquelle une étape de LIFTING a été ajoutée, peut être effectuée à l'aide d'opérations matricielles.

La transformée en ondelettes directe associée à un *cakewalk* peut être représentée par une matrice $\boldsymbol{W}$. Cette matrice est le produit d'autant de matrices que le *cakewalk* contient d'étapes de LIFTING et DUAL LIFTING :$\boldsymbol{W} = \boldsymbol{S}_n \cdots \boldsymbol{S}_0$. Ces matrices $\boldsymbol{S}$ sont celles du théorème du LIFTING SCHEME.

Dans notre cas, n'ayant qu'une étape de DUAL LIFTING, la transformée n'implique qu'une matrice. Si les valeurs sur lesquelles la transformée est appliquée sont stockées dans un vecteur de telle manière que les n_i valeurs de la résolution grossière soient les premières, et les $n_{i+1} - n_i$ valeurs associées aux nœuds milieux soient stockées à la suite[6], alors $\boldsymbol{W}$ peut être construite comme suit :

- $W_{k,k} = 1.0, \forall k \in [0, n_i - 1]$.
- $W_{k,l_1} = W_{k,l_2} = -0.5, \forall k \in [n_i, n_{i+1}]$, l_1 et l_2 étant les indices des nœuds parents de λ^i_k.

Ainsi :

$$\left| \begin{array}{l} \Lambda^i \\ \Gamma^i \end{array} \right. = W \cdot \Lambda^{i+1} \tag{3.43}$$

où Λ^i est un vecteur contenant le signal à la résolution grossière et Γ^i un vecteur contenant les coefficients ondelettes.

[6]Ce classement des points est en fait la *Lazy Wavelet*.

Transformée en Ondelettes Inverse De manière similaire, la transformée en ondelettes inverse est le produit d'autant de matrices qu'il y a d'étapes de LIFTING et DUAL LIFTING : $\boldsymbol{W}^{-1} = \boldsymbol{S}_n^{-1} \cdots \boldsymbol{S}_0^{-1}$. Là encore, il n'y a qu'une étape de DUAL LIFTING, donc seule une matrice est nécessaire. W^{-1} est alors :

- $W_{k,k} = 1.0, \forall k \in [0, n_i - 1]$.
- $W_{k,l_1} = W_{k,l_2} = 0.5, \forall k \in [n_i, n_{i+1} - 1]$, l_1 et l_2 étant les indices des nœuds parents de λ_k^i.

Ce qui donne :

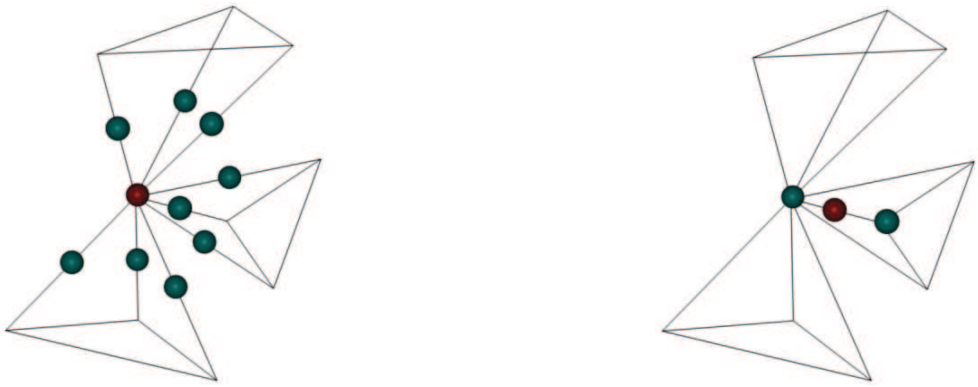
$$\Lambda^{i+1} = W^{-1} \cdot \left| \begin{array}{c} \Lambda^i \\ \Gamma^i \end{array} \right. \tag{3.44}$$

3.5.5 Enrichissement des propriétés de la transformée en ondelettes volumique

Étapes de LIFTING et DUAL LIFTING supplémentaires

La transformée en ondelettes volumique que nous avons ainsi créée ne présente pas de propriété particulière, et notamment, elle ne garantie pas la conservation de la moyenne d'une résolution à l'autre ni pour le signal, ni pour les fonctions ondelettes.

Conservation de la moyenne pour les fonctions ondelettes BOSCARDIN à montré dans [BCCG06] qu'il est possible de conserver la moyenne facilement pour les fonctions ondelettes d'un résolution à l'autre. Cela signifie que la moyenne des γ_l^i est constante d'une résolution à l'autre si une étape de DUAL LIFTING est ajoutée.



(a) (b)

FIG. 3.14 – Additionner sur tout le maillage la somme pondérée de tous les voisins bleus du nœud rouge revient à additionner sur tout le maillage la somme pondéré par les coefficients associés aux deux voisins rouge du nœud bleu.

Conservation de la moyenne pour le signal Conserver la moyenne pour les fonctions ondelettes est simple à mettre en œuvre, mais n'est pas particulièrement intéressant. Ce qui serait plus avantageux serait de pouvoir conserver la moyenne du signal d'une résolution à l'autre.

Dans le cas 1D, SWELDENS a montré qu'une étape supplémentaire de LIFTING permet d'assurer facilement cette conservation. Or, il apparaît en étudiant les équations menant à cette étape

de conservation de la moyenne, que cette dernière est due au fait qu'un noeud à une résolution i est entouré exactement de deux nouveaux noeuds à la résolution $i+1$.

En effet, si les coefficients λ_j^i à la résolution i sont mis à jour, dans le but de conserver la moyenne, de la façon suivante :

$$\lambda_j^i = \lambda_{2j}^{i+1} + \sum_{k_j \in V_j} \alpha_{k_j j} \gamma_{k_j}^i \tag{3.45}$$

alors la somme de ces coefficients en chacun des noeuds est :

$$\sum_{j \in \Lambda^i} \lambda_j^i = \sum_{j \in \Lambda^i} \lambda_{2j}^{i+1} + \sum_{j \in \Lambda^i} \sum_{k_j \in V_j} \alpha_{k_j j} \gamma_{k_j}^i \tag{3.46}$$

A noter que dans la suite, dans un soucis de simplification, les symboles Λ^i et Γ^i font aussi référence aux indices j des nœuds associés aux λ^i et γ^i respectivement. C'est à dire que $j \in \Gamma^i$ signifie que j parcourt l'ensemble des indices des éléments γ^i de j Γ^i

La double somme

$$\sum_{j \in \Lambda^i} \sum_{k \in V_j} \alpha_{k_j j} \gamma_{k_j}^i$$

peut être simplifiée en

$$\sum_{j \in \Gamma^i} (\alpha_{j1} + \alpha_{j2}) \gamma_j^i$$

en remarquant que sommer pour chaque nœud de Λ^i la somme pondérée par les α_{kj} dans coefficients ondelettes γ_k^i revient à sommer pour chaque nœuds de Γ^i le produit du γ_j^i par chaque α_{kj} associé aux deux nœuds pères de γ_j. Dit autrement, sommer pour chaque nœud de la résolution grossière la somme pondérée de valeurs associées aux nouveaux nœuds *impairs* voisins, revient à sommer pour chaque nouveau nœuds *impairs* la somme pondéré des deux valeurs associées au nœud père. La figure 3.14 illustre ce changement de sommation.

D'où

$$\sum_{j \in \Lambda^i} \lambda_j^i = \sum_{j \in \Lambda^i} \lambda_{2j}^{i+1} + \sum_{j \in \Gamma^i} (\alpha_{j1} + \alpha_{j2}) \gamma_j^i \tag{3.47}$$

Or, notant d'après l'étape de DUAL LIFTING, que

$$\gamma_j^i = \lambda_{2j+1}^{i+1} - \frac{\lambda_{2j}^{i+1} + \lambda_{2j+2}^{i+1}}{2}$$

il vient :

$$\sum_{j \in \Lambda^i} \lambda_j^i = \sum_{j \in \Lambda^i} \lambda_{2j}^{i+1} + \sum_{j \in \Gamma^i} (\alpha_{j1} + \alpha_{j2}) \lambda_{2j+1}^{i+1} - \sum_{j \in \Lambda^i} \sum_{k_j \in V_j} \lambda_{2j}^{i+1} \alpha_{k_j j} \tag{3.48}$$

Donc,

$$\sum_{j \in \Lambda^i} \lambda_j^i = \sum_{j \in \Lambda^i} (1 - \sum_{k_j \in V_j} \alpha_{k_j j}) \lambda_{2j}^{i+1} + \sum_{j \in \Gamma^i} (\alpha_{j1} + \alpha_{j2}) \lambda_{2j+1}^{i+1} \tag{3.49}$$

Ainsi, pour assurer que $\frac{\sum_{j \in \Lambda^i} \lambda_j^i}{N^i} = \frac{\sum_{j \in \Lambda^{i+1}} \lambda_j^{i+1}}{N^{i+1}}$, c'est à dire que la moyenne soit la même aux deux résolutions, il faut que :

$$(1 - \sum_{k_j \in V_j} \alpha_{k_j j}) = (\alpha_{j1} + \alpha_{j2}) = \frac{N^i}{N^{i+1}} \tag{3.50}$$

ce qui n'a pas de solution que pour le cas 1D, où les voisinages V_j ne contiennent que deux nœuds et où $\alpha_{j1} = \alpha_{j2} = \frac{1}{4}$, et où le rapport entre les deux résolutions $\frac{N^i}{N^{i+1}} = \frac{1}{2}$. Il semble donc difficile de conserver la moyenne dans le cas 3D.

3.5.6 Élements Finis et Ondelettes

La décomposition en ondelettes que nous avons créée s'appuie sur le pendant 3D de la fonction chapeau 1D. C'est donc une fonction associée à un nœud, et dont le support est constitué par l'ensemble des tétraèdres contenant ce sommet. Voir la figure 3.16. Cette fonction vaut 1 en ce nœud et décroît linéairement jusqu'à atteindre 0 sur les faces opposées.

Cette fonction est donc différente de celle utilisée classiquement avec des éléments finis linéaires. En effet, les fonctions de forme utilisées dans le cas linéaire sont définies sur un tétraèdre, sont au nombre de 4, et valent 1 en un nœud et 0 sur la face opposée. Nous allons donc voir comment construire des éléments finis avec ces fonctions, et quel lien peut être établi avec les matrices éléments finis habituels et les éléments finis que nous nommons *chapeau* en référence à la fonction chapeau 1D ou 2D qui est utilisée ici en 3D.

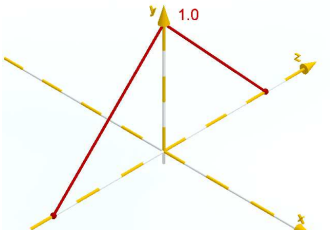


FIG. 3.15 – La fonction chapeau 1D.

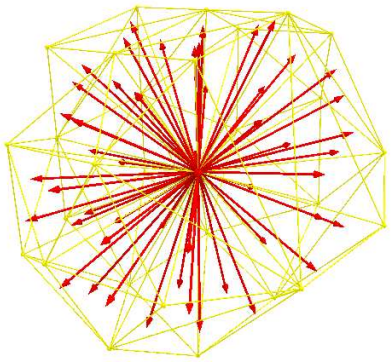


FIG. 3.16 – La fonction chapeau 3D. Elle vaut 1 au centre, et décroît linéairement selon les flèches rouges jusqu'à 0.

Élements Finis *chapeau*

Nous employons pour les éléments finis *chapeau* la même méthode que celle employée avec les éléments finis tétraèdriques. Nous cherchons toujours à intégrer une loi physique sur un domaine Ω quelconque dont la forme ne permet pas d'intégration analytique. Nous discrétisons donc à nouveau le domaine en éléments ayant cette fois-ci comme fonction d'interpolation la fonction *chapeau*. Cela nous permet à nouveau d'obtenir un opérateur d'interpolation global $\mathcal{I}_h$. Enfin, l'utilisation de cet opérateur nous permet de découper l'intégrale en une somme d'intégrales sur des domaines simples.

Nous allons donc définir l'opérateur d'interpolation locale puis globale pour ce type d'éléments finis dans les paragraphes suivantes.

Opérateur d'interpolation Locale Dans le cas canonique, l'opérateur d'interpolation locale est utilisé pour calculer la valeur d'une grandeur physique au sein d'un tétraèdre, en connaissant sa valeur pour chaque sommet du tétraèdre.

Pour notre élément fini *chapeau*, nous ne connaissons qu'une valeur, celle associée au sommet commun à tous les tétraèdres du support. Ce n'est pas suffisant pour interpoler une grandeur physique. La valeur de la grandeur physique au sein de l'élément dépend des autres fonctions *chapeau* dont le support intersecte le support de la fonction φ_b considérée. Nous devons donc interpoler cette grandeur physique en utilisant les restrictions sur le support de φ_b des fonctions *chapeau* voisines $\varphi_{i|b}$.

L'opérateur d'interpolation global devient alors :

$$I_b : v \rightarrow v(x_b) \cdot \varphi_b + \sum_{i=1}^{n_{\mathcal{V}(b)}} v(x_i) \cdot \varphi_{i|b} \tag{3.51}$$

où v est la fonction à interpoler, x_b et φ_b sont respectivement la position du nœud et la fonction *chapeau* associées à l'élément *chapeau*. x_i et $\varphi_{i|b}$ sont respectivement la position du nœud et la restriction de la fonction *chapeau* associées aux nœuds voisins $\mathcal{V}(b)$.

Ainsi, $I_b(u)$ interpole le déplacement u à l'intérieur d'un élément *chapeau*.

Opérateur d'interpolation Globale L'opérateur d'interpolation globale $\mathcal{I}_h$ peut maintenant être défini facilement pour le domaine Ω_h entier. Cette opération, faite élément par élément dans le cas tétraèdrique est faite fonction par fonction :

$$\mathcal{I}_h : v \rightarrow \sum_{b \in \mathcal{N}_h} v(x_b) \cdot \varphi_b \tag{3.52}$$

où $\mathcal{N}_h$ est l'ensemble des fonctions *chapeau* du maillage.

Lien avec les éléments finis tétraèdriques Dans l'équation 3.51, il apparaît que pour chacun des tétraèdres du support de φ_b, la restriction des fonction voisines $\varphi_{i|b}$ dote chacun de ces tétraèdres de 4 fonctions d'interpolation qui sont les même que dans le cas d'éléments finis tétraèdriques. Il en résulte que les matrices globales de l'équation 2.44 du chapitre précédent sont les mêmes que l'on utilise des éléments finis tétraèdriques ou *chapeau*.

3.5.7 Solveur hiérarchique et ondelettes

Comme nous venons de la voir dans la section ci-dessus, les matrices globales obtenues par assemblage selon la méthode des éléments finis sont les mêmes que soient utilisés des éléments finis tétraèdriques ou *chapeau*. Il est donc possible de ré-utiliser directement du code existant pour créer des solveurs hiérarchiques par ondelettes.

Opérateur de prolongation et de restriction

Nous l'avons vu dans la section 3.3.3, les solveurs hiérarchiques reposent sur l'alternance de phases de lissage et de phases de transfert d'une résolution à une autre.

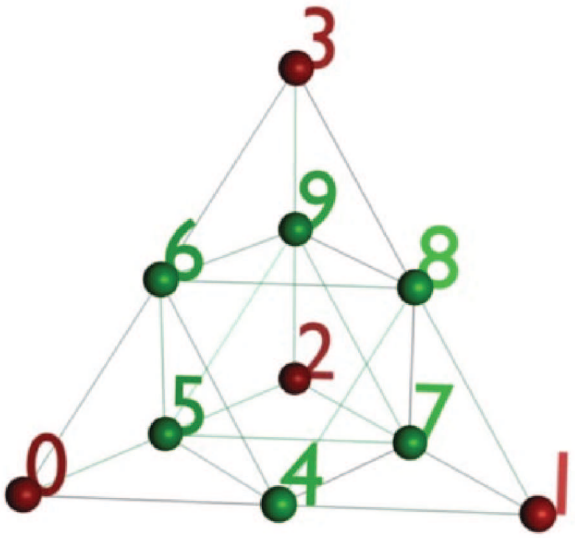


FIG. 3.17 – Numérotation du schéma de subdivision.

Introduire des transformées en ondelettes dans ce type de solveur se fait assez naturellement en utilisant des opérateurs de prolongation $\boldsymbol{P}$ et restriction $\boldsymbol{R}$ basés sur les transformées en ondelettes :

$$\boldsymbol{P} = \boldsymbol{W}_*^{-1} \tag{3.53}$$

$$\boldsymbol{R} = \boldsymbol{P}^T = \boldsymbol{W}_*^{-T} \tag{3.54}$$

où $\boldsymbol{W}_*^{-1}$ contient seulement les colonnes de $\boldsymbol{W}^{-1}$ correspondant aux nœuds grossiers, *i.e.* les coefficients ondelettes sont écartés :

$$\Lambda^i = W_*^{-1} \cdot \Lambda^{i-1} \tag{3.55}$$

La résolution se poursuit ensuite suivant l'algorithme 5.

L'interpolation barycentrique comme transformée en ondelette pour les maillages hiérarchiques imbriqués

La transformée en ondelettes volumique que nous avons définie est exactement identique à une interpolation barycentrique !

En effet, la transformée en ondelettes requiert des maillages affinés successivement selon le schéma de subdivision de la figure 3.17, où chaque nouveau point se trouve au milieu d'un segment. Les coordonnées barycentriques d'un de ces points est une permutation des valeurs $\{0, 0, 0.5, 0.5\}$. Par exemple le point 5 de la figure 3.17 a pour coordonnées barycentriques $\{0.5, 0, 0.5, 0\}$. Les coordonnées associées aux parents du nœuds sont donc 0.5. Donc pour un maillage à la résolution $i+1$ construit avec ce schéma depuis un maillage à la résolution i, l'interpolation barycentrique $\boldsymbol{B}$ s'écrit :

- $\boldsymbol{B}_{k,k} = 1.0, \forall k \in [0, n_i - 1]$.
- $\boldsymbol{B}_{k,l_1} = \boldsymbol{B}_{k,l_2} = 0.5, \forall k \in [n_i, n_{i+1} - 1]$, l_1 et l_2 étant les indices des nœuds parents du nœud k.

Sous réserve que les nœuds du maillage grossier i soient stockés aux indices $k \in [0, n_i - 1]$. Cette matrice $\boldsymbol{B}$ contient alors justement les valeurs qui sont utilisées dans la partie W_*^{-1} de la matrice W^{-1} qui est utilisée comme prolongateur.

3.6 Perspectives

Les travaux que nous avons présentés dans cette section sont évidemment perfectibles.

Amélioration du mailleur hiérarchique

C'est le cas notamment de notre mailleur hiérarchique, qui s'intéresse pour l'instant uniquement à la surface du maillage. Nous avons commencé à développer quelques une nouvelle méthode permettant de tenir compte aussi du maillage intérieur.

L'heuristique que nous avons proposé pour adapter la surface peut être améliorée en appliquant les méthodes proposées par WINKLER dans [WHG08], pour non seulement améliorer l'approximation de la surface cible, mais aussi pour assurer une qualité de maillage tétraèdrique supérieure. Nous étendons donc ici leur méthode au cas des tétraèdres.

Dans un premier temps, pour assurer une meilleure approximation de la surface, nous utilisons la distance de HAUSDORFF simplifiée que WINKLER propose :

$$\sum_i \alpha_i ||v_i - \tilde{v}_i||^2 + \sum_j \beta_j ||w_j - \tilde{w}_j||^2 \tag{3.56}$$

où les $\boldsymbol{v}_i$ sont des points de la surface à améliorer, et les $\tilde{\boldsymbol{v}}_i$ sont les points correspondants de la surface cible les plus proches, et de manière similaire, les $\boldsymbol{w}_j$ sont des points de la surface cible tandis que les $\tilde{\boldsymbol{w}}_j$ sont les points correspondants les plus proches de la surface à améliorer. Le calcul de la valeur des paramètres α_i et β_j est détaillé dans [WHG08].

Les paramètres entrant en compte pour la minimisation de cette distance sont donc les $\boldsymbol{v}_i$ et les $\tilde{\boldsymbol{w}}_j$, les $\tilde{\boldsymbol{v}}_i$ et $\boldsymbol{w}_j$ étant fixes puisque appartenant à la surface cible. De plus, les $\tilde{\boldsymbol{w}}_j$ peuvent être exprimés à l'aide de coordonnées barycentriques en fonction des $\boldsymbol{v}_i$, du coup, la distance à minimiser est simplement une fonction quadratique de v_i et sa minimisation se fait immédiatement en résolvant un système linéaire de la forme :

$$\boldsymbol{A}\boldsymbol{v} = \boldsymbol{c} \tag{3.57}$$

Dans un second temps, pour assurer une forme plus homogène aux tétraèdres, et éviter ainsi l'apparition de tétraèdres dégénérés, la position des sommets $\boldsymbol{v}_i$ des tétraèdres est lissée en appliquant itérativement :

$$\bar{\boldsymbol{v}}_i = \sum_{j \in N_i} \lambda_{ij} \boldsymbol{v}_j \tag{3.58}$$

où N_i est l'ensemble des indices des sommets voisins de i, et les λ_{ij} sont définis tels que $\sum_{j \in N_i} \lambda_{ij} = 1$, ce qui garanti que $\boldsymbol{v}_i$ restera à l'intérieur de l'enveloppe convexe créée par les $\boldsymbol{v}_j$. Il existe plusieurs possibilités pour le choix des λ_{ij}, comme une valeur constante. Cependant WINKLER indique que le meilleur résultat est obtenu en utilisant les cordonnées moyennes de $\boldsymbol{v}_i$

(a) (b)

FIG. 3.18 – Mailleur hiérarchique avec distance de Hausdorff. (a) : Maillage non adapté ; (b) : Maillage affiné et adapté.

par rapport au $\boldsymbol{v}_j$. Voir [Flo] pour plus de détails. Ce lissage, bien qu'itératif sur les sommets, peut être effectué de manière matricielle en résolvant le système :

$$\boldsymbol{B}\boldsymbol{v} = 0 \tag{3.59}$$

Les coefficients B_{ii} de la matrice $\boldsymbol{B}$ valent 1, tandis que les coefficients B_{ij} sont nuls sauf si le sommet j est un voisin du sommet i auquel cas $B_{ij} = -\lambda_{ij}$.

Ce lissage ne peut cependant pas être appliqué sans contrainte, sous sa forme itérative, car il a tendance à faire rétrécir le maillage. Sous sa forme matricielle, sa résolution implique aussi l'existence de contraintes, sans lesquelles la solution obtenue est $\boldsymbol{v} = 0$. C'est pourquoi son application est couplée à l'optimisation de la surface pour former un unique système :

$$(\mu\boldsymbol{B} + (1-\mu)\boldsymbol{A})\,\boldsymbol{v} = (1-\mu)\boldsymbol{c} \tag{3.60}$$

Selon WINKLER, $\mu = \frac{||\boldsymbol{B}||}{3||\boldsymbol{A}||+||\boldsymbol{B}||}$ constitue une bonne valeur en pratique.

Résultats Cette méthode fonctionne parfaitement bien, comme en témoignent les figures 3.18, pour ce qui est de l'adaptation de la surface du nouveau maillage à la surface cible. Elle semble aussi améliorer l'homogénéité et la qualité du maillage tétraèdrique interne, mais nous manquons pour l'instant d'outils d'évaluation pour faire les tests nécessaires et le prouver définitivement.

Amélioration des méthodes ondelettes

Tous les travaux que nous avons effectués sur les ondelettes, bien qu'ayant un attrait théoriques, n'ont pas permis d'accroître les performances ou la précision de nos solveurs hiérarchiques. L'un des freins est notamment l'impossibilité de garantir en utilisant le LIFTING SCHEME la conservation de la moyenne d'une résolution à l'autre dans le cas 3D.

Pour éviter cet écueil, il serait intéressant de construire directement la méthode des éléments finis en utilisant des fonctions de forme qui soient justement des fonctions ondelettes. C'est à dire, au lieu d'utiliser un schéma discret, le LIFTING SCHEME, pour construire les transformées, il

serait intéressant de procéder de manière continue en utilisant directement des fonctions échelles et ondelettes dans les équations. Ainsi, en choisissant judicieusement ces fonctions il devrait être possible d'assurer la conservation de la moyenne et peut-être même d'autres moments, ce qui devrait impacter sur la qualité du solveur.

3.7 Conclusion

Nous avons rappelé dans la section 3.2 de ce chapitre comment une équation différentielle pouvait être intégrée dans le temps. Nous avons en présenté les deux types de schémas d'intégration envisageables, explicite et implicite. Leurs propriétés en terme de stabilité et d'aisance de résolution ont été décrites rapidement ce qui nous a permis de retenir le schéma implicite pour nos simulations en raison de sa stabilité accrue.

La nécessité de résoudre efficacement et précisément les équations dérivées de ces schémas nous a amenés à étudier les différents types de solveurs généralement rencontrés. A l'issue de cette présentation, dans la section 3.3, nous avons identifié deux limitations des solveurs hiérarchiques : leur incapacité de résoudre avec la géométrie exacte d'un objet lorsque des hiérarchies de grilles imbriquées sont utilisées, mais aussi la lenteur de l'application des opérateurs de prolongation et restriction.

Pour obvier à la première limitation, nous avons proposé un méthode pour calculer rapidement le produit de deux matrices creuses, avec pour seule condition la connaissance préalable de leur profil. Or dans le cas de simulation éléments finis, cette condition est toujours réalisée. Cette optimisation permet un gain de temps substantiel par rapport à un produit classique, même avec une librairie optimisée comme la MKL d'INTEL.

Pour la seconde limitation, nous avons introduit le concept de mailleur hiérarchique, dont le but est de construire des hiérarchies de maillages tétraèdriques imbriquées dont la surface converge vers la surface réelle de l'objet à simuler. De cette manière, nous pouvons bénéficier de la rapidité d'application des opérateurs de restriction et de prolongation propre aux solveurs hiérarchiques tout en garantissant une résolution pour une géométrie très proche de celle de l'objet réel.

Ces deux idées ont fait l'objet d'une publication à l'INTERNATIONAL SYMPOSIUM ON VISUAL COMPUTING [SDG07].

Enfin, le principe même des solveurs hiérarchiques ou mutligrilles, qui consiste en l'utilisation de plusieurs résolutions nous a amené à nous intéresser au lien entre ce type de solveurs et la théorie des ondelettes. Après avoir rappelé le fonctionnement du LIFTING SCHEME, nous avons montré comment utiliser ce schéma pour construire des transformées en ondelettes en 3D. Ensuite, nous avons détaillé comment utiliser ces transformées en ondelettes dans un contexte éléments finis. Enfin, nous avons établi que les opérateurs de restriction et de prolongation barycentriques, largement utilisés dans la communauté COMPUTER GRAPHICS était en fait constructibles depuis une transformée en ondelettes.

La décomposition ondelettes proposée est encore simple et dispose de peu de propriété analytiques sophistiquées. Si le problème a été bien traité pour une géométrie 1D, et également abordé dans le cas des surfaces, la question de savoir comment construire une hiérarchie volumique disposant de propriétés intéressantes, orthogonalité, vitesse de convergence supérieure à 1, reste à notre connaissance une question encore ouverte. Disposer, avec l'algorithme proposé,

de constructions de meilleures bases d'ondelette permettra, dans le framework défini, d'améliorer le comportement numérique de la méthode proposée.

Chapitre 4

Gestion des Contacts Efficace

Sommaire

4.1 Introduction

La gestion des contacts est une étape délicate de la simulation d'objects déformables ou non. En témoigne le nombre d'articles s'intéressant, depuis de nombreuses années, à ce problème : entre autres [BFA02, MC95, WT06, GOM+06, Ben07, MW88, JP04, MAC04, FBAF08, CMT02]. La difficulté réside dans le caractère non prévisible du contact, dans sa nature unilatérale, mais aussi, dans le cas du frottement, dans les non-linéarités. Tout cela concourre à rendre ces interactions potentiellement génératrices d'instabilités et difficiles à simuler. Il existe plusieurs approche à ce problème, suivant le comportement désiré, plus ou moins physique, et suivant la puissance de calcul à disposition.

Pour situer notre contribution, précisons d'abord la place de la gestion des contacts dans la simulation. Jusqu'à présent nous avons vu comment simuler la déformation d'un objet lorsque ce dernier est soumis à une force externe ou interne. Nous savons donc déformer un objet en appliquant une force sur ces frontières. Cependant, dans la réalité, lorsque un objet est manipulé,

ce dernier n'est pas soumis directement à des forces, mais plutôt à des contacts dont dérivent des forces.

Ce chapitre présente comment ces forces de contacts peuvent être calculées, et comment notre méthode, le COMPLIANCE WARPING permet la gestion des contacts par des problèmes de complémentarité de manière interactive et avec des fondements physiques.

Pour calculer des forces de contact, il faut dans un premier temps déterminer s'il y a contact. Il faut pour cela calculer la distance entre les frontières des deux objets. Cette première phase est nommée *détection de collision*. Ensuite, une fois cette distance calculée, si les deux objets sont en contact, alors les forces de contact peuvent être calculées. Cette étape, nommée *gestion des contacts*, peut-être extrêmement consommatrice en temps de calcul si elle est basée sur le respect de lois physiques telles que les lois de SIGNORINI et COULOMB dans la mesure où il faut pour cela calculer la compliance de l'objet.

Le principe de notre méthode, le COMPLIANCE WARPING est justement d'éviter ce calcul est utilisant un compliance précalculée. Cette compliance précalculée, tire parti de la bonne approximation d'un modèle corotationnel élémentaire que constitue un modèle corotationnel nodal. Utilisée conjointement à la rapidité d'inversion du modèle nodal, cette propriété permet de gérer de manière interactive, stable, générique et sur des bases physiques le contacts entre un objet déformable et son environnement. Cette contribution a fait l'objet d'une publication dans COMPUTER GRAPHICS INTERNATIONAL [SDCG08].

Ce chapitre suit donc une progression naturelle, partant de la description de la phase de détection de collision dans la section 4.2.1, puis détaillant dans la section 4.2.2 les diverses méthodes permettant de calculer les forces de contacts. Suit ensuite une digression, dans la section 4.4, sur l'utilisation du modèle corotationnel nodal comme préconditionneur pour une simulation basée sur un modèle corotationnel élémentaire. C'est cette idée, qui nous a conduit à la mise au point de notre méthode de COMPLIANCE WARPING, qui permet de gérer rapidement et physiquement les contacts, et qui est détaillée dans la section 4.5.

4.2 Détection de Collision et Gestion des Contacts

4.2.1 Détection de collision

L'objet de la détection de collision est, étant connus n objets, de déterminer si ces objets sont en contact. Le premier constat qui peut être fait, est que la complexité de cette détection croît en $O(n^2)$. Il est donc nécessaire de restreindre rapidement et efficacement ce nombre n d'objets à un nombre $m << n$ d'objets potentiellement en contact. Cette première étape est nommée *broad phase collision detection*.

Ensuite, connaissant les m objets potentiellement en contact, il va falloir déterminer si ces objets sont réellement en contact. Pour cela, une phase de détection de collision, dite *narrow phase collision detection*, se charge de calculer la distance d'interpénétration des deux objets. Il existe de multiples méthodes pour calculer ces informations de contacts, et ces dernières utilisent généralement des structures d'accélération qui dépendent en partie de la manière dont sont décrites les surfaces des objets.

À ces deux difficultés vient s'ajouter un autre problème, dû au caractère déformable des objets. En effet, ce type d'objet, par essence, voit sa géométrie évoluer au cours de la simulation,

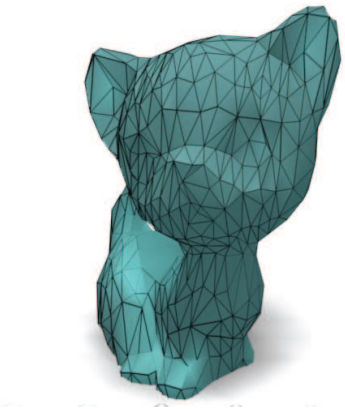


FIG. 4.1 – Représentation par soupe de triangles.

FIG. 4.2 – Représentation par voxels.

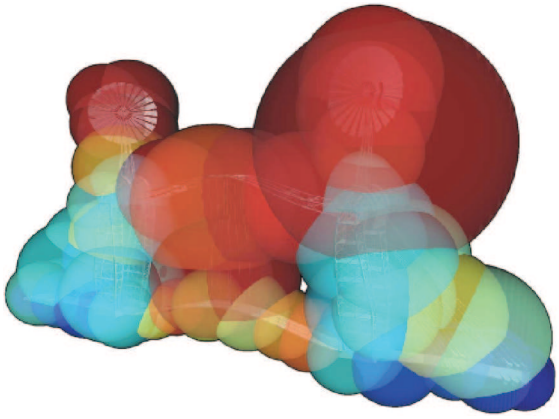


FIG. 4.3 – Bounding Sphere. Issue de [JP04].

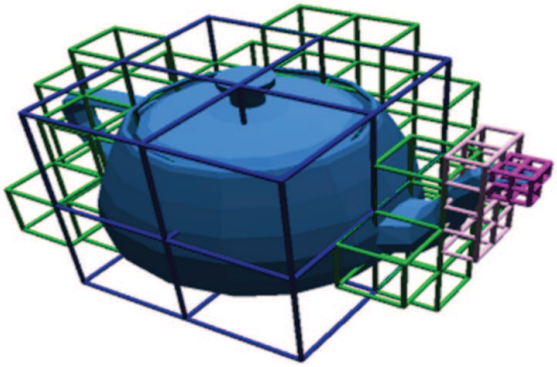


FIG. 4.4 – Octree. Issue de [DMG05].

et requière la mise à jour des structures d'accélération utilisées pour la détection. Or cette opération peut être très gourmande en temps de calcul.

Nous présentons donc rapidement dans la suite les réponses apportées par la communauté à ces problèmes. Celles-ci sont reprises de [TKH+05] et enrichies des derniers articles significatifs.

Description des géométries

Soupe de Polygones La grande majorité des objets graphiques 3D sont représentés à l'aide de soupe de polygones, généralement des triangles. Voir figure 4.1 Cette représentation est donc utilisée régulièrement pour effectuer des détections de collisions.

Le problème de ce genre de représentation réside dans le coût du calcul de la distance minimale entre deux objets. En effet, si ces deux objets sont chacun décrit par m et n polygones, alors il faut effectuer mn calculs de distance. Il est donc nécessaire d'introduire des structures d'accélération pour réduire au maximum le nombre de calcul de distances. Ces structures sont présentées dans la suite.

Les méthodes basées sur une hiérarchie de volumes englobants

En COMPUTER GRAPHICS, la surface d'un objet est traditionnellement représentée par une soupe de polygones, ou plus communément de triangles. Ainsi, lorsque la ou les distances minimales entre deux objets doivent être déterminées, il faut parcourir l'ensemble des polygones d'un maillage et calculer pour chacun de ces polygones la distance à chacun des polygones de l'autre surface. Le temps de calcul nécessaire à ce genre d'opération est extrêmement prohibitif et interdit toute simulation interactive dès lors que les modèles utilisés sont complexes.

Le but des BOUNDING VOLUME HIERARCHIES, ou BVHs est de réduire ce temps de calcul en regroupant les polygones décrivant la surface suivant des critères de proximités, et ce récursivement. C'est à dire que partant d'une soupe de polygones, n groupes distincts sont créés, et les polygones de chacun de ces n groupes sont à nouveau regroupés en sous-groupes distincts suivant des critères de proximité. A noter qu'en procédant de la sorte, les sous groupes sont contenus dans le groupe père, d'où la notion de hiérarchie. La décomposition en sous-groupes est généralement menée jusqu'à ce que les feuilles de l'arbre ainsi créé ne contiennent qu'un très faible nombre de polygones.

Les différents types de BVHs se distinguent par la façon dont elle construisent ces hiérarchies de volumes englobant. Les plus usitées sont détaillées ci-dessous.

Les Axis Aligned Bounding Box Tree Les AXIS ALIGNED BOUNDING BOXES, ou AABBs sont des boites englobantes dont les faces sont othogonales aux 3 directions de l'espace. C'est un cas particuliers des K-DOPs, avec $k = 6$. La simplicité de construction de ces boites et la facilité avec laquelle l'intersection entre deux AABB est réalisée ont rendu leur usage très fréquent, notamment avec des modèles déformables [LAM01, vdB97].

Les sphères englobantes Là encore, le principe est très similaire aux volumes englobants précédents. La seule différence est qu'on utilise des sphères englobantes. Voir la figure 4.3. Là encore, la simplicité du calcul d'intersection entre deux sphères, et l'invariance des sphères aux rotations ont généralisé leur usage. HUBBARD les présente en détail dans [Hub95].

Les méthodes basées sur une partition de l'espace

Les méthodes basées sur la partition de l'espace poursuivent un but similaire à celui des hiérarchies de volumes englobant, c'est à dire qu'elles regroupent les triangles d'un maillage surfacique suivant des critères de proximité géométrique, mais en partitionnant cette fois l'espace dans lequel les objet sont inclus.

Comme dans les hiérarchies de volume, de nombreuses méthodes existent, et elles se distinguent par le procédé employé pour partitionner l'espace. Les méthodes les plus courantes sont présentées ci-dessous.

Les Kd-tree Les KD-TREE ont pour but de partitionner l'espace en séparant un ensemble de polygones ou de triangles par un plan orthogonal à l'une des 3 directions de l'espace. En opérant alternativement des séparations suivant le plan orthogonal à $\boldsymbol{x}$, $\boldsymbol{y}$ ou $\boldsymbol{z}$, une partition de l'espace est obtenue dans laquelle tout les polygones sont isolés les uns des autres. Ce type de partition,

du fait que les polygones traités soient d'un coté ou de l'autre d'un plan, appartient à la classe des BINARY SPACE PARTITIONS, ou BSPs.

Les méthodes de détection de collision se basent rarement totalement sur les KD-TREE. Généralement, ils ne sont utilisés que pour la *Broad Phase* [LJF05], et encore, on leur préfère généralement les BSPs qui utilisent des plans non orthogonaux aux directions principales. Les KD-TREE se cantonnent pour l'instant au domaine du lancé de rayons, mais l'apparition dans ce domaine de méthodes permettant une mise à jour très rapide en ligne offre des persepectives intéressantes.

Les Octrees Les OCTREES sont une structure de partitionnement de l'espace basée sur la subdivision du cube. Ainsi, la boite englobante d'un objet est successivement subdivisée en 8 cubes. Chacun de ces 8 cubes possède un flag indiquant s'il contient ou non de la matière. Si le cube ne contient pas de matière, la subdivision s'arrête, tandis que s'il en contient, il est a nouveau divisé en 8. Voir la figure 4.4.

Dans le cadre de la collision de détection, les cubes de la résolution la plus fine contenant de la matière stockent les triangles de la géométrie qu'ils approximent et qu'ils contiennent. La détection de collision se fait alors en trouvant pour deux OCTREES les triangles contenus dans leur intersection, et en effectuant des tests sur ces triangles.

KITAMURA dans [KSTK98] utilise ainsi pour le déformable des OCTREES qu'il reconstruit pour l'intersection des boîtes englobantes de deux objets, et ce pour chacun des deux objets. Les cellules de ces OCTREES sont marquées pleines si elles contiennent une triangle. Les travaux de DEQUIDT *et al.* [DMG05] utilisent aussi ce type de structure.

Le problème de ce type de partition de l'espace, c'est que le calcul d'intersection entre deux OCTREE est coûteux du fait même de l'utilisation de cube, ce qui n'est pas le cas des hiérarchies de sphères englobantes. Par contre, la construction de la partition est très rapide, contrairement aux hiérarchies de sphères englobantes.

Récemment RUFFALDI *et al.*, a donc essayé d'allier les avantages des sphères englobantes aux OCTREEs dans[RMB+08], en proposant d'accélérer les OCTREES en remplaçant chaque cube par une sphère. Le calcul d'intersection est alors plus rapide, du fait de l'utilisation de sphère, mais aussi en raison de leur invariance aux rotations.

Voxels / Points Shell L'autre représentation de volume couramment rencontrée est la *voxelisation*. Dans ce cas, l'espace est découpé, généralement en une grille de petits cubes. A chacun de ces cubes est associée un flag qui indique si ce cube est sur la surface. La surface d'un objet est alors représentée par un ensemble de cubes pleins. A cette carte de *voxels*, la *voxmap*, est associée une *point shell*, qui est une discrétisation sous forme de points et de normales de la surface de l'objet. Voir la figure 4.2.

La *voxmap* sert alors de structure d'accélération permettant de savoir immédiatement si un point est à l'intérieur ou à proximité d'un objet, tandis que la *point shell* est un ensemble de point approximant l'objet et qui seront testés pour détecter des collisions.

L'avantage d'une telle représentation est qu'elle permet de déterminer très rapidement si un point est à l'intérieur d'un objet. Pour cela, il suffit de déterminer dans quel cube ce point se trouve, et de tester ensuite si ce cube est plein, et à quel objet il appartient. Par contre, sa construction est assez lente.

Ce type de représentation est couramment construit pour des modèles issus de scanner 3D qui fournissent déjà des points organisés sous forme de grille, mais peut aussi être utilisé pour simuler des vêtements [ZY00].

Les méthodes basées sur le hardware graphique

Ce type de méthode exploite les capacités de projection des cartes graphiques pour estimer rapidement la collision entre deux objets, et éventuellement pour calculer une distance minimale entre ces deux objets. L'avantage principal est qu'aucun pré-calcul n'est nécessaire. L'auto collision peut aussi être gérée [FBAF08]. Par contre, ces méthodes entrent en concurrence avec les procédures de rendu graphique ce qui altère leurs performances.

L'une des première méthode de ce type, présentée dans [SF91] puis améliorée dans [BWS98] utilise simplement le *z buffer* de deux objets pour déterminer le plus petits intervalle séparant ces objets, ce qui est très efficace mais ne fonctionne qu'avec des objets convexes.

Une première approche pour gérer la détection de collision entre objets complexes, utilisant le graphique hardware peut-être trouvée dans [GRLM03]. Dans ce papier, Govindaraju *et al.* déterminent le Potentially Colliding Set, ou PCS, c'est à dire l'ensemble des objets s'intersectant potentiellement. Ils utilisent pour cela le *z-buffer* qui permet, suivant les 3 axes de l'espace, de savoir si un objet en recouvre un autre. Une fois le PCS déterminé, la détection est affinée de manière similaire pour trouver précisément les polygones s'intersectant potentiellement. Le calcul de distance se fait alors sur l'ensemble de ces polygones. Dans [GKLM07], Govindaraju a adapté plus particulièrement cette méthode au déformable.

Wei Chen *et al.* ont proposé dans [CWZ+04] une nouvelle méthode fonctionnant pour des objets quelconques, convexes ou concaves. Ils suivent les étapes suivantes : une Région d'Intérêt, ROI, est calculée à l'aide du CPU. Il s'agit de l'intersection de la boite englobante de deux objets. Cette ROI est ensuite voxelisée à l'aide d'un nouvel algorithme pour GPU. Ces voxels sont ensuite stockés dans un buffer de texture sous la forme d'une image 2D. La détection de collision se fait alors simplement en comparant les deux images. Cette méthode est applicable aussi bien au rigide qu'au déformable et permet d'atteindre des temps de calcul interactifs pour des maillages allant jusqu'au million de triangles.

Une approche sensiblement différente des précédentes, dans le sens ou elle n'utilise pas le *z-buffer*, mais des Layered Depth Images, a été présentée par Heidelberger *et al.* dans [HTG03]. Les LDI [SGwHS98] sont des images dans lesquelles chaque pixel contient les informations de profondeur des objets rendus. Cette information permet de calculer rapidement l'intersection entre deux objets.

Les méthodes stochastiques

Les méthodes de détection de collision que nous avons vues jusqu'à présent déterminaient quel point de chaque objet est le plus proche d'un autre, et essayaient ensuite de calculer précisément la distance séparant ces deux points. La distance minimale entre deux objets était alors connue exactement.

Bien que nécessaire dans des applications où l'exactitude des résultats importe, il s'avère que dans le cas de simulations interactives, où la plausibilité suffit, la connaissance précise des distances minimales n'est pas toujours nécessaire.

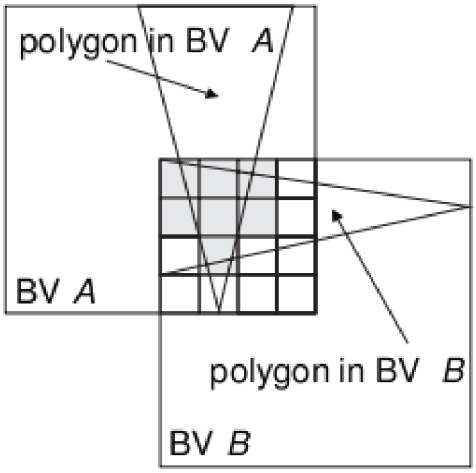


FIG. 4.5 – Le découpage en grille des intersections. Issue de [KZ03].

Les méthodes stochastiques s'appuient sur cette constatation pour calculer une estimation relativement précise de la distance minimale entre deux objets, et ce dans un temps moindre que les méthodes exactes. Deux travaux sont généralement cités pour illustrer ces méthodes.

Le premier [KZ03], développé par KLEIN *et al.*, s'appuie sur une hiérarchie de volumes englobants, et détermine s'il y a collision entre deux objets en s'aidant de probabilité de collision. La traversée de la hiérarchie est alors guidée par des probabilités basées sur la distribution des triangles de chaque nœud. Simplement, il calcule l'intersection $A \cap B$ de deux volumes englobants A et B de deux objets, puis la subdivise en une grille de cellules. Voir figure 4.5. Si une cellule contient suffisamment de polygones, cette cellule est considérée comme *possible collision cell*. Si une cellule est marquée comme *possible collision cell* pour les deux volumes A et B alors elle est reconnue comme *collision cell*. La connaissance du nombre total de cellules de $A \cap B$ étant des *collision cells* permet de calculer la probabilité qu'il y ait alors au moins x cellules en collision, ce qui permet d'évaluer la probabilité que les polygones de A et B s'intersectent. L'idée est alors de parcourir en priorité les nœuds ayant la plus grande probabilité d'intersection.

L'autre papier régulièrement cité est le rapport technique de DEBUNNE et al. [GD04]. L'idée de base de cette méthode est de calculer un ensemble de distances de manière quasi aléatoire, en favorisant néanmoins les zones où il y avait contact précédemment. La cohérence temporelle des collisions est donc exploitée à travers le maintient en permanence des collisions précédentes, et par l'ajout de manière stochastique de nouvelles distances de collision à proximité des anciennes. Cela permet d'obtenir d'excellentes performances, à condition que les contacts n'évoluent pas trop rapidement, du fait de la déformation ou du glissement.

Enfin, JOUSSEMET a développé dans [JCA06] une méthode de détection de collision basée sur des algorithmes génétiques.

Local Minimal Distance

Maintenant que nous avons présenté les différentes méthode permettant de déterminer si des objets s'intersectent, nous allons préciser la mesure de la distance que nous souhaitons obtenir : il s'agit de la Distance Minimale Locale, ou LMD.

De nombreuses applications nécessitent de connaître la distance minimale entre deux objets. Cette information, lorsque la géométrie de l'objet est modélisée par une soupe de polygones, peut être calculée à l'aide d'une hiérarchie de volumes englobants. En parcourant cette structure pour deux objets, la distance globale minimale est déterminée en descendant successivement dans les nœuds les plus proches. L'inconvénient de cette méthode est qu'elle ne donne que la distance minimal globale entre les deux objets. Or pour la gestion des contacts, il est important de connaître les distances minimales locales.

La LOCAL MINIMUM DISTANCE à un point est définie comme étant la norme du plus court vecteur reliant ce point à la surface de l'autre objet et qui soit parallèle à la normale aux deux surfaces.

Cette définition est généralisée des équations qui donnent les distances minimales locales pour des surfaces paramétrées $\boldsymbol{F}(u,v)$ et $\boldsymbol{G}(u,v)$. La distance entre deux points (u,v) de $\boldsymbol{F}$ et (s,t) de $\boldsymbol{G}$ est donnée par $||\boldsymbol{F}(u,v)-\boldsymbol{G}(u,v)||^2$, et cette distance est minimisée par les solutions de :

$$(\boldsymbol{F}-\boldsymbol{G})\cdot\frac{\partial \boldsymbol{F}}{\partial u}=0 \tag{4.1}$$

$$(\boldsymbol{F}-\boldsymbol{G})\cdot\frac{\partial \boldsymbol{F}}{\partial v}=0 \tag{4.2}$$

$$(\boldsymbol{F}-\boldsymbol{G})\cdot\frac{\partial \boldsymbol{G}}{\partial s}=0 \tag{4.3}$$

$$(\boldsymbol{F}-\boldsymbol{G})\cdot\frac{\partial \boldsymbol{G}}{\partial t}=0 \tag{4.4}$$

Cela signifie qu'une distance est une distance minimale locale si elle est parallèle aux normales des deux surfaces.

JOHNSON a proposé dans [JC01, JWC05] d'utiliser une hiérarchie de volumes englobants doublés de cônes décrivant l'étendu des normales pour déterminer rapidement ces LMD.

Méthode retenue

De toute les méthodes présentées précedemment, nous utilisons celles permettant de calculer les Distance Minimale Locales, c'est à dire les BOUNDARY VOLUME HIERARCHY. En effet, les LMD permettent de garantir la robustesse de la simulation, en déterminant non seulement la distance globale minimale à l'objet, mais aussi les potentiels futurs points de contacts.

L'exemple classique est celui d'un objet se déplaçant à proximité de la surface concave d'un autre objet. Cet objet peut entrer en collision en se déplaçant en ligne droite, mais peut tout aussi bien interpénétrer les flancs de la concavité en se déplaçant latéralement. Dans ce cas, les LMD, contrairement à la distance minimale globale, fournissent ces contacts potentiels et permettent leur intégration dans la simulation. Ainsi, si l'objet se déplace latéralement, la contrainte de collision est déjà intégrée dans le processus de résolution.

Autre conséquence de cette approche, les points de contact fournis ne varient pas brutalement dans le temps, contrairement à ceux fournis par une distance globale minimale. Cela contribue évidemment à assurer la stabilité de la simulation.

Par contre, elles présentent l'inconvénient de n'être pas très rapides en raison de la mise à jour nécessaire à chaque fois que l'objet se déforme.

4.2.2 Gestion des contacts

La phase de détection ayant été détaillée, nous savons maintenant déterminer si deux objets sont en contact, et le cas échéant nous savons calculer la distance d'interpénétration ainsi que la normale au contact. Ce que nous souhaitons faire maintenant, c'est, connaissant ces informations de contact ainsi que le comportement physique de l'objet déformable, calculer les efforts de contact et de frottement, et en définitive, calculer sa déformée.

Cette opération, qui consiste à déterminer l'effort généré par le contact, est une composante déterminante de la simulation d'un corps déformable. Le comportement des objets dans la scène est largement impacté par la façon dont sont gérés les contacts et les frottements. Une mauvaise gestion de ces derniers tend à rendre l'ensemble de la simulation peu réaliste. Il est donc primordial pour de nombreuses applications que les contacts et frottements simulés soient le plus possibles conformes à la réalité physique. Nous avons choisi, dans ce but, de suivre autant que possible les lois de COULOMB et SIGNORINI.

Sur cette contrainte de réalisme, qui nous contraint surtout sur le modèle à utiliser, se greffe en plus une contrainte de stabilité, qui incide sur le choix de la méthode. En effet, la gestion des contacts doit être suffisamment rapide pour permettre une simulation interactive, voire haptique, ce qui n'est pas évident si l'on désire un bon niveau de réalisme. De plus, lorsque plusieurs objets sont en contacts multiples, le calcul des efforts de contact devient particulièrement coûteux, ce qui altère encore les performances du système. Il faut donc employer des méthodes particulièrement efficaces !

Enfin, suite à une collision, le mouvement d'un objet est souvent radicalement modifié. La méthode employée doit donc être suffisamment stable pour que l'objet n'*explose* pas après la collision, et qu'il n'y ait tout simplement pas d'artefacts visuels.

Plusieurs méthodes tentent de répondre à ces exigences, avec plus ou moins d'efficacité. Nous les détaillons dans les sections suivantes, par ordre de réalisme mais aussi de complexité croissante, afin de motiver notre choix pour les méthodes par contraintes.

La gestion des contacts

Lorsque l'on souhaite gérer les contacts dans une simulation, la boucle de la simulation est généralement altérée comme précisé dans l'algorithme 11, même s'il est possible, comme dans le cas d'une résolution par *pénalités* de s'affranchir des étapes 4 à 6 en prenant en compte les forces de contact dès l'étape 1.

Le problème que nous cherchons à résoudre à travers la gestion des contacts correspond à l'étape 5 de l'algorithme 11. A l'issue de l'étape de détection de collision nous obtenons un ensemble de points de collision $\boldsymbol{p}$ auxquels sont associés des distances d'interpénétration $\boldsymbol{\delta}$ et des normales au contact $\boldsymbol{n}$. Le but de la gestion de contact est alors de déterminer les forces $\boldsymbol{f}$, qui appliquées aux points de contact $\boldsymbol{p}$ vont assurer la non-interpénétration des deux objets en contact. Notamment, les forces $\boldsymbol{f}$ et les distances d'interpénétration après application des forces de contact doivent respecter les lois de SIGNORINI et de COULOMB.

Loi de Signorini La loi de SIGNORINI établit l'orthogonalité entre les forces de contact $\boldsymbol{f}$ et les distances d'interpénétration $\boldsymbol{\delta}$:

$$\mathbf{0} \leq \boldsymbol{f} \perp \boldsymbol{\delta} \geq 0 \tag{4.5}$$

Algorithm 11 Étapes de la boucle de simulation. Les étapes en bleu sont celles ajoutées lorsqu les contacts son gérés.

1: **while** 1 **do**
2: Les forces externes s'appliquant sur l'objet sont déterminées.
3: Le mouvement libre est intégré sous l'action de ces forces. La déformée est alors connue.
4: Les hypothétiques collisions sont déterminées. Les distances d'interpénétration et les normales aux contacts sont alors connues.
5: Les forces de contact et de frottement sont calculées. C'est la phase de gestion de contacts.
6: Le mouvement est à nouveau calculé, mais cette fois avec les forces de contacts. C'est le mouvement contraint qui doit garantir la non-interpénétration des objets.
7: **end while**

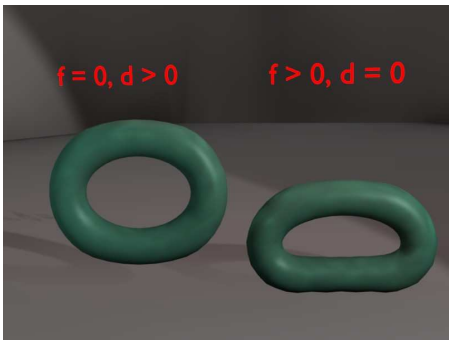


FIG. 4.6 – Loi de SIGNORINI.

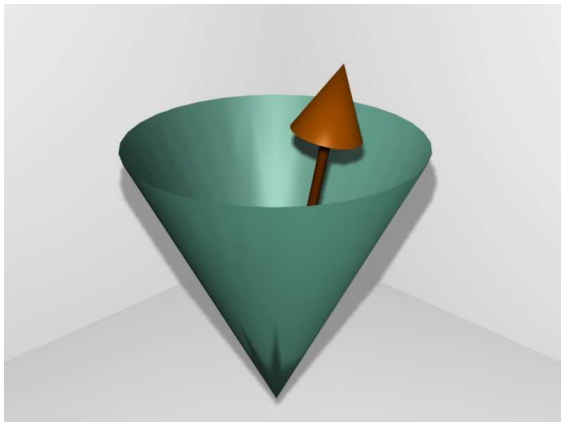


FIG. 4.7 – Cône de frottement et loi de COULOMB.

Elle caractérise le fait que deux configurations orthogonales peuvent être admisses par $\boldsymbol{f}$ et $\boldsymbol{\delta}$. Voir figure 4.6. Soit il y a contact, la distance d'interpénétration est nulle et la force de contact est positive, soit il n'y a pas contact, la distance d'interpénétration est donc positive et la force de contact nulle.

Loi de Coulomb La loi de COULOMB, quand à elle, caractérise le frottement sec. Elle affirme notamment que si la force de contact reste confinée à l'intérieur d'un cône de frottement, dont l'angle solide est déterminé par les propriété mécanique de l'objet, alors il n'y a pas glissement. Il y a alors adhérence. A l'opposé, si la force est sur le cône de frottement, alors il y a glissement avec frottement. Voir figure 4.7.

Nous allons voir dans les sections suivantes quelles méthodes peuvent être utilisées pour gérer le contact, et à quel point elles permettent de respecter les lois de SIGNORINI et COULOMB.

Espace de contacts et espace des mouvements L'une des premières difficultés inhérente à la gestion des contacts est la différence entre l'espace décrivant la déformation d'un modèle, et l'espace décrivant les contacts. En effet, comme en atteste la figure 4.8, lorsque le modèle est

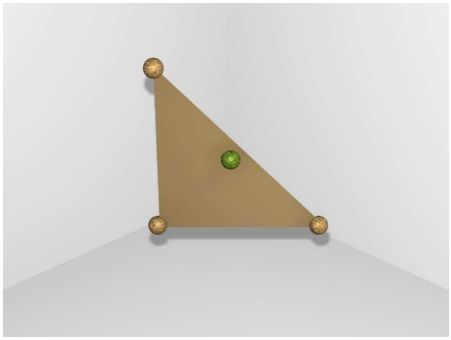


FIG. 4.8 – Espaces des contacts, en vert, et de mouvements, en brun.

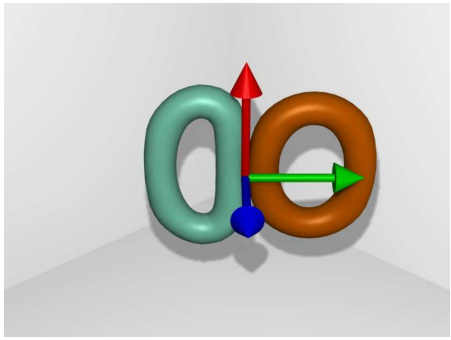


FIG. 4.9 – Le repère au contact, avec la normale $\boldsymbol{n}$ en vert, $\boldsymbol{t}$ en rouge, et $\boldsymbol{s}$ en bleu.

discrétisé, que ce soit un modèle surfacique, volumique, élément finis, *et cætera*, les déplacements ne sont connus qu'en des points discrets. Les déplacements associés à l'ensemble de ces points forment l'*espace des déplacements*.

A *contrario*, une collision peut avoir lieu en n'importe quel point de la surface de l'objet. L'espace décrit par ces points de contact est nommé *espace des contacts*.

Il est donc nécessaire de définir un opérateur permettant de passer de l'espace des contacts à celui des déplacements, afin de pouvoir résoudre les collisions. Soit $\boldsymbol{\alpha}$ un contact entre deux objets A et B. Il existe toujours une fonction $\mathbb{A}_\alpha$ qui permette de décrire le mouvement relatif des objets du contact en fonction des paramètres $\boldsymbol{q}_A$ et $\boldsymbol{q}_b$ de A et B, *i.e.* leur position :

$$\boldsymbol{\delta}_\alpha = \mathbb{A}_\alpha(\boldsymbol{q}_A, t) - \mathbb{A}_\alpha(\boldsymbol{q}_B, t) \tag{4.6}$$

Cette équation 4.6 peut être linéarisée pour obtenir une relation cinématique entre ces deux espaces :

$$\Delta\boldsymbol{\delta}_\alpha = \frac{\mathbb{A}_\alpha(\boldsymbol{q}_A, t)}{\partial \boldsymbol{q}}\Delta\boldsymbol{q}_A - \frac{\mathbb{A}_\alpha(\boldsymbol{q}_B, t)}{\partial \boldsymbol{q}}\Delta\boldsymbol{q}_B \tag{4.7}$$

Et avec $\mathbb{H}_\alpha = \frac{\mathbb{A}_\alpha}{\partial \boldsymbol{q}}$ et la relation duale :

$$\boldsymbol{r}_A = \mathbb{H}_\alpha(\boldsymbol{q}_A)^T\boldsymbol{f}_A,\ \boldsymbol{r}_B = \mathbb{H}_\alpha(\boldsymbol{q}_B)^T\boldsymbol{f}_B \tag{4.8}$$

il est possible de lier les forces dans l'espace des contacts $\boldsymbol{f}$ et dans l'espace des mouvements $\boldsymbol{r}$.

Dans la pratique, l'opérateur $\mathbb{H}_\alpha$ est souvent un interpolateur barycentrique, c'est à dire qu'une distance d'interpénétration $\boldsymbol{\delta}_\alpha$ est distribuée à l'ensemble des sommets du triangle où le contact à lieu en utilisant comme poids les coordonnées barycentriques du point de contact dans le triangle.

Par assemblage, la matrice $\mathbb{H}$ permettant le passage de l'espace des contacts à celui des déplacements pour tous les points de contact peut être construite.

Les méthodes par pénalités

Principes Cette méthode est une technique numérique fréquemment utilisée qui permet d'observer certaines contraintes.

Dans le cas d'un contrainte de non-interpénétration de deux objets, il a été proposé d'utiliser des ressorts associés à chaque contact. Cela revient a *pénaliser* le loi de SIGNORINI.Voir figure 4.10. Ainsi, lorsque deux objet s'intersectent, les ressorts génèrent des forces de contacts qui tendent à faire ressortir les deux objets du contact.

Cette méthode fut l'une des premières utilisées, et est connue sous le nom de MÉTHODE PAR PÉNALITÉS. Sa première application dans le cadre du COMPUTER GRAPHICS a été effectuée par MOORE *et al.* dans [MW88]. La raideur des ressorts qu'ils utilisent est proportionelle $\frac{1}{d}$, où d est la distance entre les objets. Plus les objets sont près, plus la raideur du ressort est importante. Autre particularité de leur méthode, ils distinguent le cas où les deux objets entre en collision du cas où les deux objets sortent de collision. A chaque cas est associée un raideur différente.

Cette méthode a été reprise dans de nombreux articles, pour des objets déformables volumiques [JP04, MAC04, HFS+01] aussi bien que surfaciques [BFA02].

Discussions L'essor des MÉTHODES PAR PÉNALITÉS découle de leur simplicité de mise en œuvre, de leur efficacité ainsi que de leur généricité. Elles permettent en un minimum d'effort d'ajouter une gestion des contacts rapide et adaptée à de nombreux modèles, des corps rigides aux corps déformables de tout type, aussi bien volumiques que surfaciques.

Cependant, elles souffrent de nombreuses limitations. Premièrement, la détermination de la raideur utilisée est délicate. Sa valeur dépend de nombreux paramètres de la simulation, et ne peut être calculée précisément. Il faut établir des heuristiques plus ou moins efficaces pour en fixer la valeur et assurer que visuellement il n'y aura pas d'interpénétration.

Ensuite, même si cette raideur est bien choisie, il n'est en aucun cas possible de garantir le respect de la loi de SIGNORINI. Les MÉTHODES PAR PÉNALITÉS permettent seulement de réduire les interpénétrations, pas de les annuler.

Dans le même esprit, ces interpénétrations ne peuvent être réduites à loisir en augmentant la raideur, car plus cette dernière augmente, plus le système à résoudre présente un mauvais conditionnement, et, dans le cas où un solveur explicite est utilisé, plus il faut réduire le pas de temps δt. Cela pose donc un problème de stabilité.

Enfin, ce type de méthode n'est pas physique. Le couplage mécanique entre les différents points de contacts n'est pas pris en compte, et il n'est pas possible de gérer le frottement.

Les méthodes par impulsions

Principes Ces méthodes ont été introduites par MIRTICH et CANNY dans [MC95]. Initialement développées pour la simulation de corps rigides, elles ont été adaptées au déformable. Dans ces méthodes, lorsque deux objets entrent en collision, ils sont chacun soumis à une impulsion de valeur opposée de manière à éviter l'interpénétration. Ces méthodes prennent un parti différent des MÉTHODES PAR PÉNALITÉS pour rendre compte des contacts : elles sont basées sur l'observation que dans la réalité, il n'y a ni contrainte, ni force de rappel qui empêchent l'interpénétration. C'est la collision, le choc entre deux objets, qui altère leurs vitesses et évite de ce fait l'interpénétration.

Les MÉTHODES PAR IMPULSIONS suivent donc ce précepte pour éviter les interpénétrations. Ainsi, au lieu d'appliquer une force pour accélérer les points en interpénétration, une impulsion

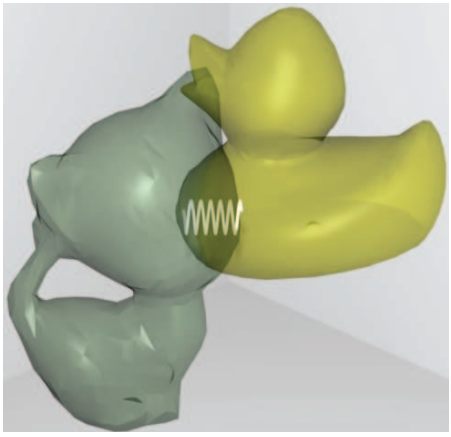


FIG. 4.10 – Méthode par pénalités. Un ressort est associé à chaque point de contact.

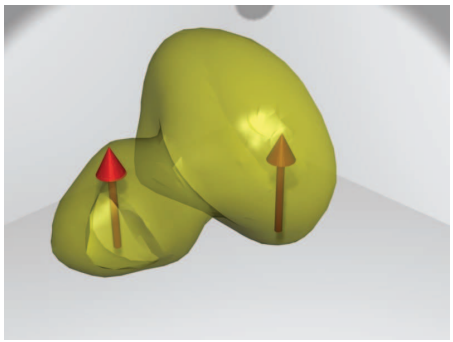


FIG. 4.11 – Couplage mécanique entres les différents points de contacts.

$\boldsymbol{p}$ est appliquée pour faire varier directement la vitesse. Une impulsion est définie comme étant :

$$\boldsymbol{p} = \int_0^t f(\tau) d\tau \tag{4.9}$$

Lorsqu'une impulsion est appliquée à un point matériel, elle modifie son moment, d'où :

$$\Delta \boldsymbol{v} = \boldsymbol{M}^{-1} \boldsymbol{p} \tag{4.10}$$

Toute la difficulté des méthodes par impulsions réside alors dans le choix de l'impulsion à appliquer pour obtenir la modification de vitesse voulue. MIRTICH détaille dans [MC95] les calculs dans le cas d'un contact frottant, collant, ou même pour simuler des articulations. Dans [BFA02, WT06] les auteurs proposent d'annuler simplement la composante normale de la vitesse au contact en appliquant une impulsion $\frac{\boldsymbol{M}\boldsymbol{v}_N}{2}$, où $\boldsymbol{v}_N$ est la composante normale de la vitesse. Il est aussi possible tout simplement de changer directement la vitesse d'un point sans se préoccuper de l'impulsion requise, comme le font CORDIER *et al.* dans [CMT02].

Discussion Ce type de méthode solutionne de nombreux problèmes des MÉTHODES PAR PÉNALITÉS. Notamment elles n'affectent pas le conditionnement des systèmes simulés, ce qui n'impose pas de restriction sur les pas de temps utilisés ou la stabilité du système. De plus, elles garantissent la non interpénétration des objets et autorisent la simulation du frottement en imposant des impulsions dans les directions tangentes à la normale au contact.

Par contre, elles gèrent mal les contacts multiples et continus, et il est impossible de maintenir un objet en adhérence avec du frottement. Il aura toujours tendance à se déplacer sous l'effet des impulsions.

À noter que dans [Ben07], BENDER montre que les systèmes à résoudre pour imposer des impulsions peut prendre la même forme que les systèmes obtenus pour déterminer des multiplicateurs de Lagrange.

Les méthodes par contraintes

Les méthodes présentées jusqu'à présent souffrent d'un mal commun, en plus de leurs limitations particulières : elles ne rendent pas compte du couplage mécanique entre les différents point de contacts. En effet, chacun des points de contact est traité par ces méthodes indépendamment des autres. Or, comme l'illustre la figure 4.11, les différents contacts dépendent étroitement les uns des autres, et la résolution de l'un d'entre eux peut résoudre une partie de autres.

Cette prise en compte est d'autant plus importante lorsque l'intégration temporelle se fait à l'aide d'une méthode implicite, dans la mesure où le couplage prend alors en compte non seulement la masse mais aussi la raideur et l'amortissement.

Il existe une classe de méthodes, les MÉTHODES PAR CONTRAINTES, qui permettent de pallier à ce manquement. Elles reposent sur la définition sous forme d'un jeu de contraintes des contacts, et résolvent ces contraintes en tenant compte des couplages mécaniques entre ces contacts à travers l'utilisation des matrices de masse $\boldsymbol{M}$, raideur $\boldsymbol{K}$ et amortissement $\boldsymbol{B}$. Les contacts sont alors résolus exactement, c'est à dire qu'il n'y a plus d'interpénétrations après la résolution, et la solution tient compte du couplage mécanique.

Deux méthodes sont régulièrement mises en œuvre pour résoudre ces contraintes de non-interpénétration : les formulation par MULTIPLICATEURS DE LA LAGRANGE et par LINEAR COMPLEMENTARY PROBLEM. La première est présentée ci-dessous, tandis que la seconde fait l'objet d'une section propre 4.3.

Résolution par Multiplicateurs de Lagrange L'importance des MULTIPLICATEURS DE LAGRANGE a été soulignée par BARAFF dans [Bar96]. L'idée sur laquelle les MULTIPLICATEURS DE LAGRANGE reposent est de forcer le respect des contraintes à l'aide de forces $\hat{\boldsymbol{f}}$ qui ne créent pas de travail. Ce type de contraintes, qui ne créent pas de travail, sont connues sous le nom de contraintes holonomes.

La démarche suivie pour calculer ces forces, ou MULTIPLICATEURS DE LAGRANGE est donnée dans l'annexe A.1.

Dans la pratique, il s'agit de résoudre le système :

$$\left(\begin{array}{cc} \boldsymbol{M} & -\boldsymbol{C}^T \\ \boldsymbol{C} & 0 \end{array}\right) \left| \begin{array}{c} \dot{\boldsymbol{x}} \\ \boldsymbol{\lambda} \end{array} \right. = \left| \begin{array}{c} \boldsymbol{f} \\ -\dot{\boldsymbol{C}}\dot{\boldsymbol{x}} \end{array} \right. \tag{4.11}$$

Où $\boldsymbol{M}\ddot{\boldsymbol{x}} = \boldsymbol{f}$ décrit le comportement du système, et $\boldsymbol{C}(\boldsymbol{x}) = 0$ impose les contraintes à respecter.

Le problème d'une résolution par MULTIPLICATEURS DE LAGRANGE est qu'elle ne permet de résoudre que des contraintes holonomes, or les lois de SIGNORINI et COULOMB ne sont pas holonomes. On obtient en effet des contraintes bilatérales, alors que les contraintes imposées par les lois de contact et frottements sont unilatérales par essence. Ces deux raisons conduisent à une résolution approximative des contacts et frottements par une résolution directe des MULTIPLICATEURS DE LAGRANGE.

Pour assurer une résolution qui puisse prende en compte des contraintes unilatérales, il est possible d'uiliser des algorithmes de type QUADRATIC PROGRAMMING, QP. L'inconvénient de ces méthodes réside dans leur lenteur.

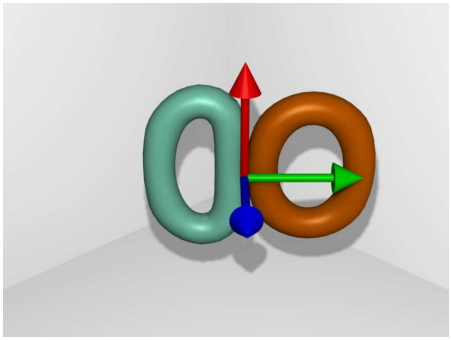


FIG. 4.12 – Le trièdre associé à un point de contact.

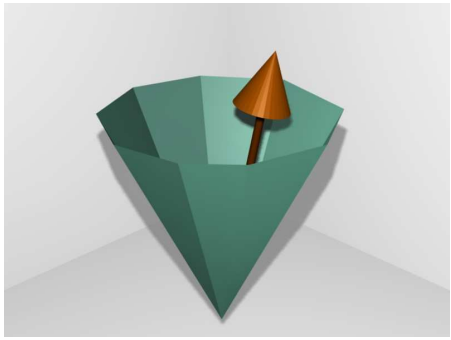


FIG. 4.13 – Cône de frottement linéarisé.

4.3 Gestion des contact par LCP

Toutes les méthodes que nous avons survolées jusqu'à présent ne satisfont pas toutes nos exigences concernant la vraisemblance de la simulation. Nous présentons ici une méthode qui permet de résoudre finement les contacts et le frottement, en modélisant précisément les lois de SIGNORINI et COULOMB.

Pour cela, ces lois seront formulées sous la forme d'un PROBLÈME LINÉAIRE DE COMPLÉMENTARITÉ, ou LCP.

4.3.1 Linear Complementary Problem

La forme canonique d'un LCP, telle qu'elle est définie par MURTY dans [Mur97] est :

$$\boldsymbol{w} - \boldsymbol{A}\boldsymbol{z} = \boldsymbol{q} \tag{4.12}$$

$$\boldsymbol{w} > 0 \perp \boldsymbol{z} > 0 \tag{4.13}$$

$\boldsymbol{w}$ et $\boldsymbol{z}$ sont les inconnues du système, tandis que $\boldsymbol{A}$ est une matrice carrée et $\boldsymbol{q}$ est un vecteur donné. L'équation 4.13 impose l'orthogonalité entre les contraintes $\boldsymbol{w} > 0$ et $\boldsymbol{z} > 0$, c'est à dire qu'à une w_i positif doit correspondre un z_i nul, et inversement à un w_i nul doit correspondre un z_i positif. Ce type de contrainte n'est pas sans rappeler la loi de SIGNORINI.

Les LCP ont une interprétation géométrique dont le détail peut être trouvé dans [Mur97]. Diverses méthodes, directes ou itératives sont envisageables pour les résoudre. Pour nos applications, nous utilisons un solveur itératif de type GAUSS-SEIDEL.

4.3.2 Compliance

Description

Avec cette résolution des contacts par LCP nous cherchons à prendre en compte les couplages entre les différents points de contact.

Nous cherchons donc à savoir quel est le déplacement induit par l'application d'une force en un noeud particulier. Formulée ainsi, cette information semble être l'inverse de celle obtenue par la

raideur du système, qui donne pour un déplacement donné, les forces générées. Et effectivement, la matrice permettant de connaître ces informations, nommée compliance $\boldsymbol{C}$, est dans le cas statique l'inverse de la matrice de raideur, et dans le cas de la dynamique :

$$\boldsymbol{C} = (\frac{\boldsymbol{M}}{h^2} + \frac{1}{h}\frac{\partial \boldsymbol{F}_{int}}{\partial \dot{\boldsymbol{x}}} + \frac{\partial \boldsymbol{F}_{int}}{\partial \boldsymbol{x}})^{-1} \tag{4.14}$$

Il faut noter que cette matrice $\boldsymbol{C}$ est tirée du membre gauche de l'équation 3.7, qu'elle dépend donc du schéma d'intégration utilisé. De plus, elle a été rendue homogène à l'inverse d'une raideur par la multiplication par $\frac{M}{h^2}$.

Dans le cas où la simulation est statique, $\boldsymbol{C}$ est simplement l'inverse de la matrice de raideur $\boldsymbol{K}$ du système :

$$\boldsymbol{C} = \boldsymbol{K}^{-1} \tag{4.15}$$

Construction

Le problème, avec la compliance, c'est qu'elle est construite en inversant une matrice. Or nous verrons dans la suite que cette inversion doit être faite à chaque pas de temps de simulation. Sa construction est donc une opération particulièrement coûteuse. Heureusement, dans la pratique, il s'avère que toute les colonnes de cette matrice ne sont pas utilisées. Seuls les colonnes associées avec des points de contact doivent être connues.

Ainsi, une première optimisation consiste à ne calculer que les colonnes requises. Pour cela, il suffit de remarquer que chaque colonne $\boldsymbol{c}_i$ de cette matrice de compliance est en fait le déplacement de chacun des degrés de liberté dû à l'application d'une force unitaire $\boldsymbol{f}_i$ sur le degré de liberté i. Ainsi $\boldsymbol{c}_i$ satisfait cette équation :

$$(\frac{\boldsymbol{M}}{h^2} + \frac{1}{h}\frac{\partial \boldsymbol{F}_{int}}{\partial \dot{\boldsymbol{x}}} + \frac{\partial \boldsymbol{F}_{int}}{\partial \boldsymbol{x}})\boldsymbol{c}_i = \boldsymbol{I}_i \tag{4.16}$$

où $\boldsymbol{I}_i$ vaut 0 partout sauf en i où il vaut 1.

À noter que pour chaque point de contact i il faut trouver trois $\boldsymbol{c}_{i,x}, \boldsymbol{c}_{i,y}, \boldsymbol{c}_{i,z}$, une pour chaque direction de l'espace. Cette optimisation permet d'accélérer considérablement la résolution des contacts, dans la mesure où la matrice $(\frac{\boldsymbol{M}}{h^2} + \frac{1}{h}\frac{\partial \boldsymbol{F}_{int}}{\partial \dot{\boldsymbol{x}}} + \frac{\partial \boldsymbol{F}_{int}}{\partial \boldsymbol{x}})$ est particulièrement creuse et qu'il est possible d'utiliser un solveur itératif rapide pour résoudre l'équation 4.16.

4.3.3 Application au contact et au frottement

La construction d'un LCP pour la loi de SIGNORINI est immédiate. L'inconnue $\boldsymbol{w}$ est en fait le vecteur des distances d'interpénétration $\boldsymbol{\delta}_n$ suivant la normale au contact $\boldsymbol{n}$, tandis que $\boldsymbol{z}$ est le vecteur des forces normales au contact $\boldsymbol{f}_n$. $\boldsymbol{q}$ est le vecteur des distances d'interpénétration $\boldsymbol{\delta}^{free}$ suivant la normale obtenu après le mouvement libre. Enfin, la matrice $\boldsymbol{A}$ est une matrice $\sum_o \mathbb{H}\boldsymbol{c}_{ii}^o\mathbb{H}^T$ qui devra relier les forces aux déplacements induits par l'application de ces forces. La somme sur les objets o en contact assure que la compliance de chaque objet impliqué dans ce contact est prise en compte. D'où, dans l'espace des contacts :

Ce qui donne pour chacune des forces de contact $\boldsymbol{f}_i$ issues de l'espace des mouvements :

$$\left\{ \begin{array}{ccc} \boldsymbol{n}_i^T(\sum_o \mathbb{H}_i\boldsymbol{c}_{ii}^o\mathbb{H}_i^T)\boldsymbol{n}_i f_{i,n} + \delta_{i,n}^{free} & = & \delta_{i,n} \\ f_{i,n} > 0 & \perp & \delta_{i,n} > 0 \end{array} \right. \tag{4.17}$$

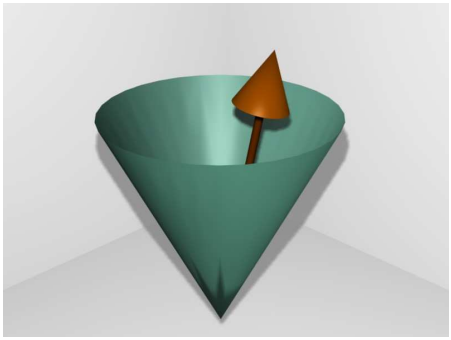


FIG. 4.14 – Cône de frottement.

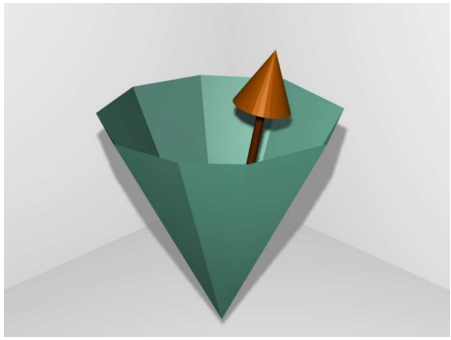


FIG. 4.15 – Cône de frottement linéarisé.

où la matrice $\mathbb{H}$ est l'opérateur permettant de passer de l'espace des mouvements à celui des contacts, comme présenté dans la section 4.2.2. $f_{i,n}$ est la norme de la composante normale, i.e. $\boldsymbol{f}_i = f_{i,n}\boldsymbol{n}_i + f_{i,t}\boldsymbol{t}_i + f_{i,s}\boldsymbol{s}_i$. Voir la figure 4.12.

La matrice $\sum_o \mathbb{H}\boldsymbol{c}^o\mathbb{H}^T$ est nommée opérateur de DELASSUS et les sous-matrices $\boldsymbol{c}^o_{ii}$ qu'elle utilise sont issues de la compliance $\boldsymbol{C}^o$.

La gestion du frottement se fait aussi à l'aide d'un LCP. La loi de COULOMB présentée en 4.2.2 stipule que pour une point de contact, il y a adhérence si la force au contact est dans le cône, et glissement avec frottement si la force de frottement est sur le cône. Soit $\boldsymbol{f}_c$ la force au contact, $\boldsymbol{f}_n$ sa composante normale, $\boldsymbol{f}_t$ sa composante tangentielle, et μ le coefficient de frottement, alors :
- si $||\boldsymbol{f}_t|| < \mu||\boldsymbol{f}_n||$, il y a adhérence.
- si $||\boldsymbol{f}_t|| = \mu||\boldsymbol{f}_n||$, il y a glissement avec frottement.

Le problème de cette loi tient en ce qu'elle utilise un cône, qui est une primitive non-linéaire, ainsi qu'une contrainte sur $||\boldsymbol{f}_t||$ et $||\boldsymbol{f}_n||$ qui, faisant appel à la racine carrée, et à la fonction carré, est non-linéaire.

ANITESCU et al. dans [APS98] proposent donc de linéariser cette loi, tout d'abord en facettisant le cône de frottement comme sur la figure 4.15, et en décrivant l'espace tangent comme un espace vectoriel engendré par ces directions d_i définies comme sur la figure 4.15.

Le LCP alors obtenu dans l'espace des contacts est pour un point i :

$$\begin{cases} \boldsymbol{n}_i^T(\sum_o \mathbb{H}_i\boldsymbol{c}^o_{ii}\mathbb{H}_i^T)\boldsymbol{n}_i f_{i,n} + \delta^{free}_{i,n} & = & \delta_{i,n} \\ 0 < f_{i,n} & \perp & \delta_{i,n} > 0 \\ 0 < \lambda + \delta_{i,t_1} & \perp & f_{i,t_1} > 0 \\ \cdots & & \\ 0 < \lambda + \delta_{i,t_r} & \perp & f_{i,t_r} > 0 \\ 0 < \mu f_{i,n} + \sum_{k=1}^r f_{i,t_k} & \perp & \lambda > 0 \end{cases} \tag{4.18}$$

$$\begin{cases} \boldsymbol{p}_i^T(\sum_o \mathbb{H}_i\boldsymbol{c}^o_{ii}\mathbb{H}_i^T)\boldsymbol{p}_i f_{i,p} + \delta^{free}_{i,n} & = & \delta_{i,p}, \boldsymbol{p} \in \{\boldsymbol{n}, \boldsymbol{t}_1, \cdots, \boldsymbol{t}_k\} \\ 0 < f_{i,n} & \perp & \delta_{i,n} > 0 \\ 0 < \lambda + \delta_{i,t_1} & \perp & f_{i,t_1} > 0 \\ \cdots & & \\ 0 < \lambda + \delta_{i,t_r} & \perp & f_{i,t_r} > 0 \\ 0 < \mu f_{i,n} + \sum_{k=1}^r f_{i,t_k} & \perp & \lambda > 0 \end{cases} \tag{4.19}$$

Les deux premières lignes garantissent le respect de la loi de SIGNORINI. Les r lignes sui-

vantes assurent que s'il y a une force de frottement tangentielle f_{t_i}, alors cette dernière s'oppose impérativement au déplacement δ_{t_i}. Enfin, la dernière assure la loi de COULOMB proprmement dite, c'est à dire soit il y a déplacement et la force tangente est précisément $\boldsymbol{f}_t = \mu \boldsymbol{f}_n$, soit il n'y a pas déplacement, et la force tangentielle respecte $\boldsymbol{f}_t < \mu \boldsymbol{f}_n$.

En utilisant la compliance c_{ii} du point i, ce système peut être réécrit de manière à ce que les inconnues soient les composantes normales et tangentielles de la force de contact :

$$\left\{\begin{array}{ccc} \boldsymbol{p}_i^T(\sum_o \mathbb{H}_i \boldsymbol{c}_{ii}^o \mathbb{H}_i^T)\boldsymbol{p}_i f_{i,p} + \delta_{i,n}^{free} & = & \delta_{i,p}, \boldsymbol{p} \in \{\boldsymbol{n}, \boldsymbol{t}_1, \cdots, \boldsymbol{t}_k\} \\ 0 < f_{i,n} & \perp & \delta_{i,n} > 0 \\ 0 < \lambda + \boldsymbol{t}_{i,1}^T \sum_o (\mathbb{H}_i \boldsymbol{c}_{ii}^o \mathbb{H}_i^T)\boldsymbol{t}_{i,1} f_{i,t_1} & \perp & f_{i,t_1} > 0 \\ \cdots & & \\ 0 < \lambda + \boldsymbol{t}_{i,r}^T \sum_o (\mathbb{H}_i \boldsymbol{c}_{ii}^o \mathbb{H}_i^T)\boldsymbol{t}_{i,r} f_{i,t_r} & \perp & f_{i,t_r} > 0 \\ 0 < \mu f_n + \sum_{k=1}^r f_{t_k} & \perp & \lambda > 0 \end{array}\right. \tag{4.20}$$

4.3.4 Résolution par GAUSS-SEIDEL

Il existe plusieurs manières de résoudre un problème de complémentarité. DURIEZ a montré dans [DDKA06] qu'une résolution basée sur un GAUSS-SEIDEL permettait une résolution temps réel, aussi nous allons détailler cette méthode pour une application donnée.

Le principe d'une résolution de l'équation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ par GAUSS-SEIDEL consiste en l'amélioration progressive d'une solution $\tilde{x}$ en corrigeant successivement chacun des éléments $\tilde{x}_i$ de la solution à condition que la matrice $\boldsymbol{A}$ soit à diagonale strictement dominante. Cette amélioration itérative est basée sur le fait que si $\boldsymbol{x}$ est la solution de $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, alors :

$$x_i = \left(b_i - (\sum_{k=1}^{i-1} a_{ij} x_j + \sum_{k=i-1}^{n} a_{ij} x_j) \right) / a_{ii} \tag{4.21}$$

Il parait alors possible, qu'en appliquant itérativement ce calcul à tous les $\tilde{x}_i$ il soit possible de converger vers la solution $\boldsymbol{x}$. Cette propriété a été prouvée, et le schéma itératif 12, dit de GAUSS-SEIDEL a été construit :

Algorithm 12 Résolution itérative par GAUSS-SEIDEL

1: **while** $\sqrt{\boldsymbol{r}^T\boldsymbol{r}} > \epsilon$ **do**
2: **for** $1 <= i <= n$ **do**
3: $x_i^{k+1} = \left(b_i - (\sum_{j=1}^{i-1} a_{ij} x_j^{k+1} + \sum_{j=i-1}^{n} a_{ij} x_j^k) \right) / a_{ii}$
4: **end for**
5: $\boldsymbol{r} = \boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{b}$
6: $k = k + 1$
7: **end while**

L'avantage de cette méthode, et des méthodes itératives en général, est quelle est bien adaptée aux matrices creuses, non seulement parce qu'elle nécessite d'autant peu de calculs que la matrice est creuse, mais aussi parce que la forme même de son algorithme la rend bien adaptée au format de stockage creux. Elle présente l'intérêt de converger aussi, au moins localement, quand le système est sur-contraint et qu'il existe plusieurs solutions. L'utilisation de la cohérence

temporelle permet aussi de partir d'une solution proche pour converger plus rapidement, ce qui compense la convergence peu efficace des algorithmes par relaxation.

De plus, elle peut être facilement adaptée à la résolution de problème de complémentarité. Il suffit pour cela de tester à chaque pas de temps le respect des contraintes. Si une contrainte est violée, alors il suffit de forcer la valeur de l'inconnue associée de manière à en assurer le respect.

Ainsi, l'algorithme pour la résolution des contacts, qui peut d'ailleurs être utilisé pour toute contrainte de type complémentarité est donné par l'algorithme 13. Les sous-matrices $\tilde{\boldsymbol{w}}_{ij}$ de DELSASSUS liant les forces appliquées en un point quelconque j avec les déplacements du point quelconque i sont construites de manière à donner les déplacements suivant les directions normales et tangentes $\boldsymbol{n}_i$, $\boldsymbol{t}_i$, $\boldsymbol{s}_i$, au point i et ce pour des forces exprimées suivant le repère $\boldsymbol{n}_j$, $\boldsymbol{t}_j$, $\boldsymbol{s}_j$ au point d'application j. C'est à dire :

$$\tilde{\boldsymbol{w}}_{ij} = \begin{bmatrix} \boldsymbol{n}_i \\ \boldsymbol{t}_i \\ \boldsymbol{s}_i \end{bmatrix} \left(\sum_o \mathbb{H}_i \boldsymbol{c}_{ij}^o \mathbb{H}_j^T\right) \begin{bmatrix} \Delta\boldsymbol{n}_j & \boldsymbol{t}_j & \boldsymbol{s}_j \end{bmatrix} \tag{4.22}$$

Algorithm 13 Résolution du contact et des frottements par GAUSS-SEIDEL

while $\sum_{i=0}^{n} \frac{||\delta_i^{k+1} - \delta_i^k||}{||\delta_i^{k+1}||} > \epsilon_1$ **do**
  for $1 \leq i \leq n$ **do**
    $\boldsymbol{\delta}_i^{k+1} = \left(\boldsymbol{\delta}_i^{free} - (\sum_{j=1}^{i-1} \tilde{\boldsymbol{w}}_{ij}\boldsymbol{f}_j^{k+1} + \sum_{j=i-1}^{n} \tilde{\boldsymbol{w}}_{ij}\boldsymbol{f}_j^k)\right)$
    $\boldsymbol{\delta}_i = \tilde{\boldsymbol{w}}_{ii}\boldsymbol{f}_i^k$
    $f_{i,n}^{k+1} = f_{i,n}^k - [1\ 0\ 0]\tilde{\boldsymbol{w}}_{ii}^{-1}\boldsymbol{\delta}_i$
    if $f_{i,n}^{k+1} < 0$ **then**
      $\boldsymbol{f}_i^{k+1} = 0$
    else
      $f_{i,t}^{k+1} = f_{i,t}^k - [0\ 1\ 0]\tilde{\boldsymbol{w}}_{ii}^{-1}\boldsymbol{\delta}_i$
      $f_{i,s}^{k+1} = f_{i,s}^k - [0\ 0\ 1]\tilde{\boldsymbol{w}}_{ii}^{-1}\boldsymbol{\delta}_i$
      if $\sqrt{(f_{i,t}^{k+1})^2 + (f_{i,s}^{k+1})^2} > \mu f_{i,n}^{k+1}$ **then**
        $f_{i,t}^{k+1}* = \frac{f_{i,n}^{k+1}}{\sqrt{(f_{i,t}^{k+1})^2 + (f_{i,s}^{k+1})^2}}$
        $f_{i,s}^{k+1}* = \frac{f_{i,n}^{k+1}}{\sqrt{(f_{i,t}^{k+1})^2 + (f_{i,s}^{k+1})^2}}$
      end if·
      $\boldsymbol{\delta}_i^{k+1} = \tilde{\boldsymbol{w}}_{ii}\boldsymbol{f}_i^k$
    end if·
  end for
  $k = k + 1$
end while

Les forces $\boldsymbol{f}_j$ sont données dans le repère $\boldsymbol{n}_j$, $\boldsymbol{t}_j$, $\boldsymbol{s}_j$ tandis que les $\boldsymbol{\delta}_i$ sont donnés dans $\boldsymbol{n}_i$, $\boldsymbol{t}_i$, $\boldsymbol{s}_i$

Cet algorithme peut bien sûr être optimisé. Les matrices w_{ii}^{-1} gagnent à être précalculées dans une phase d'initialisation, mais il est même possible d'aller encore plus loin comme le proposent

DURIEZ *et al.*. Ils montrent que la matrice diagonale :

$$\begin{bmatrix} w_{nn} & 0 & 0 \\ 0 & \frac{\lambda_1+\lambda_2}{2} & 0 \\ 0 & 0 & \frac{\lambda_1+\lambda_2}{2} \end{bmatrix} \tag{4.23}$$

est une bonne approximation de $\boldsymbol{w}_{ii}$. $\lambda_{1,2}$ sont les valeurs propres minimale et maximale de $\boldsymbol{w}_{ii}$. Le calcul de l'inverse est alors immédiat.

De même, chaque point de contact peut se voir associer un état actif ou non, *i.e.* le contact est actif si la force de contact associéee est positive. Ainsi, dans la phase de calcul de $\boldsymbol{\delta}_i^{k+1}$, seuls les contacts actifs peuvent être pris en compte ce qui accélère le traitement.

4.3.5 Discussion

Cette manière de résoudre les contacts par complémentarité satisfait le mieux nos exigences : elle garantit le respect des lois de SIGNORINI et COULOMB, et pour peu qu'une valeur initiale correcte soit fournie, la convergence se fait en quelques itérations. Cette méthode garantit donc une simulation physique et une résolution rapide.

Le problème, c'est que cette méthode repose sur une phase de précalcul visant à déterminer la valeur de la compliance $\boldsymbol{C}$, or le calcul de cette compliance est très coûteux, même si seules les colonnes nécessaires sont calculées comme présenté en 4.3.2.

Cependant, en dépit de cette accélération, le calcul des $\boldsymbol{c}_i$ reste rédhibitoire. Cette méthode souffre en effet de deux limitations qui opèrent avant même de commencer la résolution par GAUSS-SEIDEL :

1. Le nombre de colonnes de $\boldsymbol{C}$ à calculer dépend du nombre de contacts multiplié par 3.·
2. Le calcul de chaque colonne dépends de la complexité du modèle. Plus ce dernier est complexe, plus la dimension de la matrice C est importante, et plus le calcul de chaque $\boldsymbol{c}_i$ est coûteux.

Nous allons donc présenter dans les sections suivantes une de nos contributions qui permet d'éviter cette phase de calcul des $\boldsymbol{c}_i$ et d'atteindre ainsi des temps interactifs même pour des modèles complexes ayant de nombreux points de contact.

4.4 Pré-conditionneur corotationnel nodal

Avant de de présenter notre méthode de gestion des contacts par COMPLIANCE WARPING, nous aimerions montrer comment nous avons abouti à cette idée, et surtout justifier sa validité. Pour cela, nous allons présenter le modèle corotationnel nodal, et montrer qu'il donne une bonne approximation du modèle corotationnel élémentaire. C'est de cette particularité dont nous tirerons parti dans la section 4.5 pour gérer efficacement les contacts.

4.4.1 Modèle corotationnel nodal

Nous avons décrit dans la section 2.5.4 les modèles corotationnels, et notamment les modèles corotationnels par élément, où l'abstraction des rotations se fait au niveau des éléments. Il existe

cependant une autre approche aux modèles corotationnels qui est basée sur l'abstraction des rotations aux noeuds, et qui fut d'ailleurs la première proposée par MÜLLER [MDM+02].

Principe Le but de cette méthode est de nouveau de s'abstraire des non-linéarités géométriques qui induisent des déformations aberrantes lorsque le tenseur de CAUCHY est utilisé. Pour ce faire, MÜLLER *et al.* proposent de calculer les forces en estimant une rotation au niveau de chaque nœud i du maillage de la manière suivante :

$$\boldsymbol{f}_i = \boldsymbol{R}_i \sum_{j=1}^{n} k_{ij}(\boldsymbol{R}_i^T \boldsymbol{x}_j - \boldsymbol{x}_{0j}) \tag{4.24}$$

Ils notent aussi qu'il est possible d'utiliser la méthode suivante :

$$\boldsymbol{f}_i = \boldsymbol{R}_i \sum_{j=1}^{n} k_{ij}(\boldsymbol{R}_j^T \boldsymbol{x}_j - \boldsymbol{x}_{0j}) \tag{4.25}$$

Ce dernier schéma de calcul des forces internes est très intéressant, car il permet de construire la matrice de raideur globale comme une rotation de la matrice de raideur initiale :

$$\tilde{\boldsymbol{K}} = \boldsymbol{R}\boldsymbol{K}\boldsymbol{R}^T \tag{4.26}$$

dont l'inverse peut être calculée très facilement :

$$\tilde{\boldsymbol{K}}^{-1} = \boldsymbol{R}\boldsymbol{K}^{-1}\boldsymbol{R}^T \tag{4.27}$$

si K^{-1} est précalculée.

Cette propriété est très intéressante, car l'inverse de la matrice $\tilde{\boldsymbol{K}}$ peut être utilisée pour calculer les déplacements connaissant les forces appliquées sur l'objet. Dans la pratique cependant, $\boldsymbol{K}^{-1}$ étant pleine, et $\boldsymbol{K}$ étant creuse, il est plus efficace de calculer le déplacement à l'aide d'un solveur itératif.

Évaluation des rotations aux nœuds L'évaluation des rotations aux nœuds peut être faite de diverses manières. Un trièdre peut être associé à chaque nœud. La rotation au nœud est alors la rotation permettant de passer du trièdre dans le repère courant au repère initial.

Dans le cas d'un maillage tétraèdrique, la rotation peut-être estimée à l'aide des rotation des tétraèdres contenant le nœud. Ce type d'estimation est particulièrement adaptée aux modèles corotationnels.

Selon notre expérience, et dans le cadre de l'utilisation de ce modèle comme pré-conditionneur, il semble que les meilleurs résultats soient obtenus en utilisant la même méthode que pour le modèle élémentaire, voire même les mêmes valeurs de rotation. Cela assure une grande similitude entre les deux modèles.

Limitations Ce type de modèle souffre d'une limitation majeure, qui a conduit à son abandon au profit des modèles corotationnels par éléments. Il fait apparaître des forces fantômes qui entraînent une déformation peu naturelle dans le cas d'objet encastrés, et qui peut faire dévier de leur trajectoire normales des objets libres.

4.4.2 Utilisation du corotationnel nodal comme pré-conditionneur

Principe

Nous avons déjà vu en détail l'intérêt des pré-conditionneurs dans la section 3.3.2. Ces derniers visent à accroître le taux de convergence des solveurs itératifs en améliorant le conditionnement du problème.

De nombreux pré-conditionneurs existent, et il n'existe aucune contrainte pour leur construction. Les seuls impératifs sont que leur inverse soit une bonne approximation de l'inverse du système à résoudre, de telle sorte que l'application de ce pré-conditionneur réduise le conditionnement, et que leur application soit rapide.

Or, il s'avère que la matrice tangente au système non linéaire à résoudre à chaque pas de temps :

$$\boldsymbol{A} = (\boldsymbol{I} + h\boldsymbol{M}^{-1}\frac{\partial \boldsymbol{F}_{int}}{\partial \dot{\boldsymbol{x}}} + h^2\boldsymbol{M}^{-1}\frac{\partial \boldsymbol{F}_{int}}{\partial \boldsymbol{x}}) \tag{4.28}$$

issue de l'équation 3.7, est bien approximée, lorsqu'un modèle corotationnel élementaire est utilisé, par la matrice suivante :

$$\boldsymbol{P} = \boldsymbol{R}\left[(\boldsymbol{I} + h\boldsymbol{M}^{-1}\frac{\partial \boldsymbol{F}_{int}}{\partial \dot{\boldsymbol{x}}}^{0} + h^2\boldsymbol{M}^{-1}\frac{\partial \boldsymbol{F}_{int}}{\partial \boldsymbol{x}}^{0})\right]\boldsymbol{R}^T \tag{4.29}$$

qui est construite sur le principe d'un modèle corotationnel nodal, et dont l'inverse peut être calculée rapidement, comme le montre l'équation 4.27, pour peu que sa valeur pour des rotations nulles ait été précalculée.

Résultats

Pour évaluer l'efficacité de notre pré-conditionneur, nous avons calculé le conditionnement du système avec et sans pré-conditionnement. La valeur ce de conditionnement est donnée par le rapport entre valeurs singulières minimales et valeurs singulières maximales, *i.e.* :

$$\kappa(\boldsymbol{A}) = \frac{\sigma_{max}(\boldsymbol{A})}{\sigma_{min}(\boldsymbol{A})}$$

La figure 4.16 compare les conditionnements obtenus avec le corotationnel nodal comme pré-conditionneur , avec la matrice initiale $\boldsymbol{A}_0$ comme pré-conditionneur ou encore sans pré-conditionneur. L'objet déformé est un tore modélisé par 1470 tétraèdres, et doté d'un module de Young de 10000 et d'un coefficient de Poisson de 0.4.

Il apparaît clairement que ce conditionnement est amélioré de façon substantielle par l'application du préconditionneur corotationnel, et ce quelque soit la déformation de l'objet. Les gains sont alors de l'ordre d'un facteur 20. Il est aussi intéressant de noter que le conditionnement s'améliore lorsque l'objet déformé reprend une forme proche de celle au repos, et ce même si son orientation diffère. Notamment, le plateau allant du pas de temps 60 à 70 correspond au moment où le tore glisse sans se déformer sur le tore fixe.

Néanmoins, le pré-conditionneur utilisé étant une matrice pleine, le coût de l'application de ce dernier le disqualifie pour une utilisation dans une simulation interactive si l'implémentation

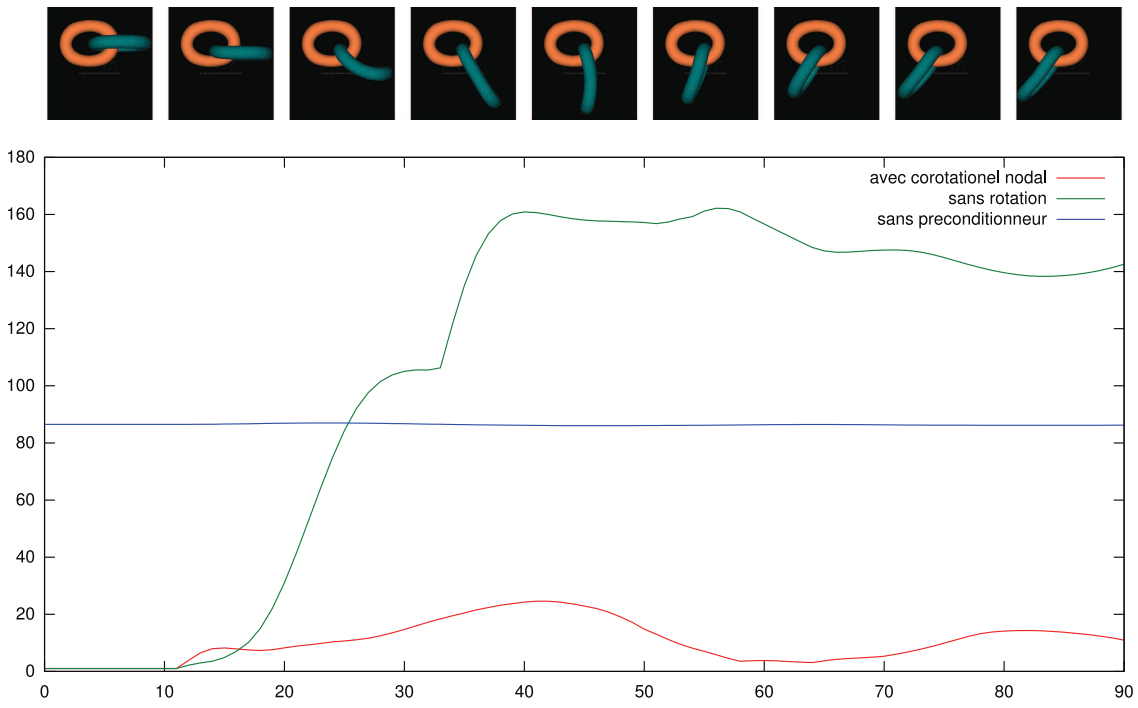


FIG. 4.16 – Le conditionnement du système suivant le pré-conditionneur utilisé.

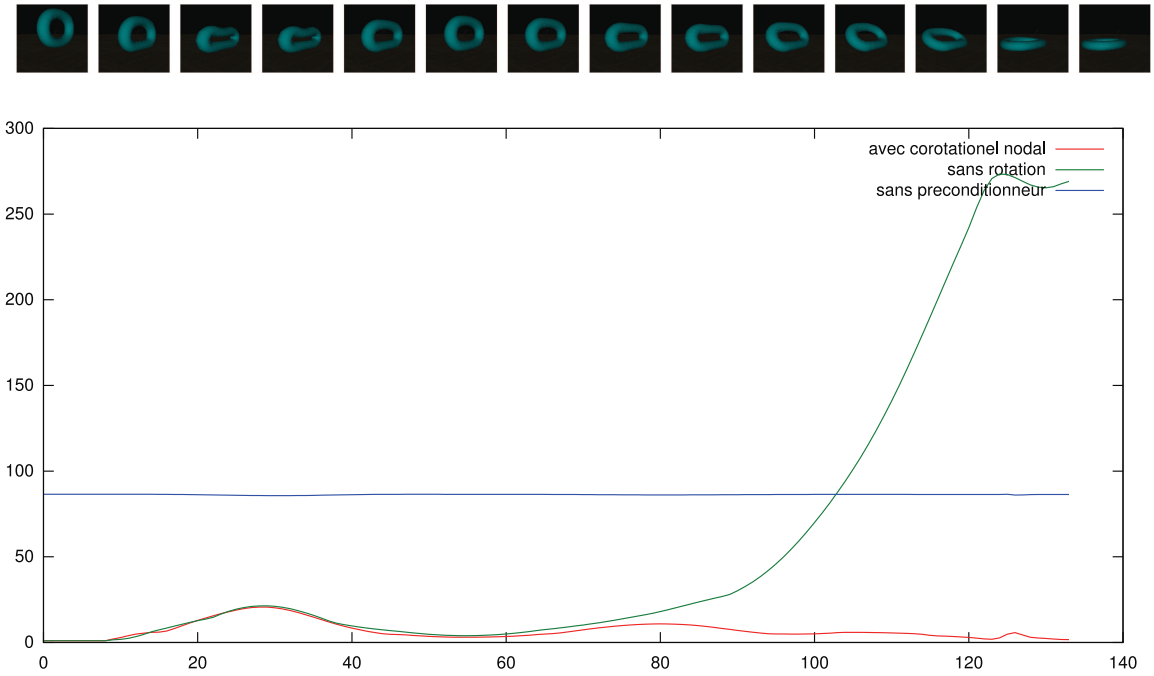


FIG. 4.17 – Le conditionnement du système suivant le pré-conditionneur utilisé.

est faite sur CPU. Une implémentation GPU, particulièrement adaptée aux calculs lourds et répétitifs devrait permettre un gain de temps, de même que l'utilisation d'un pré-conditionneur exprimé à une résolution inférieure dont l'interpolation à la résolution courante pourrait se faire avec des opérateurs de restriction et prolongation de type *ondelette*. Nous n'avons pas eu le temps d'implémenter cette méthode, mais c'est une perspective intéressante.

Il apparaît aussi que la simple matrice $\boldsymbol{A}_0$ à l'origine fournit assez naturellement un meilleur préconditionneur aux alentours de l'origine, mais devient rapidement inefficace puis néfaste. Dans la figure 4.17, le tore est lâché verticalement, et s'écrase sous son propre poids. Jusqu'au pas de temps 80, il ne subit que de faibles rotations locales, ce qui explique que $\boldsymbol{A}_0$ reste un bon préconditionneur. Après, par contre, le tore pivote et se couche sur le côté. $\boldsymbol{A}_0$ n'assure dès lors plus un bon conditionnement. Cela laisse à penser que s'il est possible de recalculer régulièrement et d'inverser cette matrice $\boldsymbol{A}_0$, même à une faible fréquence, alors il est possible aussi d'améliorer le conditionnement. Cependant, le pré-conditionneur nodal semble plus efficace et nécessite moins de calcul.

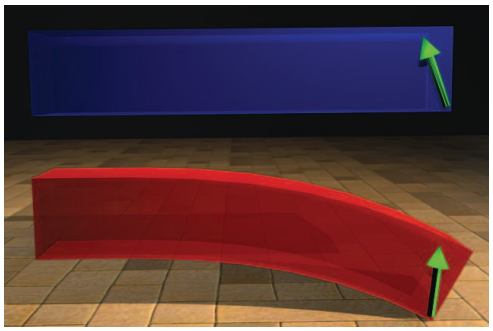


FIG. 4.18 – Les contacts sont ramenés à l'aide d'une rotation dans la configuration d'origine.

Conclusion

Tout cela tend à prouver qu'un modèle corotationnel nodal fournit une bonne approximation de la matrice tangente d'un modèle corotationnel élémentaire, puisque $\boldsymbol{P}$ est un bon préconditionneur. La gestion des contacts peut donc tirer parti de cette bonne approximation comme nous le montrons dans la suite avec notre résolution des contacts par COMPLIANCE WARPING.

4.5 Compliance Warping

Comme nous l'avons vu dans les sections précédentes, la méthode la plus adaptée à notre souci de réalisme physique est la méthode par contrainte. Cependant, cette méthode nécessite le calcul d'une partie de la matrice de compliance $\boldsymbol{C}$. Or le calcul des colonnes de $\boldsymbol{C}$ nécessaires à la résolution des contacts et des frottements est fonction du nombre de contacts ainsi que de la complexité du modèle simulé.

Cela implique que plus le nombre de points de contact s'accroît, et plus le modèle est complexe, plus la résolution des contacts devient coûteuse.

D'un autre côté, nous avons vu dans la section 4.4 que le modèle corotationnel nodal, facilement inversible, donnait une bonne approximation de la matrice tangente du modèle corotationnel élémentaire, or le calcul de la matrice de compliance s'appuie sur l'inverse de cette matrice tangente. Ce modèle nodal introduit par contre des forces fantômes lorsqu'il est utilisé pour calculer les forces internes. Il paraît donc séduisant d'utiliser ce modèle pour gérer les contacts, tout en conservant un modèle corotationnel élémentaire pour le calcul des déformations.

Ainsi, pour pallier aux limitations de la méthode classique de gestion des contacts par LCP, nous proposons le COMPLIANCE WARPING, qui va permettre de résoudre les contacts avec une méthode par contrainte en supprimant la phase de calcul de la compliance. Pour ce faire, nous construisons une approximation de la matrice de raideur en un temps indépendant du nombre de contacts et de la complexité du modèle utilisé. Cette méthode a fait l'objet d'une publication [SDCG08].

4.5.1 Approximation du modèle de contact

Le modèle corotationnel nodal que nous venons de présenter offre deux avantages :
– Son inverse est facilement calculable.

– C'est une bonne approximation du modèle corotationnel élémentaire que nous utilisons pour le calcul des déformations.

Ces deux propriétés en font un très bon candidat pour l'amélioration de la gestion des contacts par contrainte, et nous avons donc essayé d'en tirer parti dans notre méthode, le COMPLIANCE WARPING, qui est basée sur l'approximation du modèle de contact. C'est à dire qu'au lieu de calculer précisément la matrice $\boldsymbol{C}$, ou à *fortiori* les colonnes nécessaires, nous allons utiliser une approximation.

Approximation de la compliance

Notre approximation de la compliance est basée sur une approche similaire à celle des modèles corotationnel. C'est à dire que nous allons chercher à ramener les contacts de la configuration courante à la configuration initiale, non pas cette fois-ci pour éviter des non-linéarités, mais pour simplifier notre modèle et accélérer le calcul de le compliance. C'est exactement ce que fait le modèle corotationnel nodal.

La configuration initiale étant connue avant même la simulation, nous pouvons facilement en calculer la compliance hors-ligne. Ensuite, pour chacun des noeuds impliqués dans un contact, nous pouvons évaluer la rotation entre sa position courante et sa position initiale. Cette rotation est ensuite utilisée pour faire TOURNER le contact, c'est à dire sa normale, depuis l'orientation courante vers l'orientation d'origine.

Une fois les contacts ramenés dans la configuration d'origine, la compliance initiale pré-calculée peut être utilisée pour résoudre rapidement les contacts avec un GAUSS-SEIDEL. La compliance ne devant pas être recalculée, cette étape est rapide et indépendante de la complexité du modèle.

Enfin, les forces de contact ayant été déterminées dans la configuration d'origine, elles peuvent être ramenées à nouveau dans la configuration courante à l'aide des rotations inverses.

Il ne reste plus alors qu'à appliquer ces forces pour effectuer le mouvement contraint assurant le respect des lois de SIGNORINI et COULOMB. Nous détaillons cette étape dans la section 4.5.2.

Cette méthode revient donc à faire tourner la compliance en la pré- et post-multipliant par une matrice diagonale par blocs $\boldsymbol{R}$, dont les $\boldsymbol{r}_{ii}$ sont les matrices estimant la rotation au nœud i :

$$\boldsymbol{C} = \boldsymbol{R}\boldsymbol{C}^0\boldsymbol{R}^T \tag{4.30}$$

Justification de l'approximation

Notre approximation est donc basée sur un modèle corotationnel nodal, or nous avons établi que ce type de modèle était une bonne approximation du modèle corotationnel élémentaire.

De plus, même si nous faisons une approximation, nous conservons le couplage mécanique entre les divers points de contacts, ce qui n'est pas le cas des méthodes par pénalité ou par impulsions. Nous obtenons ainsi une simulation sensiblement plus réaliste.

Enfin, les lois de SIGNORINI et COULOMB sont forcément respectées pour peu que le solveur de contacts trouve une solution et que la même compliance soit utilisée dans l'étape de correction du mouvement libre. Voir la section 4.5.2 pour plus de détails.

Algorithme

Nous avons vu ci-dessus que la compliance que nous utilisons est donnée par la formule :

$$\boldsymbol{C} = \boldsymbol{R}\boldsymbol{C}^0\boldsymbol{R}^T \tag{4.31}$$

Cependant, nous n'allons pas construire cette matrice car c'est trop coûteux et inutile dans la mesure où seule une petite partie de la matrice associée aux points de contact est nécessaire.

La résolution des contacts par Compliance Warping reprend donc l'algorithme 13 dans son ensemble. La seule différence réside dans :

1. une phase de précalcul des matrices de compliance C^o pour chacun des objets o qui est faite hors-line.
2. une phase de rotation des normales aux contacts qui est faite à chaque pas de temps, avant la résolution des contacts. Ce sont les lignes 1 à 3 de l'algorithme 14.

L'algorithme 14 détaille l'enchaînement de ces étapes.

Pré-calcul de $\boldsymbol{C}^0$ Le pré-calcul de $\boldsymbol{C}^0$ est fait une fois pour toute hors-ligne, à partir d'un modèle de l'objet et d'un schéma d'intégration dans le cas dynamique. Pour cela, ses colonnes $\boldsymbol{c}_i$ sont déterminées en résolvant l'équation 4.16, qui tire partie du caractère généralement creux des matrices de masse et raideur.

Une fois cette matrice évaluée, elle peut être réutilisée dans des nombreuses simulations, pour peu que les contraintes initiales de Dirichlet soient les mêmes. Le surcoût de ce pré-calcul en est d'autant réduit.

Discussion

L'avantage principal de cette méthode réside dans son efficacité. Contrairement à la résolution classique par LCP, la compliance n'a pas à être en partie recalculée à chaque pas de temps, ce qui rend notre méthode indépendante de la complexité du modèle. La complexité de notre méthode est alors simplement fonction du nombre de point de contact, et bénéficie en plus de n'avoir pas à recalculer certaines colonnes de la compliance. Elle apporte donc une double amélioration : la suppression du calcul de compliance et l'indépendance à la complexité du modèle simulé.

Autre avantage de cette méthode : elle est directement adaptable aux objets rigides ou poly-articulés, et ce sans avoir à effectuer un seul réglage. La méthode et les compliances utilisées sont exactement les mêmes quelle que soit la raideur des objets simulés. C'est un énorme avantage par rapport aux méthodes basées sur les pénalités qui requièrent le réglage fin des raideurs de contacts.

4.5.2 Correction du mouvement libre

Méthode

Le calcul des forces de contact que nous avons opéré dans la section précédente a été fait avec la compliance approchée. Le respect des lois de Signorini et Coulomb n'est donc valable que pour un objet dont la compliance est la compliance approchée.

Algorithm 14 Résolution du contact et des frottements par GAUSS-SEIDEL et COMPLIANCE WARPING

for $1 \leq i \leq n$ **do**
  $\boldsymbol{n}_i = \boldsymbol{R}_i \boldsymbol{n}_i$
end for
while $\sum_{i=0}^{n} \frac{||\delta_i^{k+1} - \delta_i^k||}{||\delta_i^{k+1}||} > \epsilon_1$ **do**
  for $1 \leq i \leq n$ **do**
    $\boldsymbol{\delta}_i^{k+1} = \left(\boldsymbol{\delta}_i^{free} - (\sum_{j=1}^{i-1} \tilde{\boldsymbol{w}}_{ij} \boldsymbol{f}_j^{k+1} + \sum_{j=i-1}^{n} \tilde{\boldsymbol{w}}_{ij} \boldsymbol{f}_j^k) \right)$
    $\boldsymbol{\delta}_i = \tilde{\boldsymbol{w}}_{ii} \boldsymbol{f}_i^k$
    $f_{i,n}^{k+1} = f_{i,n}^k - [1\ 0\ 0] \tilde{\boldsymbol{w}}_{ii}^{-1} \boldsymbol{\delta}_i$
    if $f_{i,n}^{k+1} < 0$ **then**
      $\boldsymbol{f}_i^{k+1} = 0$
    else
      $f_{i,t}^{k+1} = f_{i,t}^k - [0\ 1\ 0] \tilde{\boldsymbol{w}}_{ii}^{-1} \boldsymbol{\delta}_i$
      $f_{i,s}^{k+1} = f_{i,s}^k - [0\ 0\ 1] \tilde{\boldsymbol{w}}_{ii}^{-1} \boldsymbol{\delta}_i$
      if $\sqrt{(f_{i,t}^{k+1})^2 + (f_{i,s}^{k+1})^2} > \mu f_{i,n}^{k+1}$ **then**
        $f_{i,t}^{k+1} * = \frac{f_{i,n}^{k+1}}{\sqrt{(f_{i,t}^{k+1})^2 + (f_{i,s}^{k+1})^2}}$
        $f_{i,s}^{k+1} * = \frac{f_{i,n}^{k+1}}{\sqrt{(f_{i,t}^{k+1})^2 + (f_{i,s}^{k+1})^2}}$
      end if
      $\boldsymbol{\delta}_i^{k+1} = \tilde{\boldsymbol{w}}_{ii} \boldsymbol{f}_i^k$
    end if
  end for
  $k = k + 1$
end while

Donc, pour assurer la bonne gestion du contact et du frottement, nous devons calculer le mouvement contraint à l'aide de la compliance approchée. Le calcul reprend un schéma similaire à celui utilisé pour la résolution des contacts avec le COMPLIANCE WARPING. C'est à dire que :

1. Les forces de contact sont ramenées dans la configuration d'origine à l'aide des rotations aux noeuds : $\boldsymbol{r}_i^0 = (\boldsymbol{H}_i\boldsymbol{R}_i)^T\boldsymbol{f}_i$.
2. Les déplacements $\boldsymbol{\delta}_{ji}^0$ induits par cette force $\boldsymbol{r}_i^0$ sont calculés dans la configuration d'origine : $\boldsymbol{\delta}_{ji}^0 = \boldsymbol{c}_{ji}^0\boldsymbol{r}_i^0$.
3. Ces déplacements $\boldsymbol{\delta}_{ji}^0$ sont ensuite sommés pour donner le déplacement total de chaque nœud j : $\boldsymbol{\delta}_j^0 = \sum_i \boldsymbol{\delta}_{ji}^0$.
4. Puis, ces déplacements $\boldsymbol{\delta}_j^0$ sont ramenés dans la configuration courante : $\boldsymbol{\delta}_j = \boldsymbol{R}_j\boldsymbol{\delta}_j^0$
5. Enfin, ces déplacements sont additionnés aux déplacements nodaux $\boldsymbol{u}_j$: $\boldsymbol{u}_j = \boldsymbol{u}_j + \boldsymbol{\delta}_j$. Le mouvement ainsi obtenu garantie la non interpénétration et le respect des lois de SIGNORINI et COULOMB.

Discussion

Les avantages de cette méthode sont multiples. Tout d'abord, le nombre de points de contact étant généralement peu important en regard du nombre total de degré de liberté, cette méthode ne nécessite que peu de calculs. En omettant la phase de précalcul de $\boldsymbol{C}^0$ et la détection de collision la résolution du problème de complémentarité est indépendante du nombre total de nœuds. Seul le nombre de points de contact est à prendre en compte. La correction du mouvement est elle linéaire en nombre de nœuds du maillage.

De plus, comme nous utilisons la compliance approchée, il n'est pas nécessaire, contrairement à la méthode traditionnelle de résoudre l'équation 3.7 pour connaître de déplacement engendré. Il suffit de multiplier les forces de contact par la compliance. Cela rend cette méthode particulièrement efficace.

Enfin, et c'est l'objet de cette méthode, les lois de SIGNORINI et COULOMB sont respectées.

Par contre, le mouvement contraint ainsi calculé n'utilise pas la vraie compliance du modèle, donc le mouvement n'est pas exactement celui qu'aurait eu l'objet avec le modèle corotationnel élémentaire. Cependant, cette approximation ne portant que sur l'effet de forces en quelques points de contact, n'altère pas de manière significative la déformée, comme en témoignent les résultats de la section 4.5.4.

De plus, une simulation respectant les lois de SIGNORINI et COULOMB mais pas le mouvement exact semble toujours plus réaliste à l'utilisateur qu'une simulation respectant précisément la loi de mouvement, mais autorisant des interpénétrations.

4.5.3 Estimation des rotations aux noeuds

La méthode utilisée pour estimer les rotations aux noeuds est importante, car c'est elle qui va garantir que notre approximation, c'est à dire notre modèle corotationnel nodal, soit proche du modèle corotationnel élémentaire qui est utilisé pour le calcul des déformations.

Estimation depuis le trièdre

La rotation au nœud peut être évaluée simplement en associant aux points un trièdre, et en évaluant la rotation permettant de passer du trièdre dans la configuration d'origine au trièdre dans la configuration courante.

Estimation à l'aide des rotations aux éléments

Notre approximation étant utilisée conjointement avec un modèle corotationnel élémentaire, il est possible de réutiliser les rotations, calculées par décomposition polaire, des éléments contenant le nœud.

Par exemple, la rotation au nœud peut être égale à la moyenne des rotations à chaque élément le contenant.

Résultats

Dans la pratique, les différences entre ces deux méthodes ne sont pas significatives. Simplement, la méthode basée sur les rotations aux éléments, réutilisant des rotations déjà calculées, est plus efficace.

4.5.4 Performances et Précision

La méthode Compliance Warping a été testée sur un Intel Celeron M 520 at 1.6 GHz avec 1Gb de RAM. À noter que cette méthode a été implantée dans SOFA et est désormais librement disponible sur internet. Il est donc particulièrement simple de la tester.

Validation du modèle

Pour nous assurer du bien fondé de notre approximation, nous avons effectué une batterie de tests permettant d'évaluer l'erreur introduite. Pour cela, nous déformons divers objets en utilisant notre approximation et la compliance exacte. Cependant, le calcul de la compliance exacte étant tellement coûteux, nous n'avons pu faire de comparaisons que pour des modèles relativement simples.

Par exemple, nous avons simulé une chaîne de 5 maillons, dont deux étaient rigides et libres, deux étaient déformables et libres, et un dernier rigide et fixé. Voir la figure 4.19-(a). Sous l'effet de la gravité ces maillons tombent, entrent en contact et la chaîne oscille. Une fois les maillons en contact, la scène comporte environ 35 points de contacts avec frottement.

Comme le montrent les figures 4.19-(b) et 4.19-(c), il n'y a pas de différence visible à l'oeil nu entre les deux modèles. L'erreur relative, basée sur la norme de la différence entre les deux déformées, est donnée dans le tableau 4.1 pour deux simulations différentes.

Dans la première, la chaîne est initialement verticale et les maillons tombent verticalement. Dans la seconde, la chaîne est orientée horizontalement. En tombant, les maillons font ainsi osciller la chaîne, et sollicitent ainsi plus les frottements.

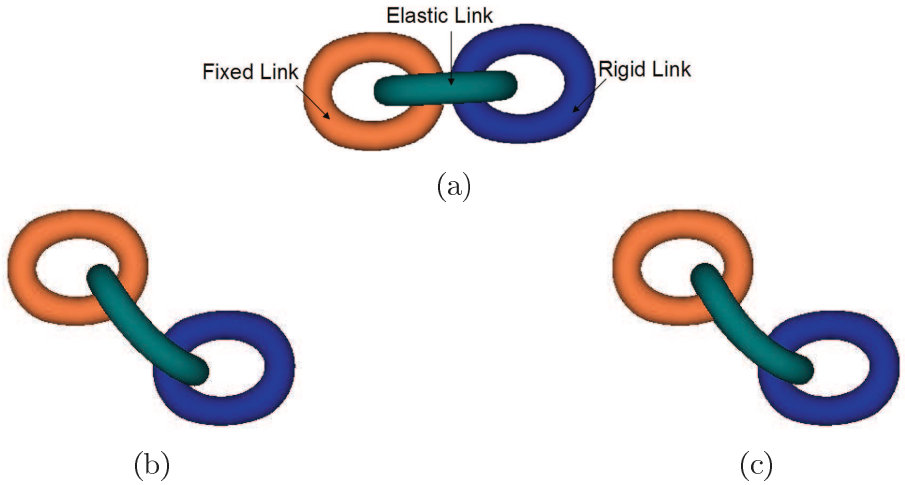


FIG. 4.19 – (a) : Configuration initiale. (b) : Déformation avec la *compliance* exacte ; (c) : Déformation avec la *compliance* exacte.

	Largest Deformation	Final Shape
Chaîne Horizontale à 3 maillons	1.5%	1.7%
Chaîne Verticale à 3 maillons	4.4%	3.1%

TAB. 4.1 – Erreur relative sur la déformée pour la déformée maximale et la déformée finale. Les 3 maillons sont initialement alignés soit verticalement soit horizontalement.

Dans les deux cas, notre méthode est indubitablement plus rapide, et ce au coût d'une légère erreur. Voir le tableau 4.2.

Performances

Nous avons comparé l'efficacité de notre méthode avec celle consistant à utiliser la compliance exacte ainsi que l'intégration exacte des déplacements. Les résultats sont présentés dans le tableau 4.2. Notre méthode permet une très large accélération, et ce d'autant plus que le modèle est complexe et que le nombre de points de contacts est important. En effet, dans la méthode exacte, plus il y a de points de contact, plus il faut calculer de lignes de la matrice de compliance, et plus le modèle est complexe, plus ces lignes sont longues à calculer. Notre méthode est indépendante de ces deux limitations.

Simulation d'objet déformables et rigides

Cette exemple illustre le fait que notre méthode s'applique indifférement pour des objets déformables, rigides ou poly-articulés. Nous avons mélangé dans la simulation les configurations de contact suivantes : Deformable - Deformable, Deformable - Rigide, Rigide- Rigide. Voir figure 4.20. Toutes ces configurations de contact sont gérées efficacement et de manière stable par notre méthode.

	Tetra. Num.	Contact Num.	Exact Compl.	Appro. Compl.
Pluie de Dino	40K	2000	N/A	3.007s
Chaîne à 3 maillons	862	35	13794.1ms	8.563ms
Chaîne à 3 maillons	862	13	6695ms	1.520ms
Chaîne à 3 maillons	862	36	16908.5ms	9.36ms
Chaîne à 3 maillons	862	12	8203.6ms	1.366ms
Chaîne à 5 maillons	2*862	67	11027.8ms	4.72ms

TAB. 4.2 – Temps de calcul de la compliance au point de contact pour différents modèles

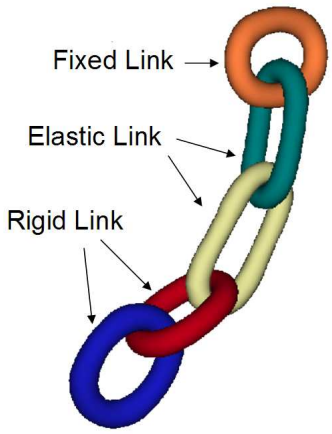


FIG. 4.20 – Différents type de contact sont gérés dans cette exemple : Deformable - Deformable ; Deformable - Rigid ; Rigid - Rigid

Pluie de Dinosaures

Pour évaluer la stabilité et la robustesse lorsqu'un grand nombre de contacts sont impliqués, nous avons simulé la chute de 40 dinosaures ($\sim 1K$ tetraèdres chacun). Cette simulation présente des changements drastiques de points de contact et frottement ($\sim 2K$ contacts frottants). Voir 4.21.

Dans ce cas exigeant, la simulation reste stable et les calculs sont effectués rapidement. Voir le tableau 4.2. Une fois encore, les performances dépendent du nombre de points de contacts impliqués. Cela dit, notre méthode est généralement plus rapide de 2 à 3 ordres de grandeur.

De plus, contrairement aux méthodes par pénalités, notre méthode ne nécessite pas de réglage et fonctionne directement, sans adaptation, avec des objets déformables de raideurs différentes ou même rigides.

Enfin, l'utilisation d'un schéma d'intégration implicite assure la stabilité quelque soit la raideur utilisée.

4.6 Conclusion

Nous avons vu dans ce chapitre l'importance d'une bonne gestion des contacts, tant pour la stabilité de la simulation que pour le confort d'utilisation et le réalisme nécessaires à l'utilisateur. Cette prise en compte des contacts repose sur deux étapes : la détection de collision, dont nous avons détaillé rapidement les diverses méthodes couramment employées pour des modèles déformables, et la gestion des contacts, dont nous avons aussi survolé les techniques de résolution les plus usitées.

Ce rapide aperçu nous a permis de mettre en exergue la méthode de gestion des contacts par Complementary Problem, qui satisfait nos exigences de réalisme, c'est à dire le respect des lois de Signorini et Coulomb ainsi que de stabilité. Cependant, cette présentation a aussi pointé la limitation majeure de cette méthode, sa lenteur, qui conduit dans la pratique à son abandon au profit de méthodes, telles que les méthodes par pénalités, plus rapides mais moins physiques et parfois moins stables.

En effet, la résolution des contacts par complémentarité s'appuie sur l'utilisation de la matrice de compliance, or cette matrice requérant pour être calculée une inversion, fait chuter les performances de la méthode. A cela s'ajoute le fait que cette compliance dépend de la complexité des modèles simulés, ce qui découle en des performances encore plus altérées lorsque l'objet simulé est complexe. Nous avons donc chercher à supprimer cette étape de construction de la compliance.

Pour cela, nous sommes partis d'une observation que nous avions faite sur les modèles corotationnels nodaux. Nous avions remarqué que ces derniers présentaient la propriété intéressante de constituer de bons pré-conditionneurs pour des systèmes basés sur des modèles corotationnels par éléments. Ils fournissent donc une bonne approximation de la matrice tangente obtenue par des modèles corotationnels élémentaires qui peut à ce titre se substituer à eux dans l'étape de résolution par complémentarité. Et ce avec le profit que leur inversion, nécessaire pour le calcul de la compliance, se fait immédiatement pour peu que leur compliance dans la configuration initiale ait été calculée.

L'utilisation de ces modèles dans la phase de gestion des contacts est donc à la base de notre méthode de Compliance Warping. Cette méthode permet de résoudre les contacts en

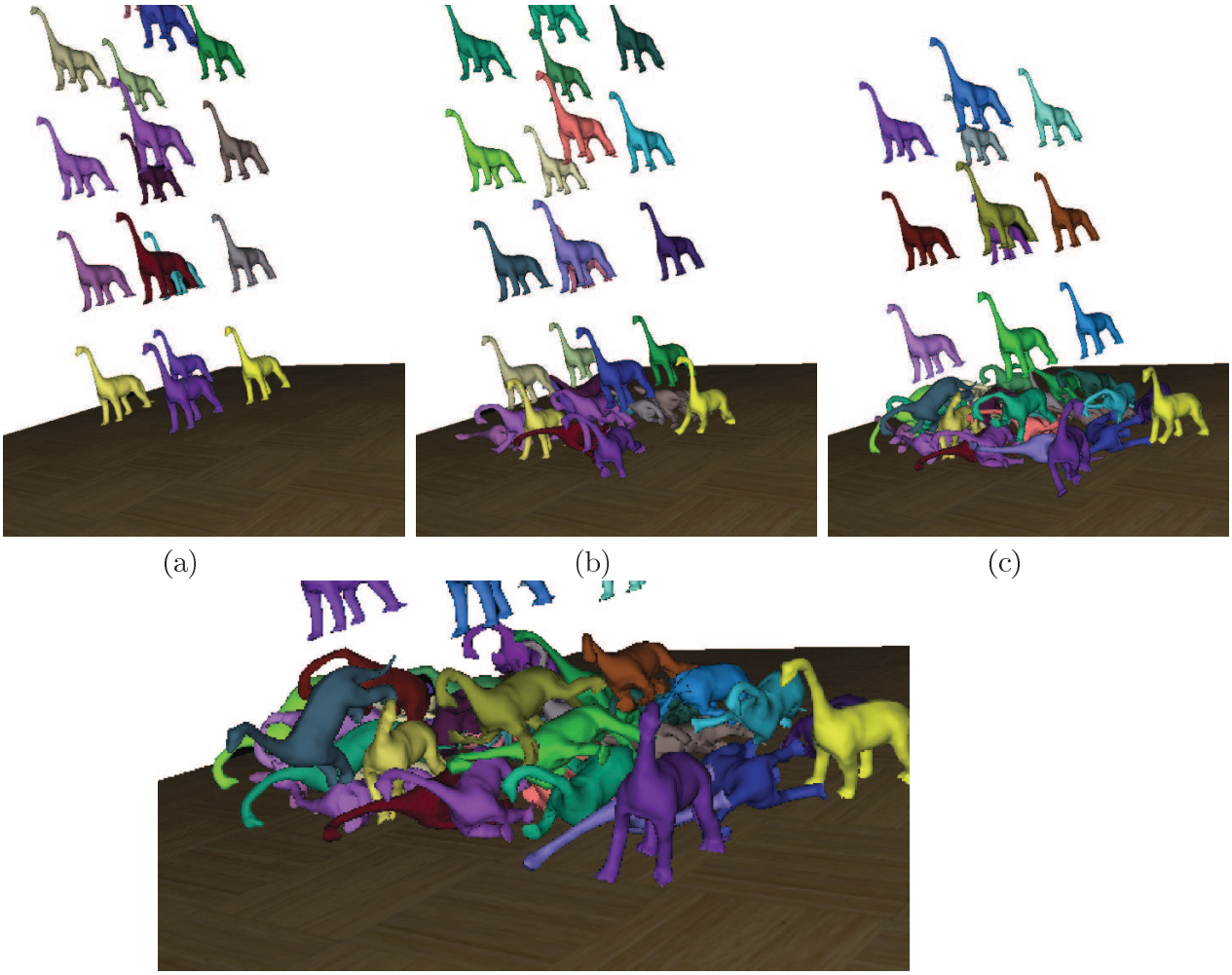


FIG. 4.21 – (a, b, c) : Les dinosaures accélèrent durant la chute libre ; (d) : Les dinosaures s'écrasent et la collision génère de nombreux points de contact et frottement.

respectant les lois de SIGNORINI et COULOMB de manière interactive, en s'affranchissant de la dépendance à la complexité du modèle simulé, et en n'étant plus fonction que du nombre de points de contact. En plus de cela, cette méthode s'applique directement à des modèles rigides ou poly-articulés et ne nécessite aucun réglage *a contratio* des méthodes par pénalités dont le choix des raideurs de contact est souvent délicat.

Ainsi, avec le COMPLIANCE WARPING, nous sommes parvenu à gérer efficacement de nombreux contacts de manière interactive, générique et physiquement plausible. Ces travaux ont fait l'objet d'une publication dans COMPUTER GRAPHICS INTERNATIONAL [SDCG08].

4.7 Perspectives

Cette méthode est très efficace, cependant, il est toujours possible de lui reprocher d'être basée sur une approximation. C'est pourquoi nous avons commencé à réfléchir à un nouveau schéma de simulation qui minimise l'impact de cette approximation.

L'idée serait de prendre en compte les forces de contact du pas de temps précédent lors de l'intégration du mouvement, ce qui devrait permettre de minimiser l'influence de l'approximation, notamment l'effet de la phase de calcul du déplacement contraint qui n'utilise pas la compliance exacte, et qui produit donc un déplacement contraint différent de celui exact.

Autre amélioration envisageable, qui serait dans la veine de nos travaux sur les ondelettes et les solveurs hiérarchiques, consisterait à stocker la compliance initiale sous une forme compressée à l'aide d'une approche multi-résolution. Plus radicalement, il serait intéressant d'utiliser une compliance initiale à une résolution grossière doublé d'un operateur de prolongation et d'un de restriction. Les calculs s'en trouveraient réduits. Resterait à évaluer l'impact de cette simplification extrême sur la qualité de la simulation.

Chapitre 5

Retour Haptique

Sommaire

5.1 Introduction

En dépit des efforts réalisés par la communauté Computer Graphics, les deux derniers chapitre ont fait clairement apparaître la lenteur, relative, des simulations de corps déformables. Ainsi, s'il est possible pour des modèles compliqués, basés sur la théorie de l'élasticité, de tourner à des fréquences visuellement satisfaisante, c'est à dire de l'ordre de 30 fps, il n'est pas encore envisageable d'obtenir des simulations tournant à une fréquence de l'ordre du KHz.

Or nous verrons dans ce chapitre que cette fréquence de 1KHz est l'une des contraintes imposées par l'introduction de l'homme dans la boucle de simulation. Ne pas la respecter implique un rendu haptique de piètre qualité qui nuit à l'immersion de l'utilisateur dans l'environnement virtuel.

Le respect de cette contrainte de fréquence par la boucle physique n'étant pas actuellement possible, nous proposons dans ce chapitre une méthode permettant de découpler la boucle physique de la boucle haptique. La boucle physique est alors libre de tourner à la faible fréquence qu'implique la complexité des équations qu'elle a à résoudre, tandis que la boucle haptique a toute latitude pour s'exécuter aussi rapidement que possible.

FIG. 5.1 – Apprentissage de gestes chirurgicaux avec SOFA.

FIG. 5.2 – Pose de mastic virtuel avec XDE au CEA.

La contribution majeure que nous proposons consiste à utiliser dans la boucle haptique le problème de complémentarité issu de la boucle physique pour fournir à l'interface haptique des forces de contacts respectant les lois de SIGNORINI et COULOMB.

Le plan suivi dans ce chapitre s'attache à présenter dans la section 5.2 l'intérêt du retour d'effort ainsi que les grands types d'interfaces haptiques existants. Puis nous nous penchons plus en détails sur les effets de l'inclusion de l'homme dans la boucle de simulation, notamment au niveau de la stabilité, dans la section 5.3. Ensuite, nous montrons comment l'instabilité introduite par l'interaction humaine peut être contournée en ajoutant un couplage entre l'homme et la simulation. Ce sera l'objet de la section 5.4. Enfin, dans la section 5.5 et ce sera notre contribution à ce domaine, nous présenterons notre schéma de couplage haptique qui découple le rendu haptique, tournant alors à la fréquence haptique et respectant les lois des SIGNORINI et COULOMB, de la boucle physique qui tourne elle plus lentement.

5.2 De l'importance du retour d'effort

5.2.1 Généralités

Avec un dispositif haptique, l'utilisateur peut déplacer un objet dans la scène virtuelle, et peut surtout ressentir les efforts appliqués sur cet objet. Les avantages de ce type d'interface sont multiples. En plus d'améliorer la perception de l'environnement par l'utilisateur, elles peuvent lui permettre de ressentir les mêmes efforts que dans l'environnement réel. Ainsi, l'apprentissage d'un geste par ce biais devient possible, que ce soit dans l'industrie, avec par exemple l'apprentissage de la pose du mastic (figure 5.2), dans le secteur médical pour l'apprentissage de gestes chirurgicaux (voir figure 5.1), avec entre autres exemples, la chirurgie endoscopique, ou dans l'aérospatiale avec les entraînements pour le SPACE SHUTTLE [CSB95].

En plus de l'ajout d'une modalité de perception supplémentaire, l'intérêt majeur du retour d'effort réside dans l'accroissement de l'efficacité de l'utilisateur lorsqu'il interagit avec un environnement distant ou virtuel. Ainsi, dans le cadre de la téléopération, il a été montré que

l'opérateur est nettement plus efficace dans les taches qu'il a à réaliser lorsqu'il bénéficie d'un retour d'effort. De manière générale, il apparaît qu'une tache est réalisée avec plus de rapidité quand il y a retour d'effort.

Les états de l'art [SCB04, OL05] sont une bonne présentation des problèmatiques haptiques et des solutions qui y sont apportées.

5.2.2 Les interfaces haptiques

Généralité

Historiquement apparues dans les simulateurs d'avions et dans le cadre de la téléopération, les interfaces à retour d'effort ont connu un regain d'intérêt avec l'avènement de la simulation informatique, de la réalité virtuelle et du jeux vidéo. Il existe à l'heure actuelle de nombreuses interfaces, qui stimulent aussi bien le sens tactile que celui kinesthésique. Il faut en effet distinguer deux types de ressenti : celui superficiel, à travers duquel nous ressentons la surface des objets : textures, température, rugosité, ... et celui propre aux récepteur proprioceptifs des muscles, à travers duquel nous ressentons des efforts mais aussi la position de nos membres. Les interfaces les plus souvent rencontrées sont, entre autres : l'OMNI et le PHANTOM de chez SENSABLE, ainsi que le VIRTUOSE de chez HAPTION, qui sont des interfaces kinesthésiques, tandis que le VITALE du LEMS au CEA est une interface tactile.

Il est important de noter qu'il existe une différence majeure entre les interfaces haptiques et les autres dispositifs interfaçant généralement une simulation avec les sens humains. En effet, les interfaces visuelles et auditives sont des médias uni-directionnels, dans le sens où l'information est simplement *lue* par l'utilisateur humain. Les interfaces haptiques sont pour leur part, des médias bi-directionnels, c'est à dire que l'utilisateur *lit* l'interface, il ressent des efforts ou des déplacements, mais il *écrit* aussi, car il impose à l'interface un déplacement ou un effort. C'est le constat que font HAYWARD et ASTLEY dans [HA96]. Ils notent aussi que la principale difficulté dans le retour haptique est justement l'écriture de ces forces ou déplacements sur l'interface.

Fonctionnement

Concernant le mode de fonctionnement, il existe principalement deux grands types d'interface à retour d'effort [SCB04] : les interfaces de type *admittance* et les interfaces de type *impédance*. Ces deux types d'interfaces sont très différents dans leur mode de contrôle. Les interfaces par *admittance* mesurent une force et en déduisent une position tandis que les interfaces par *impédance* mesurent une position et en déduisent une force.

Le controle en *admittance* est généralement recontré pour des bras non réversibles, dont l'effecteur ne peut être déplacé en appliquant un effort dessus. Son déplacement se fait nécessairement à l'aide de ses actionneurs. Ainsi, l'effecteur est parfois équipé d'un capteur d'effort qui permet alors un control en impédance. Le contrôle de l'effecteur par l'application d'une force devient alors possible avec une méthode en *impédance*.

L'interface la plus fréquemment rencontrée est l'interface en *impédance*, qui est moins coûteuse que l'interface en *admittance*. Les interfaces en *admittance* présentent par contre l'avantage de pouvoir générer de plus grandes forces et ce dans un plus grand espace de travail.

De nombreux points président à la conception d'une interface haptique, parmi lesquels les

plus importants sont :

- La transparence : l'interface doit présenter une inertie la plus faible possible ainsi que des frottements et une viscosité limitées de manière à ce que son maniement soit transparent pour l'utilisateur. Il faut que l'utilisateur en vienne à oublier l'interface. Sans cela, la manipulation d'une telle interface devient rapidement pénible.
- Des contraintes minimales sur le déplacement : l'utilisateur doit pouvoir manipuler l'interface haptique de manière isotrope, sans ressentir de contraintes dans des directions particulières, et ce dans un espace de travail aussi grand que possible.
- Il faut trouver le juste équilibre entre l'intensité maximale des forces rendues, la résolution, l'espace de travail et la bande passante des forces.
- Enfin, l'interface doit être ergonomique. Sa prise en main et son encombrement doivent être mûrement réfléchis. En plus du confort de l'utilisateur, elle doit s'assurer de ne pas blesser l'utilisateur en limitant l'intensité des forces qu'elle génère.

Historiquement, les dispositifs haptiques sont actionnées par des moteurs à courant continu. Le couple moteur/câble est fréquemment rencontré. Cependant des interfaces basées sur l'exploitation de matériaux actifs, impliquant notamment des phénomènes électro- et magnéto- rhéologiques commencent à apparaître [MZK03]. Ce type de technologie, employant des polymères electro- ou magnéto-actifs présente une bande passante de haute fréquence, une large amplitude de déformation, et peut générer des forces importantes. Il est aussi possible de se dispenser d'interface mécanique et de stimuler directement les muscles de l'utilisateur [KSB06].

Évaluation

L'évaluation de la qualité d'une interface est aussi une tâche ardue, les spécifications données par les constructeurs n'étant pas toujours fiables. Hayward et Astley détaillent dans [HA96] les principaux critères à observer :

- Le nombre de degrés de liberté. C'est le paramètre le plus facile à identifier. Il faut prendre garde de bien distinguer les degrés de liberté en déplacement de ceux en retour d'effort. Ainsi, l'Omni de Sensable offre 6 DDL en position et seulement 3 en effort. A noter qu'il n'est pas toujours utile de disposer du maximum de DDL. Certaines applications s'accommodent très bien de 1 DDL.
- L'interfaçage proprement dit avec le dipositif haptique, c'est à dire la façon dont sont couplés mécaniquement la partie du corps humain concernée et le dispositif.
- L'espace de travail, c'est à dire l'amplitude admissible pour chaque degré de liberté. C'est un critère difficile à représenter visuellement pour des interfaces ayant plusieurs DDL.
- La force maximale.
- L'inertie et l'amortissement de l'interface, soit sa transparence pour l'utilisateur.
- L'accélération maximale.
- La densité de puissance, qui caractérise la compacité du dispositif en regard des forces délivrables.
- La précision des déplacements ou des forces rendues.
- La bande passante, qui est une propriété très importante pour la qualité du ressenti. Elle doit être d'au moins 1Khz.

Certains auteurs distinguent aussi les interfaces portables de celles fixes [Smi97, MZK03].

Quelques interfaces

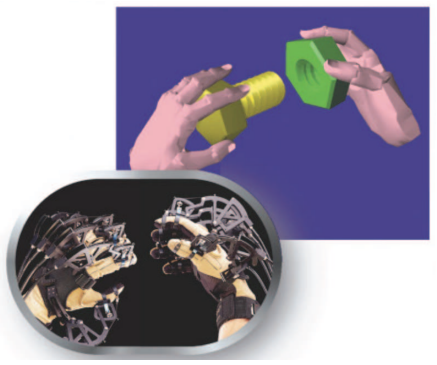


FIG. 5.3 – Le gant CYBERGRASP de IMMERSION.

FIG. 5.4 – Le gant RUTGERS MASTER II. La position des actionneurs empêchede refermer la main.

FIG. 5.5 – L'HAPTIC WORKSTATION de chez WORKSTATION.

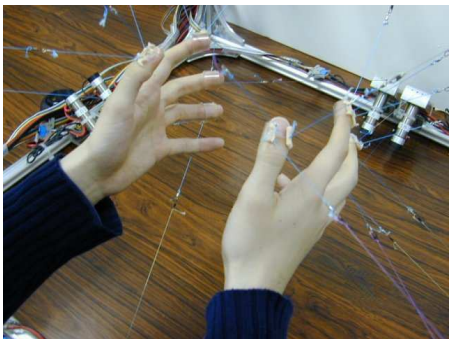


FIG. 5.6 – Le SPIDAR-8 du TOKYO INSTITUTE OF TECHNOLOGY

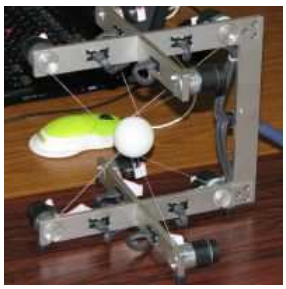


FIG. 5.7 – Le SPIDAR-G.

Les interfaces de type exosquelette Le principes des interfaces à exosquelette est d'utiliser une structure qui soit externe, une sorte de squelette extérieur muni d'actionneurs, pour appliquer des efforts sur divers membres du corps humain. Ces membres peuvent être un bras, une main, un doigt, ...

Le CYBERGRASP de la société IMMERSION, ou le RUTGERS MASTER II [BBPB02] sont des exemples de gants à retour d'effort utilisant des exosquelettes. Voir figures 5.3 et 5.4.

Le même système peut être appliqué à un, le CYBER FORCE, ou deux bras, comme le montre la figure 5.5 avec l'interface HAPTIC WORKSTATION de chez IMMERSION.

Les interfaces à câbles Ces interfaces utilisent des câbles qui sont rattachés aux différentes parties du corps à stimuler. L'actionnement de ces câbles se fait à l'aide de moteurs à courant continu. L'espace de travail et la dimension de ces interfaces peuvent varier de façon conséquente, comme en témoigne le SPIDAR-8 et le SPIDAR-G du TOKYO INSTITUTE OF TECHNOLOGY. Voir figures 5.6 et 5.7.

L'inconvénient de ces interfaces est qu'elles ne gèrent pas très bien les efforts en rotation. L'amplitude des rotations, notamment, est assez limitée.

Les bras à retour d'effort Les bras à retour d'effort sont les interfaces rencontrées le plus fréquemment. Les PHANTOM et OMNI (figure 5.8) de chez SENSABLE se retrouvent dans de nombreux laboratoires de recherche en raison de leur polyvalence. Ils se présentent sous la forme d'un bras articulé qui présente généralement 6 DDL en déplacement, et de 3 à 6 DDL en retour d'effort.

Le VIRTUOSE d'HAPTION (figure 5.9) est un dispositif similaire à ceux de chez SENSABLE, mais de plus grande dimension, donc ayant un plus grand espace de travail. Il est aussi possible de lui adjoindre un autre degré de liberté en translation, ce qui permet d'augmenter encore cet espace.

C'est sur ce type d'interface, et plus particulièrement sur l'OMNI de chez SENSABLE que nous avons testé nos algorithmes.

FIG. 5.8 – L'OMNI de SENSABLE

FIG. 5.9 – Le VIRTUOSE de HAPTION.

5.2.3 Le rendu haptique

Une fois l'interface choisie ou conçue, il faut déterminer la manière dont les forces sont rendues. Comme pour le rendu graphique, ou un objet peut être rasterisé, raytracé, ... il existe de multiples façons de rendre un objet haptique, soit en sollicitant la perception kinesthésique, soit en stimulant les sens tactiles. De nombreux algorithmes ont donc été développés.

Il est néanmoins possible de distinguer deux approches :

- le rendu direct, dans lequel il existe une bijection entre la position de l'interface haptique et l'objet lié, souvent appelé sonde, dans l'environnement virtuel. Ce type de rendu bénéficie d'un mise en oeuvre simple, mais souffre du fait qu'il est impossible dans des simulation discrètes de garantir la non-interpénétration de la sonde avec les objets virtuels. Cela introduit des instabilités
- le rendu avec couplage, dans lequel l'objet de la simulation n'est plus directement lié avec la position de l'interface haptique. Le couplage se fait généralement à travers un couple ressort / amortisseur. Ce concept de couplage a été présenté pour la première fois par COLGATE *et al.* dans [CSB95]. ZILLES et SALISBURY l'ont étendu en créant le principe du GOD-OBJECT dans [ZS95]. Nous y reviendrons plus en détail dans la section 5.4.

A ces deux approches s'ajoute le nombre de degré de liberté rendus, ce qui multiplie encore le nombre de rendus possibles. Les nombreuses interfaces développées permettent en effet de rendre de 1, comme le ??, à 6 degré de liberté, comme le VIRTUOSE. La difficulté du rendu croît assez naturellement avec le nombre de degré de liberté. Par contre, il n'est pas toujours nécessaire de d'utiliser de nombreux degrés de liberté. Une interface à un degré de liberté est parfois suffisante à l'immersion de l'utilisateur dans la réalité virtuelle. C'est le cas par exemple des simulations d'injection avec seringue.

Receptors	Meissner Corpuscles	Merkel's Disks	Pacinian Corpuscles	Ruffini Corpuscles
Location	Glabrous skin; Dermis	Glabrous skin; Epidermis	Glabrous and hairy skin; Dermis and subcutanous	Glabrous and hairy skin; Dermis and subcutaneous
Adaptation rate	Rapid; RA-I	Slow; SA-I	Rapid; RA-II	Slow; SA-II
Receptive field	Small $12mm^2$	Small $12mm^2$	Large $100mm^2$	Large $60mm^2$
Frequency range	20-100Hz	0-10Hz	100Hz-1kHz	0-10Hz
Best response	30-40Hz	0-10Hz	150-130Hz	0-10Hz
Proportion	40%	25%	13%	19%
Function	Movement, Velocity	Pressure, Vibration	Acceleration, Pressure	Pressure, Skin shear, Thermal changes

FIG. 5.10 – Les différents types de mécanorécepteurs. Issue de [MZK03].

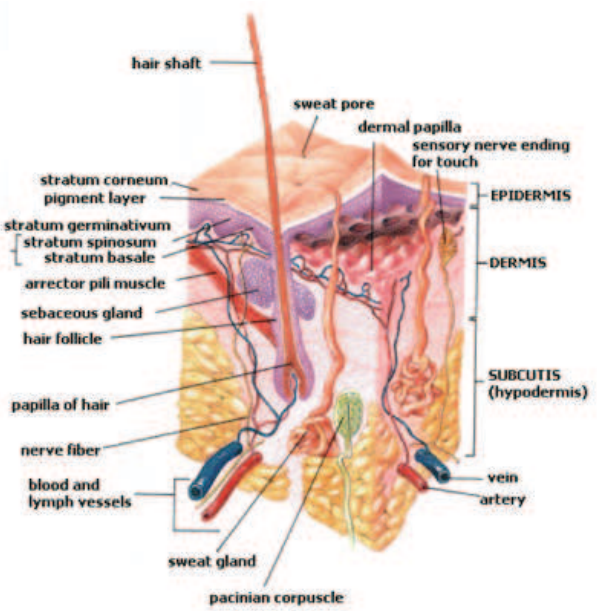


FIG. 5.11 – Répartition des méchanorécepteurs à la surface de la peau. Issue de Wikipedia.

5.3 Contraintes de l'inclusion de l'homme dans la boucle

5.3.1 Contraintes physiologiques

Les interfaces haptiques visent à stimuler un sens supplémentaire de l'humain pour améliorer sa perception d'un environnement distant ou virtuel. Il est donc important d'identifier les contraintes imposées par la physiologie humaine pour identifier clairement les contraintes nécessaires à la bonne perception du toucher.

Dans un premier temps, il est intéressant de détailler quelles sensations peuvent être ressenties par le toucher. Ces sensations sont nommées *modalités somatiques* dans la littérature, car il s'agit de modes de perception lié au système nerveux contrôlant les muscles et les récepteurs de stimuli extérieurs. Le contrôle des muscles se fait par les nerfs efférents tandis que la *lecture* des récepteurs se fait par les nerfs afférents. Ces *modalités somatiques* sont, comme le rapportent MAZZONE et al. dans [MZK03], au nombre de quatre :

- Le toucher, qui ne concerne que la perception d'état de surface. Les mecanorécepteurs se chargent de mesurer ces informations, et peuvent être classés suivant quatre grands types. Voir figures 5.10 et 5.11
- La sensation proprioceptive, qui mesure l'état statique mais aussi kinesthésique des muscles, c'est à dire leurs position et vitesse. La charge de ces mesures est laissés aux fibres afférentes de muscles. Cette modalité joue un rôle primordial dans la conservation de l'équilibre et l'exécution de mouvements.
- La douleur, qui est mesurée par les nocicepteurs.
- Enfin, la température, établie à l'aide des thermorécepteurs.

Bien qu'il existe des interfaces permettant de stimuler chacune de ces modalités, seuls les dispositifs activant les récepteurs proprioceptifs nous intéressent, puisque nous cherchons à faire du rendu kinesthésique. Ces récepteurs nécessitent un rendu à une fréquence de l'ordre du KHz [Bur96] pour fournir à l'humain une sensation qui paraisse naturelle. Cette fréquence de rendu, 1KHz, est à comparer avec celle nécessaire par exemple pour un rendu visuel fluide, environ 30Hz : le sens du toucher est beaucoup plus fin que la vue et est capable de discriminer facilement un signal discret d'un signal continu. Il faut donc rafraîchir très fréquemment les forces affichées par le dispositif actif, ce qui est particulièrement contraignant.

En effet, un fréquence de rafraîchissement de 1 KHz implique des pas de temps de simulations de 1ms, ce qui est à l'heure actuelle incompatible avec la simulation d'objet déformable. Ce type de simulation s'effectue généralement à une fréquence de l'ordre de quelques dizaines de HERTZ, pour peu que le modèle utilisé soit un tant soit peu complexe.

Cette sensibilité du toucher s'explique par le rôle prépondérant de la main dans l'interaction de l'homme avec son environnement. C'est une zone munie de nombreux récepteurs, jusqu'à 135 par cm^2, fortement innervée, et qui à de nombreuses connexions neurologiques avec le cerveau.

En plus de cette contrainte de rafraîchissement, la structure même de l'interface doit être suffisament raide pour que la sensation que ressent l'utilisateur soit directement celle calculée, et non pas une information filtrée par l'interface. C'est là aussi un aspect de la conception des interfaces à retour d'effort difficile à garantir, et dont il est possible de s'affranchir en couplant retour haptique avec des effets visuels qui "trompent" l'utilisateur [Smi97, SLA06].

5.3.2 Contraintes sur la stabilité

Dans [CSB95], COLGATE et al. montrent que les instabilités observées dans une simulation avec retour d'effort peuvent être dues à l'insertion de l'humain dans la boucle de simulation. Ils font remarquer que la présence de l'humain peut tout autant rendre instable une simulation stable, que rendre stable une simulation instable. La solution à ce problème a été apportée par le couplage haptique dont nous décrirons le fonctionnement dans la section 5.4.

5.4 Couplage haptique

Comme nous l'avons indiqué précédemment, l'introduction de l'humain dans la boucle de simulation peut rendre cette dernière instable. Pour éviter cela, COLGATE *et al.* ont suggéré d'utiliser un couplage virtuel entre l'interface haptique et la simulation. Voir figure 5.12. Ainsi, si la simulation est passive, la stabilité de la simulation couplée à l'interface haptique est assurée.

5.4.1 Couplage Virtuel

Nous cherchons à piloter un objet virtuel, soumis aux lois physiques d'une simulation, à l'aide d'une interface haptique. Suivant les raisons avancées ci desus, il n'est pas envisageable de lier directement la position de l'objet virtuel à la position de l'interface haptique par une bijection. Nous allons donc coupler ces deux objets à l'aide d'un ressort amorti virtuel. Ainsi, lorsque l'interface haptique se déplace, l'objet virtuel suit, entraîné par le ressort, ce qui revient en fait à asservir l'objet virtuel en position à l'aide d'un contrôleur Proportionnel Dérivé.

Si $\boldsymbol{\delta} = \bar{\boldsymbol{x}} - \boldsymbol{x}$ est la distance entre la position de l'interface haptique $\bar{\boldsymbol{x}}$ et la position de l'objet dans la scène $\boldsymbol{x}$, alors le couplage est gouverné par l'équation suivante :

$$\boldsymbol{M}\ddot{\boldsymbol{x}} = \boldsymbol{f}_{int} + \boldsymbol{f}_{ext} \tag{5.1}$$

où $\boldsymbol{M}$ est la masse de l'objet dans la simulation, $\boldsymbol{f}_{ext}$ les forces extérieures appliquées sur l'objet, et $\boldsymbol{f}_{int}$ les forces internes imposées par le ressort amortisseur, *i.e* :

$$\boldsymbol{f}_{int} = \boldsymbol{K}\boldsymbol{\delta} + \boldsymbol{B}\dot{\boldsymbol{\delta}} \tag{5.2}$$

Cette équation est ensuite intégrée dans le temps à l'aide d'un schéma d'intégration numérique. Nous utilisons un EULER implicite tel que décrit dans l'équation 3.7,

L'intérêt du couplage réside dans la propriété suivante, démontrée par COLGATE *at al.* dans [CSB95] : si une simulation passive est couplée à une interface haptique par un ressort amorti, alors l'ensemble simulation plus interface haptique est stable. C'est dû au fait que le couplage limite l'impédance maximale affichée par l'interface.

L'inconvénient de cette méthode, comme le fait remarquer l'auteur, est la nécessité d'utiliser des schémas d'intégration implicites pour assurer la passivité de la simulation. Or la résolution par schéma implicite est parfois coûteuse en temps de calcul.

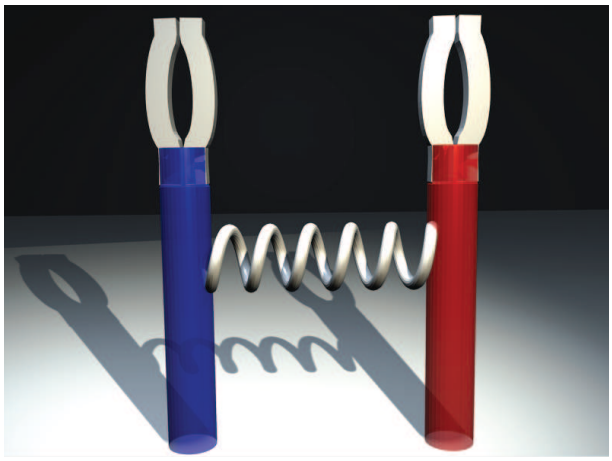


FIG. 5.12 – Couplage par ressort amortisseur.

5.4.2 God Object

Le concept ainsi que les motivations du GOD OBJET tel que défini par ZILLES et SALISBURY sont légèrement différents de ceux du couplage virtuel. Le GOD OBJET est un point virtuel, qui suit lui aussi la position de l'interface haptique, mais il n'y est couplé que par un ressort. Il suit simplement sa position, mais en respectant en plus des contraintes du type non-interpénétration.

La motivation diffère aussi en ce que le principe du GOD OBJECT a été conçu pour contourner cette observation : compte tenu de la nature discrète des simulations, s'il n'y a pas couplage entre l'interface et le point, c'est à dire si la position du point est en bijection avec celle de l'interface, il est impossible de garantir la non-interpénétration. Il devient alors difficile de calculer les forces de contact avec des objets minces ou lors de contacts multiples. En utilisant un GOD OBJECT qui se contente de suivre l'interface, et dont le mouvement est contraint par les lois de la simulation, alors ce dernier ne s'intersecte pas avec son environnement, et il n'existe plus d'ambiguïtés dans la valeur des forces. Le couplage entre l'interface et le point est assuré par un ressort dont l'énergie doit être minimisée, et le respect des contraintes est assuré par le calcul de multiplicateurs de LAGRANGE. Le système à résoudre est :

$$L = \frac{1}{2}\boldsymbol{\delta}^T\boldsymbol{\delta} + l_1(A_1\delta_x + B_1\delta_y + C_1\delta_z - D_1) + \cdots + l_n(A_n\delta_x + B_n\delta_y + C_n\delta_z - D_n) \qquad (5.3)$$

où $\boldsymbol{\delta}$ est toujours la différence entre la position du dispositif haptique $\bar{\boldsymbol{x}}$ et celle du point $\boldsymbol{x}$. Les A_n, B_n, C_n et D_n sont des contraintes planaires, et les l_n sont les coefficients de LAGRANGE correspondant.

L'avantage de cette méthode est quelle garantie aussi que l'utilisation d'une interface n'introduira pas d'instabilité, mais aussi qu'elle permet de déterminer rapidement des forces de contact. Ces forces sont données simplement par l'effort dans un ressort amorti liant l'interface au point.

Cette méthode, initialement présenté pour un simple point, peut être étendue à un objet quelconque.

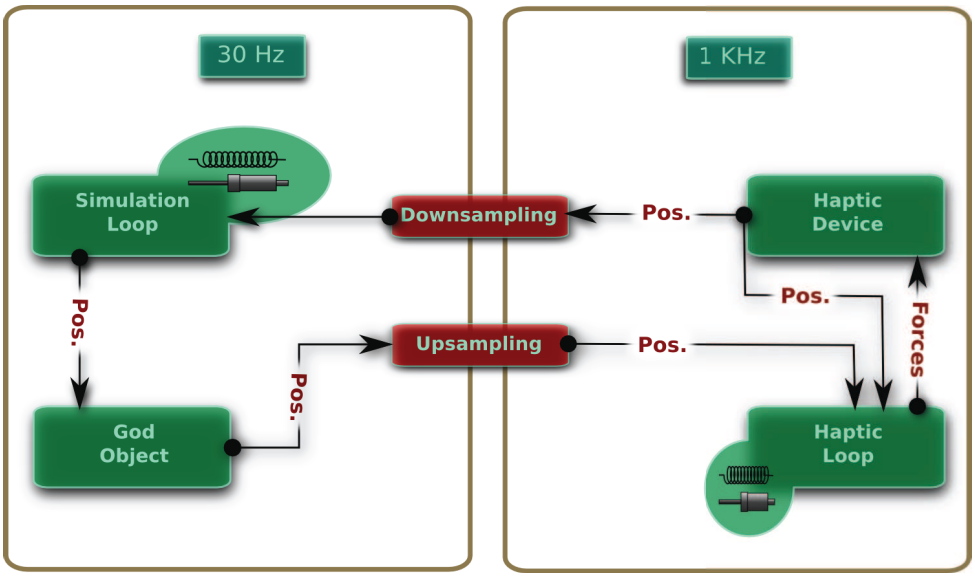


FIG. 5.13 – Le schéma du couplage d'une simulation tournant à 30 Hz et d'une interface haptique à 1 KHz.

5.4.3 Limitations du God Object et du Couplage Virtuel

Les méthodes du GOD OBJECT et du COUPLAGE VIRTUEL assurent la stabilité de la simulation. Cependant elles n'autorisent pas toujours un ressenti conforme aux lois de SIGNORINI et COULOMB.

Tous deux font apparaître une sensation de viscosité due à l'erreur de traînée entre la position de l'objet et cette de l'interface. Ce décalage est dû à la différence de fréquence entre la boucle haptique et la boucle de simulation. Ainsi, même lorsque l'objet contrôlé se déplace librement, cette différence de fréquence induit une différence de position entre l'objet simulé et la position donnée par l'interface haptique. Le couplage ressort/amortisseur est alors sollicité, et l'utilisateur ressent alors une force de rappel.

Et plus la simulation est lente, plus cette sensation de frottement visqueux s'intensifie. Cet artefact est donc particulièrement sensible dans des simulations de corps déformables qui souffrent de leur lenteur.

Dans la figure 5.13, le GOD OBJET est ainsi couplé à l'interface haptique par un ressort amorti dans la boucle de simulation. Ce couplage peut aussi être fait à l'aide de multiplicateurs de Lagrange. Les forces du retour d'effort sont aussi calculées à l'aide d'un ressort amorti dans la boucle haptique. La différence entre la consigne de l'interface et la position de l'objet dans la simulation entraîne l'apparition de frottement, même quand l'objet est libre.

Enfin, les forces de contact sont données simplement par le couplage ressort amortisseur, et ne respectent donc en rien les lois de SIGNORINI et COULOMB.

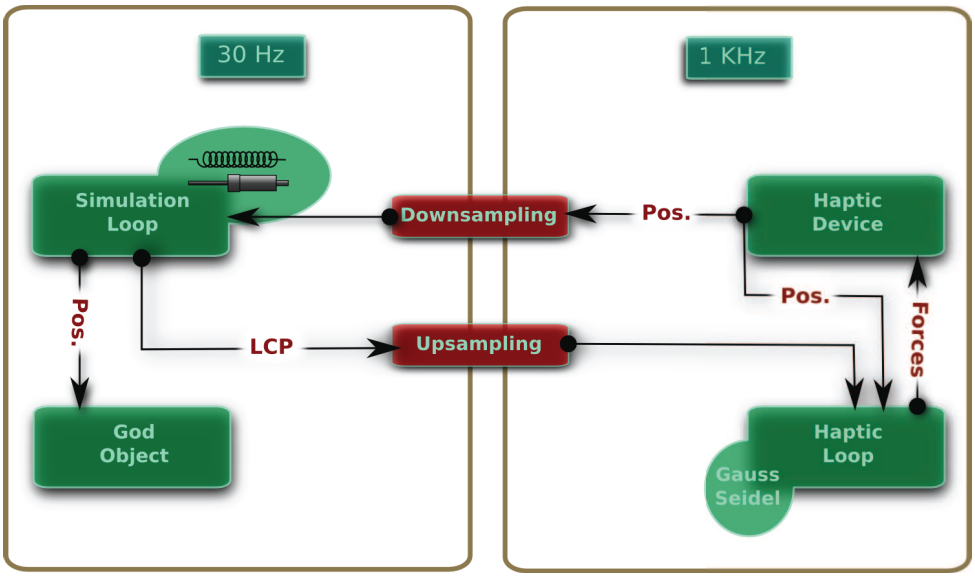


FIG. 5.14 – Le nouveau schéma de couplage.

5.5 NLCP haptique

Afin d'améliorer la vraisemblance des efforts ressentis par l'utilisateur à travers le dispositif haptique, nous avons proposé dans [SDC08] d'utiliser dans la boucle haptique le NON LINEAR COMPLEMENTARITY PROBLEM, NLCP, issu de la simulation, et rendu non linéaire du fait des frottements. Ainsi, non seulement les efforts ressentis seront conformes aux lois de SIGNORINI et COULOMB, mais la sensation de frottement visqueux due à la différence de fréquence entre la boucle haptique et celle physique disparaît.

5.5.1 Nouveau Schéma de Couplage

Avec ce nouveau schéma de couplage, nous voulons atteindre deux objectifs :

1. Nous voulons nous débarrasser du frottement visqueux dû à l'erreur de suivi entre le GOD OBJECT et le dispositif haptique.
2. Nous voulons pouvoir respecter les lois de SIGNORINI et COULOMB.

Pour cela, nous proposons d'utiliser un LCP au sein de la boucle haptique. Ce LCP sera utilisé en lieu et place du ressort/amortisseur pour calculer les efforts appliqués sur l'interface. Les lois de SIGNORINI et COULOMB sont donc de fait respectées, ce qui implique que sans contact, c'est à dire à distance d'interpénétration positives, les forces de contacts sont nulles. Donc, lorsque l'objet se déplace librement, aucun effort de traînée ne doit être ressenti.

Cette idée pose deux principaux problèmes :

1. Ce LCP ne peut malheureusement pas être construit à la fréquence de la boucle haptique. Sa construction nécessite la connaissance des LOCAL MINIMAL DISTANCES, or ces dernières ne peuvent être calculée à une fréquence aussi élevée pour des modèles déformables. De même, l'assemblage de l'oprérateur de DELASSUS, même avec la méthode présentée en 4.5 est trop gourmande en temps de calcul.

2. Construire le LCP ne suffit pas. Il faut aussi le résoudre, ce qui est aussi coûteux.

Ces difficultés peuvent être assez facilement contournées. Pour la première, il suffit de transmettre le LCP de la boucle de simulation à la boucle haptique à chaque fois qu'il est calculé. La boucle haptique travaille donc avec une matrice de compliance du LCP constante, mais avec des distances d'interpénétrations variables. Le second problème peut être résolu facilement si l'utilisation d'un LCP identique à celui de la boucle de simulation est utilisé. En effet, la boucle de simulation résout le LCP. La solution de cette résolution constitue une excellente solution initiale pour le LCP de la boucle haptique, dans la mesure où les déplacements de l'utilisateur ne sont pas très importants. Partant de cette solution, la convergence peut se faire en seulement quelques pas.

Ce nouveau calcul des efforts appliqués dans l'interface nécessite donc de modifier sensiblement le schéma du couplage.

La nouvelle version est présentée dans la figure 5.14. Le calcul des forces dans la boucle haptique ne se fait plus à l'aide d'un ressort/amortisseur, mais à l'aide d'un LCP. Ce LCP est transmis par la simulation, et est résolu dans la boucle haptique à 1 KHz. Les forces ne dépendent plus de la position du God Objet, donc le frottement observé dans les méthodes de couplage classique disparaît.

L'interface haptique continue de fournir à la simulation une position à une fréquence de 30 Hz, tandis que la simulation ne fournit plus la position du God Object, mais le LCP à une fréquence de 30 Hz.

La stabilité de la simulation n'est pas compromise par cette modification, car le God Object assure toujours le respect des contraintes physiques.

5.5.2 Implantation

Nous allons voir à présent comment ce nouveau schéma peut être intégré à une simulation classique. Il n'y a finalement pas énormément de modification à apporter.

Transmission du LCP

Il nous faut dans un premier temps préciser quelles informations du LCP sont transmises. Il s'agit donc en réalité :

- Des parties de la compliance impliquées dans le calcul des forces de contact.
- De la liste des contraintes, c'est à dire de la liste des noeuds sur lesquelles elles s'appliquent, ainsi que les directions de ces contraintes, c'est à dire des tangentes dans le cas de frottement et des normales dans le cas de contacts.
- De la liste des distances d'interpénétration associées à ces contraintes.
- De la solution du LCP obtenue par la simulation, c'est à dire les forces permettant le respect des contraintes.
- Quelques paramètres divers et propres à la résolution, comme la précision désirée, le nombre d'itérations maximal, ...

Ensuite, le LCP doit non seulement être transmis à la boucle haptique, mais il faut encore s'assurer que sa mise à jour à la fréquence haptique n'impacte pas la simulation. Pour garantir cela, le plus simple est d'utiliser un *double buffer*. Ainsi, une fois que la simulation a terminé de construire le LCP et qu'elle l'a résolu, cette dernière *swap* ce LCP courant avec le second, et

transfert le second, qui est en fait l'ancien courant à la boucle haptique.

La boucle haptique est alors libre d'utiliser ce LCP tandis que la simulation reconstruit le second sans effet pour la boucle haptique.

Mise à jour du LCP

La boucle haptique dispose donc du dernier LCP construit et résolu. A sa charge maintenant de mettre à jour les contraintes de celui-ci en modifiant les distances d'interpénétrations à l'aide de la nouvelle position fournie par l'interface. Pour cela, il suffit d'augmenter les distances d'interpénétration du déplacement effectué par l'interface en utilisant l'opérateur $\boldsymbol{H}$ permettant de passer de l'espace des mouvements à celui des contacts :

$$\boldsymbol{\delta} = \boldsymbol{H}\Delta\boldsymbol{q} \tag{5.4}$$

Cette mise à jour change la configuration du LCP, et crée donc une situation nouvelle. La matrice du LCP, par contre, n'est pas modifiée. Le objet peut alors avoir renforcé son interpénétration ou au contraire ne plus être nullement en contact.

Résolution du LCP

La solution du LCP doit donc être recalculée. Il est donc résolu en quelques itérations en repartant de la solution initiale fournie pas la solution de la simulation à l'aide d'un Gauss Seidel non-linéaire adapté. Cet algorithme a été modifié en deux points :

1. Ce solveur ne commence pas ces itérations sur un vecteur nul, mais sur la solution obtenue dans la simulation. La cohérence des contacts au cours d'un pas de temps de simulation en fait une bonne approximation de la solution courante.
2. Ce solveur ne s'arrête pas selon un critère de convergence, mais selon un critère de temps imposé par la cadence du rendu haptique. Le fait de partir d'une solution initiale proche de la solution courante garanti un résultat réaliste.

Calcul des forces appliquées à l'interface

L'ensemble des forces est ensuite sommée et cette somme est appliquée à l'interface. Si l'interface possède un retour en rotation aussi, il faut calculer le couple créer par l'application de ces forces en différents points, et l'appliquer.

Algorithme complet

L'algorithme 15 regroupe toutes ces étapes.

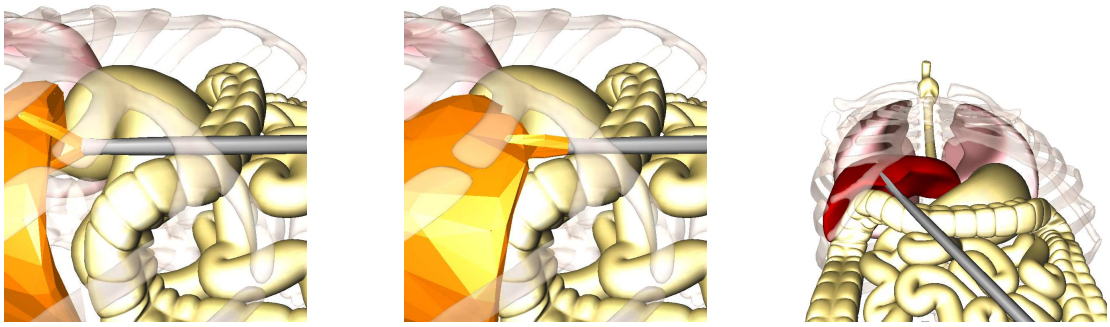
5.5.3 Application

Dans les exemples ci-dessous, nous utilisons un modèle corotationnel pour simuler la déformation non-linéaire d'un foie. Cette simulation a été exécutée sur un Intel bi-dual core 5130 @2Ghz

Algorithm 15 Évolution d'un système de particules

Ensure: Calcul les forces de contacts pour l'interface haptique.

1: $\boldsymbol{\delta} = \boldsymbol{H}\Delta\boldsymbol{q}$ *//$\Delta\boldsymbol{q}$ correspond au mouvement 6D de l'interface entre la position utilisée dans la simulation et la position courante.*
2: **for** chaque contrainte c **do**
3: $\boldsymbol{interpenetration_c} += \boldsymbol{\delta}$ *//correction en déplacement*
4: **end for**
5: Résolution par GAUSS-SEIDEL non-linéaire.
6: **for** chaque contrainte c **do**
7: $\boldsymbol{interpenetration_c} -= \boldsymbol{\delta}$ *//annulation de la correction en déplacement*
8: **end for**
9: **for** chaque contrainte c **do**
10: $\boldsymbol{f} += \boldsymbol{f}_c$
11: **end for**
12: Appliquer f à l'interface



(a) (b) (c)

FIG. 5.15 – L'interaction du forceps avec le foie. (a)le forceps touche juste la surface du foie (b) le forceps saisit le foie (c) Vue globale de la simulation

avec 2 Giga RAM. Un comportement visco-élastique est obtenu à l'aide d'un amortissement de type RAYLEIGH.

Dans ce premier benchmark, le modèle du foie contient environ 1300 tetraèdres. Nous utilisons ces contraintes de Dirichlet sur certains nœuds du maillage pour fixer le foie au reste du corps, qui est considéré rigide. Une meilleur modélisation des conditions aux limites devrait prendre en compte des données anatomiques relatives à la position des ligaments et des vaisseaux sanguins.

Dans le scénario présent (figure 5.15), le modèle du foie est manipulé avec une pince. Comme notre modèle gère la friction, nous pouvons saisir une partie du foie virtuel et tirer dessus.

Nous gérons ainsi jusqu'à 150 points de contacs, soit 450 contraintes tout en ayant une simulation tournant à 50 fps. Avec un pas d etemps de l'ordre de 20ms, un rendu haptique direct serait impossible. Cependant, comme nous avons une boucle haptique découplée de la boucle de simulation, notre rendu haptique est satisfaisant, dans la mesure où il tourne à une fréquence de 1Kz.

5.6 Conclusion

Ce chapitre a été l'occasion d'insister dans un premier temps sur l'importance de l'haptique dans les simulations. Le retour d'effort assure en effet une meilleure appréhension de l'environnement virtuel par l'utilisateur et plus encore accroît son efficacité.

Malheureusement, comme nous l'avons aussi détaillé, l'ajout de l'homme dans la boucle de simulation n'est pas trivial. Il pose de lourdes contraintes en terme de fréquence de rendu haptique, et introduit en plus une source d'instabilité qui implique l'ajout de couplage entre l'interface haptique et la simulation.

Or si nous avons vu que le problème du couplage a été bien traité, et que l'utilisation de méthodes de type God Object sont largement utilisées, il n'existe pas encore de solution générique pour permettre un retour haptique à la fréquence fatidique de 1Khz. Le seul recours à ce problème consiste généralement à faire au mieux pour augmenter la fréquence de la simulation en réduisant la complexité des modèles et des algorithmes.

Certains papiers proposent néanmoins dans le cas de solides non déformables ou poly-articulés de créer une boucle haptique découplée de la boucle physique.

De manière similaire, nous avons donc présenté dans ce chapitre une boucle haptique indépendante, tournant à une fréquence de 1Khz, qui utilise les informations de contact, le LCP et la valeur des forces de contatcs, fournis à une plus faible fréquence par la boucle physique. Cela autorise un rendu haptique assez réaliste puisqu'il respecte les lois de Signorini et Coulomb, avec l'avantage d'être assez générique puisque toute simulation résolvant ses contacts avec une méthode par contrainte pouvant fournir un LCP est compatible. Notre méthode peut donc être utilisée pour des objets rigides, poly-articulés, ou déformables.

5.7 Perspectives

Plusieurs améliorations peuvent être apportées à la méthode présentée ici. L'utilisation par exemple du Compliance Warping, reposant sur une matrice précalculée, interdit la découpe du matériau.

La simulation doit aussi être relativement rapide pour assurer une qualité de rendu satisfaisante. Une simulation trop lente rend le rendu moins agréable.

Enfin, il est nécessaire d'utiliser des LMD pour garantir une bonne cohérence des contacts au cours du temps. Il n'est pas envisageable d'utiliser d'autres méthodes de détection collision plus rapides.

Chapitre 6

Conclusion

Ce travail de thèse propose plusieurs améliorations pour la simulation de corps déformables. Ces améliorations ont été apportées pour obvier aux quelques limitations des méthodes actuelles observées en effectuant l'état de l'art et en s'appropriant ces méthodes.

En premier lieu, nous nous sommes intéressé à l'élément central d'un point de vue performance d'un simulateur physique : le solveur. Les solveurs permettent de résoudre les systèmes d'équations décrivant le comportement de nos solides déformables. Ces résolutions ne sont généralement pas triviales et demandent des solveurs complexes et gourmands en temps de calcul. Nous avons donc passer en revue les différents solveurs existants afin de déterminer lequel semblerait le mieux adapté à nos simulation. A l'issue de cette présentation, nous nous sommes aperçus que les solveurs multigrilles constituaient une bonne réponse.

Cependant, ces solveurs présentaient deux limitations, un coût de construction des matrices aux différentes résolutions important et l'incapacité d'approximer des géométries quelconques. Nous avons apportés à ces deux problèmes une solution. Ainsi, nous avons proposé un algorithme permettant de calculer rapidement les matrices aux diverses résolutions en accélérant le produit de matrices creuses. Et pour pallier à l'incapacité de modéliser des géométries complexes avec des solveurs hiérarchiques, nous avons proposé l'idée d'un mailleur hiérarchique qui tend au fur et à mesure des subdivisions à se rapprocher d'un surface cible complexe.

Les solveurs multigrilles, puisque basés sur l'utilisation d'une hiérarchie de grilles, semblent être à même de pouvoir tirer parti d'une analyse en ondeletttes. Il paraît en effet intéressant d'utiliser une telle transformée pour passer d'une résolution à une autre. Nous avons proposé dans cette thèse un premier *framework* basé sur le Lifting Scheme, permettant de construire une transformée en ondelettes 3D sur un maillage tétraèdrique. Cette transformée peut ensuite être utilisée comme opérateur de prolongation et de restriction dans un solveur hiérarchique.

Une fois le calcul de déformée rendu possible, il est naturel de chercher à permettre à nos objets virtuels d'interagir avec leur environnement. Aussi nous sommes nous intéressé au problèmes de la gestion des contacts. Nous avons ainsi constaté que les techniques employées pour résoudre les contacts ne satisfaisaient pas simultanément les contraintes de rapidité et de plausibilité physique requises pour une bonne immersion de l'utilisateur dans un environnement virtuel. D'un côté, en utilisant des méthodes du type pénalité ou impulsion, il est possible de résoudre efficacement les interpénétrations entre objets, en délaissant cependant toute considération physique. De l'autre, les méthodes par contraintes basées sur les lois de Signorini et Coulomb ont des fondements physiques, mais sont lentes et inadaptées à la simulation temps réel pour des objets autres que simples. C'est pourquoi nous avons développé notre méthode, le Compliance Warping, qui résout les contacts avec la compliance initiale de l'objet après les avoir ramenés dans la configuration initiale. Cette méthode assure ainsi une gestion des contacts rapide, respectant les lois de Signorini et Coulomb, et très proche de la solution exacte. En outre, bien qu'initialement conçue pour les modèles corotationnels élémentaires, pour lesquels nous avons prouvé

que l'approximation utilisait en fait un modèle corotationnel nodal approximant particulièrement bien le modèle élémentaire, elle peut être employée pour d'autres types d'objets, comme des solides poly-articulés. En utilisant cette méthode, il est possible de simuler de manière interactive et stable des objets déformables en grand déplacement soumis à de nombreux points de contacts, tout en respectant les lois de SIGNORINI et COULOMB, et ce sans avoir à régler de paramètres comme les raideurs dans le cas de méthode par pénalités.

Enfin, après avoir déformé nos objets et après leur avoir permis d'interagir avec leur environnement, nous avons voulu permettre à l'utilisateur d'interagir directement avec ces objets. Pour cela, nous avons identifié l'écueil posé par les simulations proposant un retour haptique : pour garantir un ressenti haptique qui puisse être perçu comme naturel pour l'utilisateur, il faut rafraîchir l'interface haptique à une fréquence de l'ordre du Khz, or les ordinateurs actuels ne sont pas suffisamment rapides pour fournir dans le cas de simulations déformables des forces à une telle fréquence. Face à cette impossibilité nous avons proposé de découpler la boucle haptique de la boucle physique. La boucle haptique tourne alors à une fréquence proche du Khz, tandis que celle physique peut tourner à une fréquence bien moindre. La boucle haptique à alors la charge de calculer uniquement les forces de contact, ce que nous proposons de faire très rapidement en résolvant le problème de complémentarité issue de la boucle physique, mis à jour à chaque pas de temps haptique par la nouvelle position de l'interface haptique. Nous obtenons ainsi un rendu haptique satisfaisant quand bien même la boucle physique tourne très lentement.

Toutes ces contributions nous ont ainsi permis de nous rapprocher sensiblement de notre objectif : la simulation interactive réaliste d'objets déformables. Notamment, nous sommes en mesure de déformer en grand déplacement des modèles contenant plusieurs centaines de tétraèdres, de gérer de nombreux contacts entre entre ces objets, mais aussi de proposer un retour haptique. Et tout cela suivant des bases physiques. A l'issue de ces travaux il apparaît qu'il est désormais possible de faire en grand déplacement ce qui était auparavant faisable uniquement en petit déplacement.

Il reste néanmoins de nombreuses améliorations à apporter. Déjà, d'un simple point de vue technologique, avec l'essor des nouveaux processeurs massivement parallèles, que ce soit au niveau de GPU, avec le framework CUDA de NVIDIA, ou au niveau CPU, avec le projet LARABEE d'INTEL. Ces changements drastiques d'architecture sont très prometteurs en terme de performance mais imposent de repenser en grande partie les algorithmes utilisés actuellement.

Une autre piste d'investigation possible réside en l'application des travaux des frères COSSERAT pour fournir un modèle éléments finis corotationnel qui soit géométriquement exact. En plus de suivre un modèle plus proche de la réalité, l'utilisation de cette théorie semble pouvoir mener à des optimisations des calculs, dues à la possibilité d'en effectuer une bonne par de manière analytique.

Les travaux menés sur le mailleur hiérarchique ne sont aussi qu'une amorce. Notamment, il faudrait améliorer la phase d'homogénéisation des tétraèdres à l'intérieur du volume de telle sorte qu'il soit possible d'améliorer la qualité du maillage et le conditionnement du système correspondant. Cela permettrait d'utiliser un autre biais pour optimiser les performances. Il semble aussi qu'il y ait une possibilité pour utiliser un MARCHING CUBE pour générer un maillage tétraèdrique facilement subdivisable et assurant un maillage particulièrement homogène, donc propice à un bon conditionnement.

Une autre difficulté, que nous avons rencontrée durant cette thèse, mais qui n'a pas été abordée dans le présent manuscrit, consiste à coupler différents modèles. Dans la simulation

médicale, par exemple, il est intéressant de coupler un système de cables 1D à un volume éléments finis 3D pour simuler un foie et le réseau de veines qui le vascularise. De même, dans l'industrie du contrôle non destructif, des capteurs en résine souple incluent des cables et des éléments rigides. Ces problèmes de couplage ont fait pour l'instant l'objet de peu d'études.

Toute ces améliorations, et bien d'autres, devraient permettre de rendre de plus en plus physique ces simulations. Ce réalisme, une fois la confiance du corps médical acquise, devait permettre des avancées spectaculaires dans le domaine de la santé, avec notamment le développement des méthodes Patient Specific qui prônent la préparation et l'accompagnement des opérations médicales à l'aide de simulateurs utilisant une modélisation précise et fidèle des organes du patient.

Appendices

ANNEXE A

Multiplicateurs de Lagrange

A.1 Les multiplicateurs de Lagrange

Principe L'importance des MULTIPLICATEURS DE LAGRANGE a été soulignée par BARAFF dans [Bar96]. L'idée sur laquelle MULTIPLICATEURS DE LAGRANGE repose est de respecter des contraintes en ajoutant des forces $\hat{\boldsymbol{f}}$ qui ne créent pas de travail. La démarche suivie pour calculer ces forces, ou MULTIPLICATEURS DE LAGRANGE suit.

Soit un système décrit par $\boldsymbol{M}\ddot{\boldsymbol{x}} = \boldsymbol{f}$ qui doit évoluer de manière à respecter les contraintes $\boldsymbol{C}(\boldsymbol{x}) = 0$, la dérivée de $C(\boldsymbol{x})$ par rapport au temps est :

$$\dot{\boldsymbol{C}}(\boldsymbol{x}) = \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{x}} = 0 \tag{A.1}$$

$\boldsymbol{J}$ étant le gradient de $\boldsymbol{C}$, ses lignes donnent la normale à l'espace des contraintes. Il faut donc se déplacer dans le plan orthogonal à $\boldsymbol{J}$ pour assurer que $\boldsymbol{C}(\boldsymbol{x})$ ne s'accroît pas et donc que $\boldsymbol{C}(\boldsymbol{x}) = 0$. C'est justement ce qu'exprime l'équation A.1, puisque $\dot{\boldsymbol{x}}$ est alors orthogonal à $\boldsymbol{J}$.

En dérivant une seconde fois par rapport au temps, il vient :

$$\ddot{\boldsymbol{C}}(\boldsymbol{x}) = \boldsymbol{J}\ddot{\boldsymbol{x}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{x}} = 0 \tag{A.2}$$

or $\ddot{\boldsymbol{x}} = \boldsymbol{M}^{-1}(\boldsymbol{f} + \hat{\boldsymbol{f}})$, d'où :

$$\boldsymbol{J}\boldsymbol{M}^{-1}\hat{\boldsymbol{f}} = -\dot{\boldsymbol{J}}\dot{\boldsymbol{x}} - \boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{f} \tag{A.3}$$

Les forces $\hat{\boldsymbol{f}}$ obtenues en résolvant l'équation A.3 assurent bien le respect de la contrainte $\boldsymbol{C}(\boldsymbol{x}) = 0$, mais ne sont pas passives. C'est pourquoi pour garantir que $\hat{\boldsymbol{f}}\dot{\boldsymbol{x}} = 0$ on impose que ces forces soient des combinaisons linéaires des lignes de $\boldsymbol{J}$, soit $\hat{\boldsymbol{f}} = \boldsymbol{J}^T\boldsymbol{\lambda}$, où $\boldsymbol{\lambda}$ sont les multiplicateurs de Lagrange.

Du coup, le système à résoudre permettant de respecter la contrainte $\boldsymbol{C}(\boldsymbol{x}) = 0$ et assurant la passivité des forces $\hat{\boldsymbol{f}}$ obtenues est :

$$\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T\boldsymbol{\lambda} = -\dot{\boldsymbol{J}}\dot{\boldsymbol{x}} - \boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{f} \tag{A.4}$$

et peut-être noté :

$$\left(\begin{array}{cc} \boldsymbol{M} & -\boldsymbol{J}^T \\ \boldsymbol{J} & 0 \end{array}\right) \left| \begin{array}{c} \dot{\boldsymbol{x}} \\ \boldsymbol{\lambda} \end{array} \right. = \left| \begin{array}{c} \boldsymbol{f} \\ -\dot{\boldsymbol{J}}\dot{\boldsymbol{x}} \end{array} \right. \tag{A.5}$$

Dans la pratique, il est intéressant de voir que la seconde ligne de cette équation A.6 impose directement une valeur à la vitesse $\dot{\boldsymbol{x}}$, la première ligne se chargeant d'assurer que cette contrainte

est bien compatible avec les équations du système simulé. Ce système, en plus de sa lisibilité, présente l'avantage d'être généralement creux ce qui permet une résolution plus efficace que celle obtenue en résolvant l'équation A.4. Ce type de système est généralisé en accord avec le modèle utilisé suivant que le paramètre d'état est l'accélération, la vitesse ou la position.

Ainsi, de manière générale, il vient :

$$\left(\begin{array}{cc} \boldsymbol{A} & -\boldsymbol{C}^T \\ \boldsymbol{C} & 0 \end{array}\right) \left| \begin{array}{c} \dot{\boldsymbol{\xi}} \\ \boldsymbol{\lambda} \end{array} \right. = \left| \begin{array}{c} \boldsymbol{f} \\ \boldsymbol{c} \end{array} \right. \tag{A.6}$$

où $\boldsymbol{\xi}$ est la variable d'état, $\boldsymbol{C}$ la matrice décrivant les contraintes, et $\boldsymbol{c}$ la valeur de ces contraintes. Le mouvement contraint est alors décrit par $\boldsymbol{A\xi} = \boldsymbol{f} + \boldsymbol{C}^T\boldsymbol{\lambda}$.

Application aux contacts La manière dont les multiplicateurs de Lagrange peuvent être calculés ayant été détaillée, il nous reste à poser les contraintes $\boldsymbol{C}(\boldsymbol{x}) = 0$ assurant la non-interpénétration.

Dans la théorie, si la contrainte est initialement respectée, *i.e.* si $\boldsymbol{C}(\boldsymbol{x}) = 0$, et si la variation de la vitesse dans le temps est nulle, *i.e.* si $\dot{\boldsymbol{C}}(\boldsymbol{x}) = 0$, alors respecter $\ddot{\boldsymbol{C}}(\boldsymbol{x}) = 0$, assure automatiquement le respect de $\boldsymbol{C}(\boldsymbol{x}) = 0$. C'est ce qu'affirme Witkin dans [WW90]. Ainsi, en imposant une accélération particulière aux degrés de liberté, il est possible de garantir le respect de tout type de contraintes.

Cependant, il est aussi possible de faire porter les contraintes sur les vitesses. Ainsi, dans [GOM+06], Galoppo *et al.* posent comme contrainte l'orthogonalité entre les vitesses au contact et les normales au contact. Ainsi, les points au contact se déplaçant dans un plan tangent à l'objet ne l'interpénètre jamais. La matrice de contrainte $\boldsymbol{N}$ a pour élément :

$$\boldsymbol{n}_{ii} = \boldsymbol{n}_i^T \tag{A.7}$$

où $\boldsymbol{n}_i$ est la normale au contact i, et le système contraint est régi par l'équation :

$$\left(\begin{array}{cc} \boldsymbol{A} & -\boldsymbol{N}^T \\ \boldsymbol{N} & 0 \end{array}\right) \left| \begin{array}{c} \boldsymbol{v} \\ \boldsymbol{\lambda} \end{array} \right. = \left| \begin{array}{c} \boldsymbol{f} \\ \boldsymbol{0} \end{array} \right. \tag{A.8}$$

où A est la matrice obtenue par un schéma d'nitégration implicite.

Autre possibilité, dans le cas où seulement la statique de l'objet est simulée : les contraintes peuvent porter directement sur la position des points au contact. C'est à dire que connaissant la distance d'interpénétration, la position que le point devrait occuper sur la surface de l'objet est calculée, et le point au contact est contraint dans cette position. Selon Cotin dans [CDA99], le système à résoudre a alors la forme :

$$\left(\begin{array}{cc} \boldsymbol{K} & -\boldsymbol{C}^T \\ \boldsymbol{C} & 0 \end{array}\right) \left| \begin{array}{c} \boldsymbol{u} \\ \boldsymbol{\lambda} \end{array} \right. = \left| \begin{array}{c} \boldsymbol{f} \\ \boldsymbol{u}^* \end{array} \right. \tag{A.9}$$

où $\boldsymbol{K}$ est la raideur du système, $\boldsymbol{u}$ le déplacement libre, et $\boldsymbol{u}^*$ le déplacement contraint. La matrice $\boldsymbol{C}$ contient des 0 partout, sauf pour ses éléments correspondant à des points dont le déplacement est imposé, où elle contient des 1.

A noter que ces méthodes peuvent être employées pour d'autre type de contraintes [Bar96], mais aussi pour coupler différents systèmes basés sur différents modèles [LF04].

Discussion Le problème des méthodes utilisant les MULTIPLICATEURS DE LAGRANGE est qu'elles ne gèrent que des contraintes bilatérales, *i.e.* des égalités. Ainsi, aussi bien dans le cas statique que dynamique, des effets d'adhérence peuvent apparaître. En effet, en statique, les points en contact sont contraints en position et restent donc collés à la surface. Dans le cas dynamique, les points voient leur vitesse contrainte à rester dans le plan orthogonal au contact ce qui crée aussi des phénomènes d'adhérence.

Ces méthodes présentent aussi des problèmes d'ordre numérique qui nécessitent l'utilisation de schémas de stabilisation [Bau72], ou l'ajout de force [WW90], sans quoi on observe une dérive de la solution.

Bibliographie

[AOW+08] Bart Adams, Maks Ovsjanikov, Michael Wand, Hans-Peter Seidel, and Leonidas J. Guibas. Meshless modeling of deformable shapes and their motion. In *SCA '08 : Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[APS98] M. Anitescu, F. Potra, and D. Stewart. Time-stepping for threedimensional rigid body dynamics. In *Computer Methods In Applied Mechanics And Engineering*, 1998.

[Bar96] David Baraff. Linear-time dynamics using lagrange multipliers. In *SIGGRAPH '96 : Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 137–146, New York, NY, USA, 1996. ACM.

[Bau72] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods Appl. Mech. Engng*, 1 :1–16, 1972.

[BBPB02] M. Bouzit, G. Burdea, G. Popescu, and R. Boian. The rutgers master ii-new design force-feedback glove. *Mechatronics, IEEE/ASME Transactions on*, 7(2) :256–263, Jun 2002.

[BCCG06] Liliana B. Boscardín, Liliana R. Castro, Silvia M. Castro, and Armando De Giusti. Wavelets bases defined over tetrahedra. *Journal of Computer Science & Technology*, 2006.

[Ben07] Jan Bender. Impulse-based dynamic simulation in linear time. *Comput. Animat. Virtual Worlds*, 18(4-5) :225–233, 2007.

[BFA02] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph.*, 21(3) :594–603, 2002.

[BHW94] David E. Breen, Donald H. House, and Michael J. Wozny. Predicting the drape of woven cloth using interacting particles. In *SIGGRAPH '94 : Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 365–372, New York, NY, USA, 1994. ACM.

[BJ05] Jernej Barbič and Doug L. James. Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM Trans. Graph.*, 24(3) :982–990, 2005.

[BPGK06] Mario Botsch, Mark Pauly, Markus Gross, and Leif Kobbelt. Primo : coupled prisms for intuitive surface modeling. In *SGP '06 : Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 11–20, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[BPWG07] Mario Botsch, Mark Pauly, Martin Wicke, and Markus Gross. Adaptive space deformations based on rigid cells. *Computer Graphics Forum*, 26(3) :339–347, 2007.

[BTH+03] Kiran S. Bhat, Christopher D. Twigg, Jessica K. Hodgins, Pradeep K. Khosla, Zoran Popović, and Steven M. Seitz. Estimating cloth simulation parameters from video. In *SCA '03 : Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 37–51, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[Bur96] Grigore Burdea. *Force and Touch Feedback for Virtual Reality*. John Wiley & Sons, New York, 1996.

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *SIGGRAPH '98 : Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, USA, 1998. ACM.

[BWS98] G. Baciu, Wingo Sai-Keung Wong, and Hanqiu Sun. Recode : an image-based collision detection algorithm. *Computer Graphics and Applications, 1998. Pacific Graphics '98. Sixth Pacific Conference on*, pages 125–133, Oct 1998.

[BYM05] Nathan Bell, Yizhou Yu, and Peter J. Mucha. Particle-based simulation of granular materials. In *SCA '05 : Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 77–86, New York, NY, USA, 2005. ACM.

[CBP05] Simon Clavet, Philippe Beaudoin, and Pierre Poulin. Particle-based viscoelastic fluid simulation. In *SCA '05 : Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 219–228, New York, NY, USA, 2005. ACM.

[CDA99] S. Cotin, H. Delingette, and N. Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *Visualization and Computer Graphics, IEEE Transactions on*, 5(1) :62–73, Jan-Mar 1999.

[CK05] Min Gyu Choi and Hyeong-Seok Ko. Modal warping : real-time simulation of large rotational deformation and manipulation. *Visualization and Computer Graphics, IEEE Transactions on*, 11(1) :91–101, Jan.-Feb. 2005.

[CMT02] F. Cordier and N. Magnenat-Thalmann. Real-time animation of dressed virtual humans. *Computer Graphics Forum*, 21(3) :327–336, September 2002.

[CSB95] J.E. Colgate, M.C. Stanley, and J.M. Brown. Issues in the haptic display of tool use. *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, 3 :140–145 vol.3, Aug 1995.

[CTA+08] Olivier Comas, Zeike Taylor, Jérémie Allard, Sébastien Ourselin, Stéphane Cotin, and Josh Passenger. Efficient nonlinear fem for soft tissue modelling and its gpu implementation within the open source framework sofa. In *International Symposium on Computational Models for Biomedical Simulation*, Lecture Notes in Computer Science (LNCS). Springer, july 2008.

[CWZ+04] Wei Chen, Huagen Wan, Hongxin Zhang, Hujun Bao, and Qunsheng Peng. Interactive collision detection for complex and deformable models using programmable graphics hardware. In *VRST '04 : Proceedings of the ACM symposium on Virtual reality software and technology*, pages 10–15, New York, NY, USA, 2004. ACM.

[DDCB01] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *SIGGRAPH '01 : Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 31–36, New York, NY, USA, 2001. ACM.

[DDKA06] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot. Realistic haptic rendering of interacting deformable objects in virtual environments. *Visualization and Computer Graphics, IEEE Transactions on*, 12(1) :36–47, Jan.-Feb. 2006.

[DG96] Mathieu Desbrun and Marie-Paule Gascuel. Smoothed particles : a new paradigm for animating highly deformable bodies. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96*, pages 61–76, New York, NY, USA, 1996. Springer-Verlag New York, Inc.

[DMG05] J. Dequidt, D. Marchal, and L. Grisoni. Time-critical animation of deformable solids : Collision detection and deformable objects. *Comput. Animat. Virtual Worlds*, 16(3-4) :177–187, 2005.

[EGS03] Olaf Etzmuss, Joachim Gross, and Wolfgang Strasser. Deriving a particle system from continuum mechanics for the animation of deformable objects. *IEEE Transactions on Visualization and Computer Graphics*, 9(4) :538–550, 2003.

[FBAF08] François Faure, Sébastien Barbier, Jérémie Allard, and Florent Falipou. Image-based collision detection and response between arbitrary volumetric objects. In *ACM Siggraph/Eurographics Symposium on Computer Animation, SCA 2008, July, 2008*, Dublin, Irlande, July 2008.

[Flo] Michael S. Floater. Mean value coordinates.

[FLS63] Richard Feynman, Robert Leighton, and Matthew Sands. *The Feynmen Lectures on Physics : Volume 1*, volume 1 of *The Feynman Lectures on Phsyics*. Addison-Wesley, Boston, 2nd edition edition, 1963.

[FLS05] Richard P. Feynman, Robert B. Leighton, and Matthew Sands. *The Feynman Lectures on Physics including Feynman's Tips on Physics : The Definitive and Extended Edition*. Addison Wesley, August 2005.

[FPS96] G. Fernández, S. Periaswamy, and Wim Sweldens. LIFTPACK : A software package for wavelet transforms using lifting. In M. Unser, A. Aldroubi, and A. F. Laine, editors, *Wavelet Applications in Signal and Image Processing IV*, pages 396–408. Proc. SPIE 2825, 1996.

[GD04] Stéphane Guy and Gilles Debunne. Monte-carlo collision detection. Technical Report RR-5136, INRIA, March 2004.

[GKLM07] Naga K. Govindaraju, Ilknur Kabul, Ming C. Lin, and Dinesh Manocha. Fast continuous collision detection among deformable models using graphics processors. *Comput. Graph.*, 31(1) :5–14, 2007.

[GKS02] Eitan Grinspun, Petr Krysl, and Peter Schröder. Charms : a simple framework for adaptive simulation. *ACM Trans. Graph.*, 21(3) :281–290, 2002.

[GOM+06] Nico Galoppo, Miguel A. Otaduy, Paul Mecklenburg, Markus Gross, and Ming C. Lin. Fast simulation of deformable models in contact using dynamic deformation textures. In *SCA '06 : Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 73–82, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[GRLM03] Naga K. Govindaraju, Stephane Redon, Ming C. Lin, and Dinesh Manocha. Cullide : interactive collision detection between complex models in large environments using graphics hardware. In *HWWS '03 : Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 25–32, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[GvL96] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[GW06] Joachim Georgii and Rüdiger Westermann. A multigrid framework for real-time simulation of deformable bodies. *Computers & Graphics*, 30(3) :408–415, 2006.

[HA96] V. Hayward and O. Astley. Performance measures for haptic interfaces. In *Robotics Research : The 7th International Symposium, Springer Verlag.*, pages 195–207, 1996.

[HFS+01] G. Hirota, S. Fisher, A. State, C. Lee, and H. Fuchs. An implicit finite element method for elastic solids in contact. *Computer Animation, 2001. The Fourteenth Conference on Computer Animation. Proceedings*, pages 136–254, 2001.

[HHK08] Woosuck Hong, Donald H. House, and John Keyser. Adaptive particles for incompressible fluid simulation. *Computer Graphics International*, June 2008.

[HLB+06] Jin Huang, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. An efficient large deformation method using domain decomposition. *Comput. Graph.*, 30(6) :927–935, 2006.

[HS04] M. Hauth and W. Strasser. Corotational simulation of deformable solids. *Proc. WSCG. (2004)*, pages 137–145, 2004.

[HTG03] Bruno Heidelberger, Matthias Teschner, and Markus H. Gross. Real-time volumetric intersections of deforming objects. In *VMV*, pages 461–468, 2003.

[Hub95] P.M. Hubbard. Collision detection for interactive graphics applications. *Visualization and Computer Graphics, IEEE Transactions on*, 1(3) :218–230, Sep 1995.

[HW04] Nathan Holmberg and Burkhard C. Wünsche. Efficient modeling and rendering of turbulent water over natural terrain. In *GRAPHITE '04 : Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 15–22, New York, NY, USA, 2004. ACM.

[JC01] David E. Johnson and Elaine Cohen. Spatialized normal come hierarchies. In *I3D '01 : Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 129–134, New York, NY, USA, 2001. ACM.

[JCA06] L. Joussemet, A. Crosnier, and C. Andriot. Espions : A novel algorithm based on evolution strategy for fast collision detection. In *Eurohaptics*, Paris, France, 2006.

[JP99] Doug L. James and Dinesh K. Pai. Artdefo : accurate real time deformable objects. In *SIGGRAPH '99 : Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 65–72, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[JP02] D.L. James and D.K. Pai. Real time simulation of multizone elastokinematic models. *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 1 :927–932 vol.1, 2002.

[JP03] Doug L. James and Dinesh K. Pai. Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Trans. Graph.*, 22(1) :47–82, 2003.

[JP04] Doug L. James and Dinesh K. Pai. Bd-tree : output-sensitive collision detection for reduced deformable models. *ACM Trans. Graph.*, 23(3) :393–398, 2004.

[JS94] B. Jawerth and W. Sweldens. An overview of wavelet based multiresolution analyses. *SIAM Rev.*, 36(3) :377–412, 1994.

[JWC05] David E. Johnson, Peter Willemsen, and Elaine Cohen. A haptic system for virtual prototyping of polygonal models. In *SIGGRAPH '05 : ACM SIGGRAPH 2005 Courses*, page 84, New York, NY, USA, 2005. ACM.

[KSB06] Ernst Kruijff, Dieter Schmalstieg, and Steffi Beckhaus. Using neuromuscular electrical stimulation for pseudo-haptic feedback. In *VRST '06 : Proceedings of the ACM symposium on Virtual reality software and technology*, pages 316–319, New York, NY, USA, 2006. ACM.

[KSTK98] Yoshifumi Kitamura, Andrew Smith, Haruo Takemura, and Fumio Kishino. A real-time algorithm for accurate collision detection for deformable polyhedral objects. *Presence : Teleoper. Virtual Environ.*, 7(1) :36–52, 1998.

[KZ03] Jan Klein and Gabriel Zachmann. Time-critical collision detection using an average-case approach. In *VRST '03 : Proceedings of the ACM symposium on Virtual reality software and technology*, pages 22–31, New York, NY, USA, 2003. ACM.

[LAM01] Thomas Larsson and Tomas Akenine-Möller. Collision detection for continuously deforming bodies. In *Eurographics*, Aire-la-Ville, Switzerland, Switzerland, 2001. Eurographics Association.

[LF04] Julien Lenoir and Sylvere Fonteneau. Mixing deformable and rigid-body mechanics simulation. In *CGI '04 : Proceedings of the Computer Graphics International*, pages 327–334, Washington, DC, USA, 2004. IEEE Computer Society.

[LJF05] Rodrigo G. Luque, ao L. D. Comba Jo and Carla M. D. S. Freitas. Broad-phase collision detection using semi-adjusting bsp-trees. In *I3D '05 : Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 179–186, New York, NY, USA, 2005. ACM.

[MAC04] D. Marchal, F. Aubert, and C. Chaillou. Collision between deformable objects using fast-marching on tetrahedral models. In *SCA '04 : Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 121–129, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[MC95] Brian Mirtich and John Canny. Impulse-based simulation of rigid bodies. In *SI3D '95 : Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 181–ff., New York, NY, USA, 1995. ACM.

[MDM+02] Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. Stable real-time deformations. In *SCA '02 : Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 49–54, New York, NY, USA, 2002. ACM.

[MG04] Matthias Müller and Markus Gross. Interactive virtual materials. In *GI '04 : Proceedings of Graphics Interface 2004*, pages 239–246, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[MHTG05] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless deformations based on shape matching. *ACM Trans. Graph.*, 24(3) :471–478, 2005.

[Mil88] Gavin S. P. Miller. The motion dynamics of snakes and worms. In *SIGGRAPH '88 : Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 169–173, New York, NY, USA, 1988. ACM.

[MKB+08] S. Martin, P. Kaufmann, M. Botsch, M. Wicke, and M. Gross. Polyhedral finite elements using harmonic basis functions. In *SGP08 : Proceedings of Eurographics Symposium on Geometry Processing*, pages 1521–1529, Copenhagen, Denmark, 2008. Computer Graphics Forum.

[MKN+04] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *SCA '04 : Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–151, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[MLM06] Servin Martin, Claude Lacoursiére, and Niklas Melin. Interactive simulation of elastic deformable materials. In *SIGRAD 2006 Conference Proceedings*, pages 22–32, Linköping, Sweden, 2006. Linköping University Electronic Press.

[Mur97] K.G. Murty. *Linear Complementarity, Linear and Nonlinear Programming*. Internet Edition, 1997.

[MW88] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. *SIGGRAPH Comput. Graph.*, 22(4) :289–298, 1988.

[MZK03] Andrea Mazzone, Rui Zhang, and Andreas Kunz. Novel actuators for haptic displays based on electroactive polymers. In *VRST '03 : Proceedings of the ACM symposium on Virtual reality software and technology*, pages 196–204, New York, NY, USA, 2003. ACM.

[NAM+06] Nealen, Andrew, Muller, Matthias, Keiser, Richard, Boxerman, Eddy, Carlson, and Mark. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4) :809–836, December 2006.

[OL05] Miguel A. Otaduy and Ming C. Lin. Introduction to haptic rendering. In *SIGGRAPH '05 : ACM SIGGRAPH 2005 Courses*, page 3, New York, NY, USA, 2005. ACM.

[PKA+05] Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré, Markus Gross, and Leonidas J. Guibas. Meshless animation of fracturing solids. *ACM Trans. Graph.*, 24(3) :957–964, 2005.

[RCG08] Guodong Rong, Yan Cao, and Xiaohu Guo. Spectral mesh deformation. In *CGI '08 : Proceedings of the Computer Graphics International*, 2008.

[Ree83] William T. Reeves. Particle systems—a technique for modeling a class of fuzzy objects. In *SIGGRAPH '83 : Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, pages 359–375, New York, NY, USA, 1983. ACM.

[RJ07] Alec R. Rivers and Doug L. James. Fastlsm : fast lattice shape matching for robust real-time deformation. In *SIGGRAPH '07 : ACM SIGGRAPH 2007 papers*, page 82, New York, NY, USA, 2007. ACM.

[RMB+08] E. Ruffaldi, D. Morris, F. Barbagli, K. Salisbury, and M. Bergamasco. Voxel-based haptic rendering using implicit sphere trees. *Haptic interfaces for virtual environment and teleoperator systems, 2008. haptics 2008. symposium on*, pages 319–325, March 2008.

[SCB04] K. Salisbury, F. Conti, and F. Barbagli. Haptic rendering : introductory concepts. *Computer Graphics and Applications, IEEE*, 24(2) :24–32, March-April 2004.

[SDC08] Guillaume Saupin, Christian Duriez, and Stéphane Cotin. Contact model for haptic medical simulations. In *International Symposium on Computational Models for Biomedical Simulation*, Lecture Notes in Computer Science (LNCS). Springer, july 2008.

[SDCG08] Guillaume Saupin, Christian Duriez, Stephane Cotin, and Laurent Grisoni. Efficient contact modeling using compliance warping. In *Computer Graphics International 2008*, 2008.

[SDG07] Guillaume Saupin, Christian Duriez, and Laurent Grisoni. Embedded multigrid approach for real-time volumetric deformation. In *International Symposium on Visual Computing*, pages 149–159, 2007.

[SF91] M. Shinya and M. Forgue. Interference detection through rasterization. In *Journal of Visualization and Computer Animation*, pages 132–134, 1991.

[SGwHS98] Jonathan Shade, Steven Gortler, Li wei He, and Richard Szeliski. Layered depth images. In *SIGGRAPH '98 : Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242, New York, NY, USA, 1998. ACM.

[SHW04] S. Schaefer, J. Hakenberg, and J. Warren. Smooth subdivision of tetrahedral meshes. In *SGP '04 : Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 147–154, New York, NY, USA, 2004. ACM.

[SLA06] Jean Sreng, Anatole Lecuyer, and Claude Andriot. Using visual cues of contact to improve interactive manipulation of virtual objects in industrial assembly/maintenance simulations. *IEEE Transactions on Visualization and Computer Graphics*, 12(5) :1013–1020, 2006. Member-Christine Megard.

[Smi97] Christopher M. Smith. Human factors in haptic interfaces. *Crossroads*, 3(3) :14–16, 1997.

[SOG08] Denis Steinemann, Miguel A. Otaduy, and M. Gross. Fast adaptive shape matching deformations. In *SCA '08 : Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[SS95] Peter Schroder and Wim Sweldens. Spherical wavelets : efficiently representing functions on the sphere. In *SIGGRAPH '95*, pages 161–172, New York, NY, USA, 1995. ACM Press.

[Swe98] Wim Sweldens. The lifting scheme : a construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2) :511–546, 1998.

[SYBF06] Lin Shi, Yizhou Yu, Nathan Bell, and Wei-Wen Feng. A fast multigrid algorithm for mesh deformation. *ACM Trans. Graph.*, 25(3) :1108–1117, 2006.

[TBHF03] J. Teran, S. Blemker, V. Ng Thow Hing, and R. Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *SCA '03 : Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 68–74, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[TF88] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation : viscolelasticity, plasticity, fracture. *SIGGRAPH Comput. Graph.*, 22(4) :269–278, 1988.

[THMG04] Matthias Teschner, Bruno Heidelberger, Matthias Muller, and Markus Gross. A versatile and robust model for geometrically complex deformable solids. In *CGI '04 : Proceedings of the Computer Graphics International*, pages 312–319, Washington, DC, USA, 2004. IEEE Computer Society.

[TKH+05] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. Collision detection for deformable objects, 2005.

[Ton92] David Tonnesen. Spatially coupled particle systems. In *SIGGRAPH '98 : Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, Chicago, Illinois, USA, 1992. ACM.

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *SIGGRAPH '87 : Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, New York, NY, USA, 1987. ACM.

[TPF91] Demetri Terzopoulos, John Platt, and Kurt Fleischer. Heating and melting deformable models. *The Journal of Visualization and Computer Animation*, 2, 91.

[TT94] Xiaoyuan Tu and Demetri Terzopoulos. Artificial fishes : physics, locomotion, perception, behavior. In *SIGGRAPH '94 : Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 43–50, New York, NY, USA, 1994. ACM.

[TW88] Demetri Terzopoulos and Andrew Witkin. Physically based models with rigid and deformable components. *IEEE Comput. Graph. Appl.*, 8(6) :41–51, 1988.

[vdB97] Gino van den Bergen. Efficient collision detection of complex deformable models using aabb trees. *J. Graph. Tools*, 2(4) :1–13, 1997.

[WHG08] T. Winkler, K. Hormann, and C. Gotsman. Mesh massage : A versatile mesh optimization framework. *The Visual Computer*, 2008. To appear.

[WT06] Rachel Weinstein and Joseph Teran. Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE Transactions on Visualization and Computer Graphics*, 12(3) :365–374, 2006. Member-Ron Fedkiw.

[WW90] Andrew Witkin and William Welch. Fast animation and control of nonrigid structures. In *SIGGRAPH '90 : Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 243–252, New York, NY, USA, 1990. ACM.

[WW04] Niniane Wang and Bretton Wade. Rendering falling rain and snow. In *SIGGRAPH '04 : ACM SIGGRAPH 2004 Sketches*, page 14, New York, NY, USA, 2004. ACM.

[ZS95] C. B. Zilles and J. K. Salisbury. A constraint-based god-object method for haptic display. In *IROS '95 : Proceedings of the International Conference on Intelligent Robots and Systems-Volume 3*, page 3146, Washington, DC, USA, 1995. IEEE Computer Society.

[ZY00] Dongliang Zhang and M.M.F. Yuen. Collision detection for clothed human animation. *Computer Graphics and Applications, 2000. Proceedings. The Eighth Pacific Conference on*, pages 328–337, 2000.