Animation multirésolution d'objets déformables en temps-réel. Application à la simulation chirurgicale

Gilles Debunne

Institut National Polytechnique de Grenoble

Thèse

pour obtenir le grade de

Docteur de l'INPG

Spécialité : Imagerie, Vision et Robotique

Préparée au sein du laboratoire *i*MAGIS-GRAVIR/IMAG-INRIA. UMR CNRS C5527
dans le cadre de l'École Doctorale Mathématiques, Sciences et Technologie de l'information, Informatique

présentée et soutenue publiquement par

Gilles DEBUNNE

le 15 décembre 2000.

Animation multirésolution d'objets déformables en temps-réel

Application à la simulation chirurgicale

Directrice de thèse : Marie-Paule CANI

Composition du jury :

Roger	MOHR	Président
Bruno	ARNALDI	Rapporteur
Daniel	THALMANN	Rapporteur
Christophe	CHAILLOU	Examinateur
Mathieu	DESBRUN	
Alan H.	BARR	
Marie-Paule	CANI	

SOMMAIRE

Notations

Nous allons être amenés à utiliser bon nombre d'équations dans cette thèse. Nous nous sommes efforcés de suivre un système de notations qui devrait en simplifier la lecture.

Les grandeurs scalaires seront désignées en italiques (s), les vecteurs et champs vectoriels, de dimension 3 dans notre cas, seront en gras ($\mathbf{v}$) tandis que les matrices et autres tenseurs seront représentés par des lettres grecques ou calligraphiques (ε, A).

Les exposants permettront d'identifier les points où est mesurée la grandeur : $\mathbf{p}^i, \mathbf{v}^i, \mathbf{a}^i$ seront respectivement les position, vitesse et accélération du point i.

Les indices désigneront quant à eux la composante : v_i est la $i^{ème}$ composante du vecteur $\mathbf{v}$. Les lettres i et j seront des indices muets prenant implicitement les valeurs 1, 2 ou 3. On pourra aussi écrire directement v_x, v_y, v_z pour désigner les trois composantes de $\mathbf{v}$.

Nous utiliserons pour désigner les dérivées d'une grandeur scalaire s par rapport à la variable x l'expression $s_{,x}$. Ainsi $v_{x,y} = \frac{\partial v_x}{\partial y}$. La notation s'étend aux vecteurs et $\mathbf{v}_{,y}$ sera le vecteur dont chaque composante est la dérivée par rapport à y de la composante associée de $\mathbf{v}$.

L'annexe A rappelle quelques notions sur les opérateurs vectoriels (**grad** , div , ...) et introduit des notations plus poussées. Des notes en bas de page indiquent quand s'y reporter durant la lecture du document.

Enfin, nous utiliserons les *italiques* à l'intérieur du texte pour introduire de nouvelles terminologies ou pour accentuer l'importance de certains mots.

Introduction

Telle est ma quête,
suivre l'étoile.
Peu m'importent mes chances,
Peu m'importe le temps.

Jacques BREL, *La quête*

L'ANIMATION de scènes en synthèse d'images est une technique maîtrisée, largement utilisée pour les effets spéciaux cinématographiques et les jeux vidéos. Elle se limite toutefois la plupart du temps à l'animation du déplacement d'objets rigides. Chercher, comme on va le faire dans cette thèse, à calculer *automatiquement* les déformations d'un objet, qui plus est en *temps-réel*, est un problème encore ouvert. Il faut pour cela mettre en œuvre des optimisations, dont la multirésolution fait partie.

Contexte

Quand on parle d'animation, aussi bien dans les jeux vidéos que dans l'industrie cinématographique, on désigne généralement la création du mouvement d'un personnage ou d'un objet qui va se déplacer dans la scène. Animer des personnages ou des objets est un véritable métier et consiste habituellement à générer les courbes de contrôle qui vont définir la trajectoire (l'angle d'une articulation du personnage ou la position d'une main au cours du temps), tout en y faisant passer l'émotion, les expressions, les mimiques voulues par le scénario [Las87]. C'est un travail fastidieux qui demande une longue pratique avant d'être maîtrisé et de nombreux essais pour arriver au résultat voulu.

L'animation par *modèle physique* consiste à générer *automatiquement* le mouvement d'un objet, de façon réaliste, en simulant les lois physiques qui gouvernent son mouvement. On parle alors de modèles *générateurs* par opposition aux procédés *descriptifs* décrits auparavant. Ces techniques, moins malléables que les précédentes, commencent à se développer dans l'industrie. Ainsi dans le dernier[1] épisode de *Star Wars*, les trajectoires des milliers de débris qui volent et s'éparpillent après le crash d'un des engins de la course dans le désert ont été calculés directement. L'animation de la veste de Geri dans *Geri's game* est elle aussi calculée automatiquement d'après les mouvements du personnage [BW98, Rob98b], tout comme celle de l'eau dans *Fourmiz* [FM96, Rob98a]. Notons que les réalisateurs, qui cherchent plus un effet visuel qu'une simulation exacte, veulent pouvoir modifier le résultat de la simulation pour obtenir telle ou telle caractéristique, ce qui ne peut se faire qu'en contraignant la simulation [PSE+00, CF00].

[1] Premier pour les puristes.

Cette thèse porte sur l'animation par modèle physique d'objets *déformables* et non plus rigides. Ce domaine est étudié depuis plus de 10 ans mais n'est pas encore très utilisé dans l'industrie audiovisuelle où ces objets sont assez rares et où il est plus commode de générer manuellement leur mouvement.

Il n'en est pas de même lorsqu'un utilisateur veut pouvoir *interagir* avec l'objet et voir immédiatement le résultat de ses actions sur l'objet, comme ce pourrait être le cas dans un jeu vidéo. Le coût des calculs à mettre en œuvre serait alors prohibitif et on aurait probablement recours à des séquences précalculées, ce qui limiterait les mouvements possibles et le réalisme.

On ne peut se permettre de telles simplification dans une application médicale temps-réel comme un simulateur chirurgical et il faut donc mettre en place de nouvelles solutions.

Motivations

On sait calculer très précisément les déformations d'un objet complexe, et c'est par exemple ainsi que sont mises au point les ailes des avions, pour lesquelles le test grandeur nature validant leur conception n'intervient qu'en fin d'étude pour vérifier les calculs. Mais de telles simulations sont très gourmandes en temps de calcul, même sur des ordinateurs très puissants, et l'on est à plusieurs ordres de grandeur d'une simulation ne serait-ce qu'interactive. On différenciera le *temps-réel* qui garantit le respect de la synchronisation entre la simulation et l'affichage et les simulations *interactives* qui assurent seulement une fréquence de rafraîchissement de plusieurs hertz (au moins 15 ou 20 pour une animation fluide). Nous reviendrons sur ces définitions à la Section 11.2 du chapitre suivant.

Le contexte applicatif de cette thèse est celui de la réalisation d'un simulateur chirurgical d'opérations laparoscopiques[2]. Ce type d'opérations, en très fort développement actuellement, consiste à opérer le patient à l'aide d'outils et d'une caméra introduits au travers de petites incisions, en suivant l'opération sur un écran. Cette technique a de nombreux avantages : minimisant le traumatisme postopératoire, elle convient au patient comme à l'hôpital qui peut le libérer plus rapidement et ainsi faire des économies.

Le problème est que cette technique est assez peu intuitive et d'un apprentissage difficile pour le chirurgien : il n'a plus la notion de profondeur née de la stéréoscopie et l'outil étant axé sur l'abdomen du patient, il lui faut par exemple le déplacer à gauche pour qu'il aille à droite, et encore, si la caméra est horizontale. Cette technique requiert de nombreuses opérations (on estime leur nombre à une cinquantaine) avant que le chirurgien ne puisse l'utiliser en salle d'opération, les entraînements se faisant sur cadavres ou sur cochons, et finalement, dans une certaine mesure, sur les premiers patients.

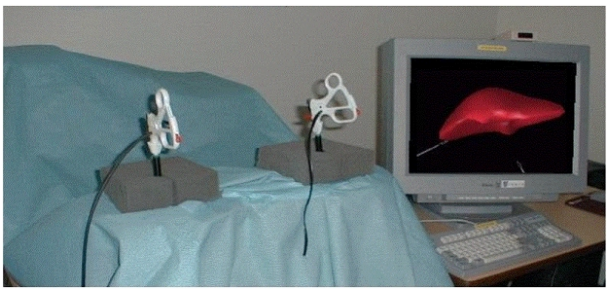


FIG. 1: Équipement d'une salle de simulation laparoscopique ©Épidaure - INRIA.

L'intérêt d'un simulateur informatique est alors évident. Le chirurgien ne regardant durant l'opération que son écran de contrôle, il s'agira simplement de générer en images de synthèses le comportement d'organes virtuels réagissant aux gestes du chirurgien. Ceux-ci seront transmis à l'ordinateur par un appareil doté de capteurs de position et qui pourra de plus exercer sur l'instrument manipulé un retour d'effort, commandé par l'ordinateur, reproduisant ainsi les forces naissant du contact entre l'outil et l'organe virtuels.

Outre son intérêt éthique, un tel simulateur peut être enrichi de fonctionnalités pédagogiques, en proposant par exemple une série d'exercices de difficulté croissante, voire la simulation de situations d'urgence, en mesurant l'efficacité du chirurgien à les gérer. Il est également viable sur le plan économique car utilisable en permanence et ne nécessitant pas d'équipement particulier. Enfin, utilisé avec des données provenant d'un

[2]On parle aussi de péritonéoscopie.

patient réel, il permettrait un entraînement pour une opération difficile et spécifique. On pourra se reporter à `www.spectrum.ieee.org/pubs/spectrum/0700/surg.html` pour d'autres détails sur la laparoscopie.

On se place donc ici dans un contexte d'application temps-réel, ce qui exigera des concessions sur la qualité de la simulation produite. Pour optimiser l'utilisation du temps de calcul, on se propose de mettre en place une méthode tirant parti de la *multirésolution*.

Simulation multirésolution

Le terme de multirésolution, qui n'existe pas officiellement dans la langue française, désignera dans cette thèse le fait de simuler un même phénomène (en l'occurrence l'animation d'un objet déformable) à plusieurs échelles. Le but des approches multirésolution est de choisir, au bon moment, au bon endroit, quelle est l'échelle qui donne le meilleur compromis qualité - rapidité de calcul.

Cette technique est très largement répandue dans cet autre domaine de la synthèse d'images qu'est la *radiosité hiérarchique*. L'éclairage des éléments d'une scène est représenté à plusieurs résolutions, ce qui permet par exemple de subdiviser précisément un grand mur à l'endroit où une ombre est projetée pour avoir des détails, mais de ne considérer que le mur dans son ensemble, en ayant perdu par moyenne l'information d'ombre, lorsqu'il s'agit de calculer quelle quantité de lumière il reflète sur un objet lointain. On perd ici en précision ce que l'on gagne en temps de calcul en évitant de calculer toutes les interactions entre les parties du mur et les objets distants.

Dans le cadre de l'animation d'objets déformables, la multirésolution va consister à mélanger les différentes *discrétisations* d'un même objet. Celui-ci va en effet être représenté, non pas comme un milieu continu, mais comme un ensemble discret de points, plus ou moins nombreux. Un très grand nombre de points générera un mouvement très subtil, chaque petite zone ayant son comportement propre, au prix de très longs temps de calcul. A l'opposé, peu de points seront simulés rapidement, mais le mouvement sera plus grossier. On a intérêt à combiner ces simulations : les parties proches de l'outil, qui subissent de fortes déformations et sur lesquelles le regard est concentré méritent une simulation précise. Au contraire, les parties lointaines peuvent avoir un mouvement très grossier sans trop altérer la qualité de la simulation. En utilisant différentes discrétisations (on parlera aussi de différentes *résolutions*), on optimisera l'utilisation du temps de calcul imparti entre deux affichages successifs de l'animation. On se référera à la Section 10 du chapitre suivant pour des explications plus détaillées ainsi que pour l'introduction de quelques définitions associées.

Objectifs de cette thèse

Cette thèse propose plusieurs approches d'animation multirésolution qui chercheront à atteindre plusieurs objectifs :

- Le *réalisme visuel* est le premier d'entre eux. Il tolère certaines imperfections dans l'animation dont la multirésolution va tirer parti, mais requiert néanmoins, dans un contexte chirurgical, un réalisme physique important. Notons que la partie réaliste du *rendu* de l'organe ne fait pas directement partie de nos objectifs.
- La garantie d'une *animation temps-réel* est l'autre difficulté de ces travaux. On ne parle pas seulement ici de fréquence d'affichage élevée, mais aussi de corrélation entre le temps ressenti par l'utilisateur et celui simulé par la machine : entre deux images affichées à une seconde d'intervalle, la machine doit effectivement avoir calculé le comportement du matériau durant une seconde, ni plus ni moins. Cet objectif est bien évidemment antagoniste du précédent et il faudra donc faire des compromis.
- Dans une moindre mesure, les modèles proposés devront être *aisément paramétrables*. On veut en particulier pouvoir utiliser les mesures qui peuvent être faites sur différents matériaux réels et les donner en paramètres au modèle pour en simuler le comportement.
- Le simulateur que l'on va créer devra également être suffisamment *générique* en ce sens qu'on peut vouloir l'utiliser pour modéliser un grand nombre de comportements différents, sur une grande variété d'objets.

Organisation du document

Ce document est divisé en 6 chapitres. Le premier présente un état de l'art des méthodes existant actuellement en animation de modèles déformables. Après une description générale, nous nous intéresserons plus particulièrement aux modèles permettant une simulation interactive ainsi qu'à ceux utilisant des modèles multirésolution.

Le Chapitre 2 décrit les modèles mis au point par les physiciens pour décrire les déformations des matériaux continus, que l'état de l'art aura montrés comme adaptés à nos buts.

Le Chapitre 3 propose un premier algorithme multirésolution utilisant ces modèles et basé sur une découpe en octree de l'espace.

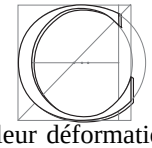
De nouvelles façons de résoudre les équations différentielles impliquées sont présentées et discutées au Chapitre 4. Une utilisation multirésolution de ces formules est détaillée au Chapitre 5 avec l'introduction d'une nouvelle méthode permettant l'utilisation conjointe de plusieurs maillages volumiques d'un objet.

Le Chapitre 6 explique les problèmes liés à l'interface : comment lier la surface de l'objet au modèle simulé et transmettre un retour d'effort à l'utilisateur.

Nous concluons par un résumé des contributions et par des pistes de recherche future, suivi de trois annexes détaillant certaines des notions utilisées dans le document.

CHAPITRE 1

Travaux antérieurs



E CHAPITRE présente quelques unes des méthodes qui ont été développées dans le domaine de l'animation en synthèse d'images pour animer des objets déformables. Le but étant d'avoir un effet visuel, on pourra définir comme déformables des objets suffisamment mous pour que leur déformation soit visible et intéressante. Nous avons regroupé les différentes approches en fonction de la méthode de résolution employée, et nous en décrirons à chaque fois les avantages, les inconvénients et l'adéquation à nos buts.

Après une revue plus exhaustive des méthodes permettant des applications temps-réel ainsi que de celles mettant en œuvre des processus multirésolution, buts visés par cette thèse, nous justifierons le choix des méthodes employées par la suite.

D'autres descriptions des modèles déformables existants peuvent être trouvés dans l'état de l'art réalisé en 1997 par Sarah Gibson et Brian Mirtich [GM97] et dans les thèses de Mathieu Desbrun [Des97] et Stéphane Cotin [Cot97] qui portent sur le sujet. Nous ne traitons pas ici le cas des objets déformables purement surfaciques comme les tissus et le lecteur intéressé pourra trouver un état de l'art plus complet dans [NG96].

1 Déformations géométriques de la surface

Cette première classe d'algorithmes est un peu à part. Elle ne considère pas l'objet déformable comme un tout avec des propriétés physiques intrinsèques, mais considère plutôt la surface qui le délimite et que l'on va chercher à animer. Ces méthodes, particulièrement employées dans le design, sont néanmoins intéressantes pour nous car elles seront par la suite reprises dans certains algorithmes plus "physiques".

1.1 Surfaces de forme libre

Historiquement, la conception géométrique assistée par ordinateur (CAO) fut l'un des premiers domaines à utiliser des techniques de déformation d'objets. Les designers voulaient pouvoir manipuler et affiner la représentation numérique de leur objet. De là sont nées les courbes de Bézier et la famille des *splines* : courbes d'interpolation, B-splines, splines rationnelles, non uniformes (NURBS), toutes définissant avec peu de valeurs une courbe lisse, en 1, 2 ou 3 dimensions (voir [BBB87, Far90] pour une étude plus complète). Ces courbes et les surfaces associées seront généralement définies par des *points de contrôle* dont le déplacement modifiera

l'aspect de la courbe de manière intuitive (voir Fig. 1.1). On peut aussi déplacer directement la courbe ou la surface B-spline plutôt que les points de contrôle [BB89].

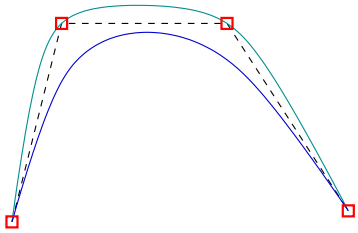
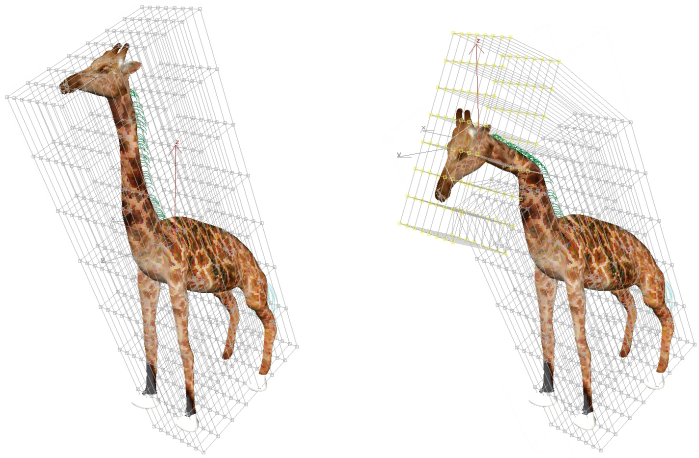


FIG. 1.1: Quatre points de contrôle, la spline d'interpolation qui passe par eux (à l'extérieur) ainsi que la courbe de Bézier qu'ils définissent, tangentes au départ et à l'arrivée au polygone de contrôle (à l'intérieur).

Des approches multirésolution à base d'ondelettes ont été développées pour permettre de modifier la courbe, ou par extension la surface, de manière précise et locale ou au contraire dans sa forme globale [FS94].

1.2 Les déformations de l'espace

Une autre façon de modifier globalement la forme d'un objet est de déformer *l'espace* dans lequel il réside. On peut par exemple remplacer la coordonnée x par $-x$ pour effectuer une symétrie. Les transformations appliquées peuvent être bien plus complexes et ont l'avantage, en déformant tout l'espace, de pouvoir s'appliquer aussi bien à des surfaces polygonales, splines, paramétriques qu'implicites. La Figure 1.2 illustre cette idée.



(a) (b)

FIG. 1.2: Déformation de l'espace illustrée sur un modèle de girafe : la grille originale (a) et celle obtenue après une déformation localisée (b).

Ces travaux furent débutés par Al Barr [Bar84] et consistaient alors en des fonctions matricielles de $\mathbb{R}^3$ dans $\mathbb{R}^3$ (provoquant des déformations rigides, des pliages, des effilements) qui pouvaient être combinées pour donner des effets complexes.

Sederberg et Parry ont généralisé cette approche [SP86] par l'introduction des FFD, pour *Free-Form Deformation*. L'objet est plongé dans un maillage qui va pouvoir être déformé, chaque cellule du maillage entraînant avec elle les points qu'elle contient au moyen d'interpolations utilisant habilement les polynômes de Bernstein. Des extensions, tirant parti de la séparation en plusieurs sous-régions du maillage déformant [Coq90] ou utilisant la multirésolution [MJ96] (le maillages pouvant être de topologie arbitraire grâce à utilisation d'algorithmes de subdivision) furent présentées.

Citons aussi une méthode permettant d'inférer par l'utilisation de moindres carrés la modification du maillage directement de celle que l'on veut faire subir aux points de la surface [HHK92].

1.3 Surfaces implicites

La surface d'un objet peut également être représentée *implicitement*, plutôt qu'*explicitement* à l'aide de polygones ou de splines. On crée pour cela un champ scalaire dans tout l'espace, et la surface est définie

comme le lieu où ce champ a une valeur donnée. Ces champs dérivent généralement de potentiels et l'on parle de *surface équipotentielle* ou d'*iso-surfaces*. Ce potentiel est souvent une fonction dépendant de façon monotone de la distance à un squelette, qui peut être un point, un segment ou toute autre forme plus complexe [BW90].

Cette formulation a de nombreuses propriétés intéressantes : selon que la valeur du champ en un point de l'espace a une valeur inférieure ou supérieure à l'iso-valeur, on sait si l'on se trouve à l'intérieur ou à l'extérieur de l'objet, ce qui permet notamment une détection des collisions très rapide. La combinaison de plusieurs objets implicites peut être très simple en sommant les différents potentiels qui les génèrent, ce qui permet également la gestion d'objets à topologie variable, l'iso-surface se divisant naturellement lorsque les deux squelettes se séparent.

La visualisation de telles surfaces est par contre plus complexe, leur polygonalisation pouvant en particulier s'avérer difficile. On pourra utiliser l'algorithme du *marching cube* [WMW86, Blo87, Blo88], des maillages locaux [DTG96] ou la méthode à base de particules de Witkin et Heckbert [WH94] pour les afficher.

Ce formalisme définit la surface d'un objet comme issue de potentiels. Son animation va alors consister à animer les squelettes qui la génèrent, la surface se modifiant en conséquence avec d'intéressantes propriétés de continuité ou de changements de topologie. On ne modifiera néanmoins que rarement manuellement la position de ces squelettes, qui sont généralement des masses ponctuelles, préférant laisser cette tâche à des modèles générateurs, comme on le verra par la suite.

1.4 Conclusion

Ces méthodes géométriques ont été surtout présentées comme une introduction car certaines seront reprises par la suite. Elles ont l'avantage d'être généralement très rapides et offrent un grand contrôle sur la déformation produite. Elles ne peuvent par contre s'appliquer directement au cas qui nous préoccupe puisque le mouvement est directement créé par l'utilisateur et que c'est de son habileté que dépendra le résultat.

On utilisera donc plutôt des modèles générateurs, qui simuleront des lois de comportement, plus ou moins physiques, et généreront donc *automatiquement* le mouvement de l'objet. Il existe beaucoup de méthodes génératrices et nous allons en décrire les principales familles dans ce qui suit.

2 Déformations globales

Ces méthodes déforment l'objet *globalement* en le considérant comme un tout mais en utilisant maintenant des lois inspirées de la physique pour générer son mouvement.

2.1 Dynamique modale

L'algorithme proposé par Pentland et Williams tire parti de l'analyse des *modes vibratoires* de l'objet [PW89]. Les lois physiques de l'élasticité linéaire conduisent à une équation du second ordre classique du type :

$$M\frac{\partial^2\mathbf{u}}{\partial t^2}+D\frac{\partial\,\mathbf{u}}{\partial t}+K\,\mathbf{u}=\mathbf{f} \tag{1.1}$$

où M, D et K sont des matrices représentant respectivement la masse, l'amortissement et la raideur du matériau. $\mathbf{u}$ mesure le déplacement des différents nœuds de discrétisation et $\mathbf{f}$ est la force qui y est appliquée. Ces trois matrices sont symétriques, et sous certaines hypothèses (D doit en particulier être égale à M à un coefficient multiplicateur près), sont diagonalisables dans la même base. L'équation précédente se simplifie alors grandement et devient, en notant avec un tilde l'expression des variables dans la nouvelle base :

$$\tilde{M}\frac{\partial^2\tilde{\mathbf{u}}}{\partial t^2}+\tilde{D}\frac{\partial\,\tilde{\mathbf{u}}}{\partial t}+\tilde{K}\,\tilde{\mathbf{u}}=\tilde{\mathbf{f}} \tag{1.2}$$

où les 3 matrices $\tilde{M}$, $\tilde{D}$ et $\tilde{K}$ sont maintenant diagonales. On n'a donc plus ici qu'une liste d'équations différentielles *indépendantes*, chacune décrivant un mode vibratoire de l'objet. Chacun de ces modes a une fréquence donnée et la simplification consiste ici à ne garder que les plus basses fréquences, qui participent le plus au

rendu du mouvement. L'animation sera alors plus stable numériquement, les hautes fréquences nécessitant un faible pas de temps ayant été supprimées[1].

Une autre approximation consiste à remarquer que les modes vibratoires de basses fréquences d'un objet dépendent plus de ses dimensions que de sa forme précise. On précalcule donc des modes vibratoires pour un solide *parallépipédique*, et on les interpole aux dimensions de la boîte englobante de l'objet.

2.2 Déformation globale dynamique

Witkin et Welch ont présenté une technique d'animation et de contrôle globaux d'objets déformables [WW90]. Elle ajoute une composante temporelle aux déformations de l'espace de Barr [Bar84] vues plus haut, en faisant varier au cours du temps les coefficients de la matrice de déformation. Les énergies cinétiques et potentielles d'une configuration donnée peuvent être calculées et d'elles dérive une équation lagrangienne qui définit la dynamique du mouvement.

Cette méthode fut ensuite reprise dans [BW92] et par [MT92] dans un modèle à couches.

2.3 Conclusion

Associer des lois physiques aux déformations globales d'un objet le rend capable de mouvements autonomes. Ces méthodes, grâce aux approximations qu'elles mettent en œuvre, sont en outre très rapides à calculer. Mais c'est de ces mêmes approximations que provient leur faible adéquation avec nos objectifs, le mouvement simulé étant trop grossier pour être visuellement réaliste. Ces déformations globales conviennent en effet mal aux détails locaux recherchés dans un simulateur chirurgical. Parfois utilisées en synthèse d'images pour leur rapidité et leur aspect intuitif, nous pourrons les réserver à l'animation d'objets distants et secondaires.

À noter, dans une optique multirésolution, l'utilisation qui pourrait être faite des modes vibratoires du modèle de [PW89]. En fonction du temps disponible ou de la qualité de simulation désirée, on pourra en effet utiliser un plus ou moins grand nombre de modes vibratoires et ainsi régler aisément le compromis qualité - rapidité de calcul de l'animation, sans toutefois pouvoir l'adapter localement.

3 Les modèles masses-ressorts

De part leur simplicité et leur aspect intuitif, les modèles masses-ressorts ont eu un grand succès dans le domaine de l'image de synthèse. Une vision *continue* de l'espace étant difficilement compatible avec la modélisation informatique, on va le *discrétiser* pour ne plus le considérer que comme un ensemble de valeurs ponctuelles, se différenciant ainsi des méthodes *globales* de la section précédente.

3.1 Principe

Un objet va ainsi être représenté par un nombre fini de points, si possible équirépartis pour une meilleure représentation. Leur comportement les uns vis-à-vis des autres est intuitivement celui de petites masses ponctuelles reliées à leurs voisines par des relations de type ressorts (Fig. 1.3).

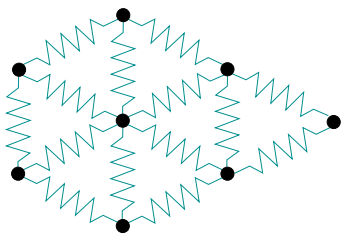


FIG. 1.3: Discrétisation d'un objet 2D à l'aide de masses-ressorts.

Avec des ressorts, il est facile de calculer la force subie par chaque masse à chaque instant en fonction des distances à ses voisines :

$$\mathbf{f}^i = \sum_{voisins\ j} k^{ij}\,(l^{ij} - l^{ij}_{initiale})$$

[1] Voir l'annexe B pour les problèmes liés à l'intégration.

où k^{ij} est la raideur du ressort entre le point i et son voisin j, l^{ij} étant sa longueur.

Les ressorts peuvent être plus complexes et comporter par exemple des termes d'amortissement. Dans tous les cas, l'équation fondamentale de la dynamique pour chaque masse sera de la forme (voir Eq. 1.1) :

$$m\,\frac{\partial^2 \mathbf{x}}{\partial t^2} + d\,\frac{\partial\,\mathbf{x}}{\partial t} + k\,\mathbf{x} = \mathbf{f}_{ext}$$

Comme vu précédemment, m, d et k sont trois scalaires désignant respectivement la masse, le coefficient de frottement et la raideur alors que $\mathbf{x}$ est lié à la position du point. $\mathbf{f}_{ext}$ désigne les forces externes appliquées à l'objet, comme la gravité. L'algorithme consistera donc à intégrer ces équations au cours du temps, et ce pour chaque point séparément[2].

De tels modèles ont été très tôt appliqués par Platt et Badler (sans coefficient d'amortissement) [PB81] à l'animation de visage, par Luciani [Luc85] pour l'animation de marionnettes ou Chadwick [CHP89] (en utilisant un réseau de ressorts pour simuler le maillage déformant d'une FFD) pour animer la chair de personnages.

3.2 Raffinements du modèle

Miller anima des reptiles avec cette méthode [Mil88] en ajoutant des contraintes de préservation de volume ainsi que la propagation d'ondes sinusoïdales à travers le corps, simulant le déplacement. À noter que l'affichage utilisait des surfaces splines et donnait un aspect très lisse cachant le réseau cubique sous-jacent.

En mélangeant des ressorts de raideurs différentes, Terzopoulos et Waters [TW90] animent un visage composé de plusieurs couches de ressorts représentant l'épiderme, le derme et les muscles, ces derniers étant *actifs* ce qui signifie que leurs paramètres étaient modifiés au cours du temps. Des contraintes additionnelles viennent ici aussi chercher à préserver le volume. La technique fut ensuite améliorée pour s'adapter à des visages réels acquis par imagerie médicale, en optimisant la position des masses en fonction des caractéristiques du visage [LTW95].

En plus des classiques ressorts longitudinaux, on peut ajouter des ressorts angulaires et ainsi obtenir des comportements nouveaux et plus complexes.

Joukhadar propose une méthode de modification du pas de temps d'intégration pour l'optimiser en évitant les divergences [Jou96]. Lorsqu'est détectée une trop grande variation de l'énergie du système, synonyme de problème d'intégration, le pas de temps est diminué.

On peut contrôler l'anisotropie du matériau en ne plaçant les ressorts que dans des directions privilégiées, ceux-ci agissant alors sur les tétraèdres qui composent l'objet [BC00]. Il existe en effet une anisotropie intrinsèque et non contrôlée dans les modèles masses-ressorts liée à la direction des ressorts.

Imposer une distance maximale d'élongation du ressort permet de limiter les trop grands étirements dans une simulation de tissus [Pro95]. Boux de Casson propose également de casser certaines des liaisons masses-ressorts en fonction des contraintes, simulant ainsi les déchirures d'un tissus [BdCL00].

Dans ce domaine, citons les récents travaux de Baraff et Witkin [BW98] qui peuvent grâce à une intégration implicite[3] simuler des ressorts de très forte raideur et obtenir de superbes animations de vêtements, utilisées dans la production audiovisuelle et disponibles dans le logiciel *Maya*.

3.3 Conclusion

Les méthodes masses-ressorts ont de très nombreux avantages. Simples à implémenter, intuitives et rapides, elles donnent de bons résultats facilement. Elles sont de plus facilement parallélisables et peuvent être temps-réel sur des machines d'entrée de gamme avec suffisamment peu de masses simulées.

Tous ces avantages devraient en faire un modèle de choix dans notre de cas, et de fait, de nombreux simulateurs chirurgicaux utilisent les réseaux masses-ressorts (voir Sec. 9). Il conviendra néanmoins de bien gérer les problèmes de stabilité numérique qui ne manquent pas de troubler le modèle pour de fortes raideurs. Un matériau organique demandera un pas de temps faible, limitant le nombre de masses simulables si l'on veut une simulation temps-réel. Nous avons pris le parti dans cette thèse d'explorer les intérêts d'une animation multirésolution et les masses-ressorts s'y prêtent par contre très mal.

[2] Voir l'annexe B.
[3] Voir l'annexe B.

S'ils permettent, en jouant sur les paramètres du système, d'obtenir un effet visuel donné à force d'essais successifs, il est impossible de prédire leur comportement dès lors que l'on modifie notablement la topologie ou la géométrie du maillage. Le *comportement dynamique* de maillages à différentes résolutions d'un même objet ne pourra donc être le même, et quand bien même on parviendrait à minimiser les différences en affinant les réglages, resterait le problème de la cohabitation des différentes résolutions. La simulation adaptative de tissus proposée par Hutchinson [HPH96] souffre ainsi de ce que le comportement du matériau évolue au cours de la simulation.

En pratique, même si l'on finit par avoir une intuition du comportement du système en fonction des paramètres qui le gouvernent, modifier la résolution du maillage pour l'affiner demande de nouvelles simulations pour obtenir l'effet désiré. Quelle raideur affecter alors à un ressort de telle longueur, dans telle configuration et avec tant de voisins ?

Ces problèmes d'*identification de paramètres* ont été abordés dans la littérature [DKT95, LPC95] en utilisant des méthodes, lentes, de recuit simulé ou des algorithmes génétiques. Mais les solutions ne cherchent à adapter les coefficients que sur des mouvements élémentaires, sans garantir le comportement de tout autre déplacement, ce qui ne saurait convenir.

4 Les systèmes de particules

4.1 Principe

Les systèmes de particules sont une extension du modèle masses-ressorts précédent. La différence réside dans le fait qu'on autorise les masses à interagir avec un ensemble de masses qui peut évoluer au cours de la simulation. Chaque masse interagit potentiellement avec toutes les autres, mais on seuille généralement à un rayon maximal sa zone d'influence pour limiter les calculs.

Terzopoulos *et al.* [TPF89] simulent un matériau en train de fondre en faisant varier la raideur des ressorts en fonction de la température (soumise à l'équation de la chaleur). Le ressort disparaît lorsque le point de fusion est atteint et l'on passe alors à un modèle de particules. Tonnesen [Ton91] utilise le terme de *particule thermique* et simule lui aussi liquides et solides. Plus souple que les masses-ressorts, les systèmes de particules n'imposent pas de restriction topologique, offrant la possibilité de simuler des fluides [MP89, LHVD95].

Les forces inter-particules sont souvent des lois d'attraction-répulsion. Similaires aux forces de *Lennard-Jones* [MP89], elles s'apparentent à celles qui existent entre deux atomes (voir Fig. 1.4) et présentent, comme les ressorts, une distance d'équilibre où la force est nulle. La force diminue par contre ici après une certaine distance pour s'annuler ensuite.

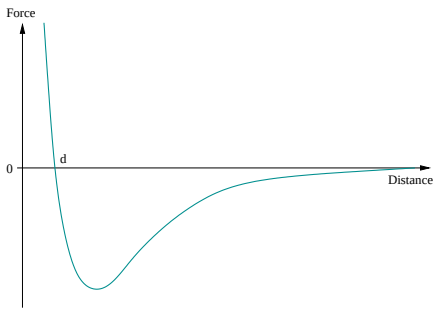


FIG. 1.4: Intensité de la force résultant d'un potentiel de Lennard-Jones en fonction de la distance.

On peut imaginer des fonctions plus complexes : les lois d'interaction peuvent être arbitrairement complexes et définies algorithmiquement [LC86], avoir plusieurs états d'équilibre [LHVD95, Rey97] ou par exemple dépendre de l'historique de la simulation. Les particules peuvent de plus avoir des "tailles" différentes [Jim93, LJR+91] ou être *orientées*, subissant alors des couples supplémentaires fonctions de l'orientation de leurs voisines [ST92, LP95].

L'intérêt des systèmes de particules est qu'ils sont faciles à mettre en œuvre et que des lois d'interaction simples qui les gouvernent peuvent naître des comportements complexes, dits *émergents*. Reeves a ainsi très tôt simulé des phénomènes tels que le feu et la fumée [Ree83] sans se préoccuper de la physique très complexe qui les régit. Reynolds a lui simulé le comportement global de bancs de poissons ou de vols d'oiseaux, chaque individu ne réagissant pourtant que selon des lois simples [Rey87].

L'affichage du modèle est simpliste avec des réseaux masses-ressorts à topologie fixe et dans lesquels les mailles surfaciques peuvent être polygonalisées. Il est par contre plus complexe dans le cas où l'on ne dispose que d'un ensemble de particules, représentant qui plus est un objet à topologie variable. Les surfaces implicites, en utilisant les particules comme squelettes générateurs de potentiel, sont alors bien adaptées, mais plus coûteuses. Elles peuvent aussi modéliser les déformations locales de l'objet autour de la zone de collision [CGD97].

4.2 Conclusion

Les systèmes de particules sont une extension des réseaux masses-ressorts. Ils permettent de simuler une plus grande variété de comportements, comme celui des gaz ou des liquides. L'algorithme reste simple et l'on peut simplement modifier la topologie de l'objet ce qui autorise les découpes. Il conviendra par contre de bien gérer la complexité algorithmique qui peut apparaître avec beaucoup de particules lorsque l'on doit chercher celles qui interagissent entre elles. On utilisera pour cela un partitionnement de l'espace qui ramènera le coût quadratique à un coût quasi-linéaire en le nombre de masses.

Le problème avec ces méthodes est, tout comme avec les masses-ressorts, qu'on ne peut généralement pas garantir un comportement indépendant de la résolution. L'exception est le modèle SPH qui le permet grâce à l'utilisation de forces inter-particules bien choisies.

5 Le modèle SPH

Le modèle SPH, pour *Smoothed Particles Hydrodynamics*, est issu de l'astrophysique et a l'avantage sur les précédents de proposer une véritable *équation d'état* du matériau, conduisant à un comportement indépendant de la résolution [Mon92]. C'est une méthode à base de particules comme vu précédemment, mais elles sont maintenant vraiment considérées comme les *points d'échantillonnage* d'un petit volume les entourant et qu'elles représentent. C'est une méthode Lagrangienne, désignée comme méthode Lagrangienne *libre libre* par Rémi Cozot dans sa thèse [Coz96].

Simplifiée, elle a été employée dans le cadre de l'animation de phénomènes gazeux [SF93, SF95] et fut à la base d'un modèle décrit dans la thèse de Mathieu Desbrun [Des97], dont nous reprenons ici quelques explications et à laquelle on se référera pour des explications plus détaillées.

5.1 Passage du continu au discret

Un champ continu f va être approximé en un point $\mathbf{x}$ par une valeur portée par particule et calculée comme une moyenne locale des valeurs du champ sur un petit volume V autour de la particule. Cette valeur moyenne locale, notée $< f(\mathbf{x}) >$, peut être écrite comme la convolution de f par un noyau de filtrage :

$$< f(\mathbf{x}) > = \int_V f(\mathbf{x'})\, W_h(\mathbf{x}-\mathbf{x'})\, d\mathbf{x'} \tag{1.3}$$

où W_h est un filtre de lissage dont le paramètre h contrôle l'étendue du filtrage. L'intégrale de W_h doit être égale à 1 quel que soit h, et W_h doit tendre vers l'impulsion de Dirac quand h tend vers zéro.

On passe d'une intégrale sur le *volume* à une intégrale sur la *masse* en utilisant la densité de masse ρ :

$$\begin{aligned} < f(\mathbf{x}) > &= \int_V \frac{f(\mathbf{x'})}{\rho(\mathbf{x'})}\, W_h(\mathbf{x}-\mathbf{x'})\, \rho(\mathbf{x'}) d\mathbf{x'} \\ &= \int_M \frac{f(\mathbf{x'})}{\rho(\mathbf{x'})}\, W_h(\mathbf{x}-\mathbf{x'})\, dm \end{aligned} \tag{1.4}$$

Une approximation discrète de l'équation précédente, obtenue par une méthode de Monte-Carlo grâce aux valeurs du champ pour les autres particules donne ($f_j = f(\mathbf{x}_j)$ et m_j est la masse de la particule j) :

$$< f(\mathbf{x}) > \simeq \sum_j m_j \frac{f_j}{\rho_j} W_h(\mathbf{x}-\mathbf{x}_j) \tag{1.5}$$

En intégrant par parties, et en remarquant que W_h s'annule au bord du volume, on peut calculer le *gradient* du champ :

$$\int_V (\mathbf{grad}\ f)(\mathbf{x'})\ W_h(\mathbf{x}-\mathbf{x'})\ d\mathbf{x'} \ = \ \int_V f(\mathbf{x'})\ \ (\mathbf{grad}\ W_h)(\mathbf{x}-\mathbf{x'})\ d\mathbf{x'} \tag{1.6}$$

Ce qui se traduit en version discrétisée par :

$$< (\mathbf{grad}\ f)(\mathbf{x}) > \simeq \sum_j m_j \frac{f_j}{\rho_j} (\mathbf{grad}\ W_h)(\mathbf{x}-\mathbf{x}_j) \tag{1.7}$$

Un ensemble, même désordonné, de points d'échantillonnage permet ainsi de calculer les valeurs de champs continus et de leurs dérivées de façon simple par filtrage local. Ceci va permettre de simuler les équations d'état qui gouvernent le comportement d'un matériau.

5.2 L'équation d'état du matériau

Les particules composant le matériau vont être soumises à un *champ de pression* qui va déterminer leurs déplacements. Les forces générées sont proportionnelles au gradient de la pression [Bat73], qui va pouvoir être calculé avec l'Équation 1.7 précédente.

Dans [Des97], l'expression du champ de pression P, qui constitue l'équation d'état a été choisie comme étant :

$$P \ = \ k(\rho - \rho_0)$$

Les forces générées chercheront ainsi à rétablir la densité ρ_0 d'équilibre en créant une pression là où ρ est différent de ρ_0, pression réglable au travers du paramètre k.

Restent à choisir la forme du filtre de lissage W_h employé (qui modifiera l'animation), à ajouter des forces de dissipation visqueuses (pour stabiliser) et à faire en sorte que les forces créées satisfassent le principe d'action-réaction (voir [Des97]). On peut alors envisager la simulation à différentes échelles du matériau.

5.3 Simulation multirésolution des SPH

Ajouter ou supprimer des points d'échantillonnage est maintenant possible, tout en continuant d'approcher un comportement donné, qui est celui dicté par l'équation d'état. En fonction de seuils de continuité, Desbrun propose de diviser ou regrouper plusieurs particules lors de la simulation, optimisant ainsi l'utilisation du temps de calcul (Fig 1.5) [Des97, DC99b].

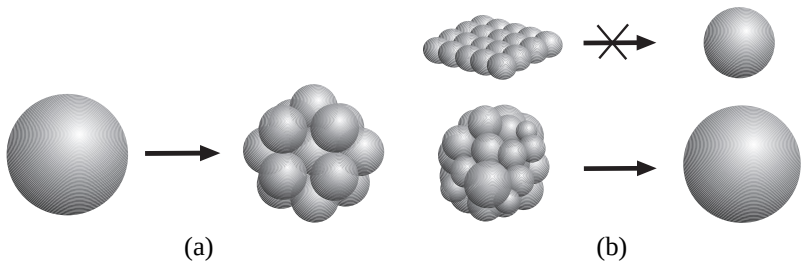


FIG. 1.5: Chaque particule peut se diviser en plusieurs autres (a). Le regroupement est possible lorsque la forme est globalement sphérique (b).

Le pas de temps d'intégration peut également être modifié au cours de la simulation en fonction de critères physiques liés à la vitesse de propagation d'une onde dans le matériau [4].

Hernquist et Katz introduisent la notion de *Tree SPH*, ou SPH hiérarchique [HK89]. Le partitionnement de l'espace en un arbre hiérarchique permet en effet, comme pour tout système de particules, d'accélérer les calculs de voisinage et d'influence. Cette découpe de l'espace n'est pas liée à une grille Eulérienne mais est Lagrangienne tout comme les SPH, et évolue avec la simulation. L'application est la simulation de très gros systèmes auto-gravitationnels en astrophysique.

[4] Voir l'annexe B.

5.4 Conclusion

Ce modèle possède beaucoup des avantages intrinsèques des modèles à base de particules décrits précédemment. À chaque instant, on est capable de calculer en un temps équivalent à celui d'un système de particules la force subie par chaque point pour ensuite l'intégrer au cours du temps. La différence est que le calcul plus subtil de la force en utilisant le formalisme SPH garantit le respect d'une équation d'état, autorisant un processus multirésolution. Dilts a montré l'équivalence des SPH avec la méthode d'éléments finis de Galerkin appliquée en temps et en espace [Dil96].

On pourra toutefois reprocher aux SPH d'être mieux adaptées à la simulation d'un comportement gazeux qu'à celle d'un véritable matériau élastique. Augmenter la cohésion interne du matériau peut se faire en augmentant le rayon d'influence h du noyau W_h et le nombre de particules, au prix de calculs plus coûteux. Desbrun annonce ainsi des temps de calculs de plusieurs heures lors de la simulation de matériaux relativement peu rigides [DC99b].

6 Les modèles continus

Tout comme le formalisme SPH, les modèles décrits ici et dans la section suivante considèrent avant tout la matière comme un *milieu continu*, soumis à des *lois de comportement* globales. L'objet simulé ne sera finalement discrétisé que pour les besoins de la simulation informatique.

On va chercher à mesurer l'énergie potentielle de l'objet sous l'action de contraintes extérieures données. Cette énergie peut être calculée en fonction d'une variable qui décrit les déformations de l'objet. Elle correspond au *travail*[5] qu'il a fallu fournir pour amener l'objet dans cette position déformée.

Le travail étant l'intégrale du produit force $\times$ déplacement, il est nécessaire de connaître les forces créés par la déformation pour le calculer et c'est une *loi de comportement* qui va donner l'expression de la force en fonction de la déformation, en faisant intervenir des coefficients propres au matériau simulé. Tout ce formalisme sera décrit plus en détails dans le chapitre suivant et il suffit ici de retenir que l'on peut calculer l'énergie potentielle en fonction des déformations.

6.1 Minimisation de l'énergie de déformation

Cette énergie potentielle de déformation a été directement utilisée dans une série d'approches. Terzopoulos *et al.* ont dès 1987 proposé un formalisme dans lequel une courbe, une surface ou un objet 3D se déforme afin de minimiser leur énergie de déformation [TPBF87]. Celle-ci est mesurée comme étant la norme d'une matrice, appelée *tenseur métrique* et qui est en fait pratiquement le tenseur des déformations de *Green-Lagrange* (dont on reparlera dans le prochain chapitre).

Les différentielles impliquées dans le calcul de l'énergie potentielle sont approximées par des différences finies. L'équation différentielle dynamique est intégrée dans le temps en utilisant une intégration *implicite* [6]. Des termes annexes, du même type, doivent être introduits dans l'évaluation de l'énergie potentielle pour des simulations d'objets 2D et 1D (voir [TPBF87, WFB87, PB88]). La méthode reste très lente car nécessitant une inversion matricielle à chaque pas de temps.

Le même type d'approche a été utilisé dans la méthode des *snakes*, ou *contours actifs* de [KWT87] dans un contexte d'analyse d'image et de recherche de contour. L'énergie de déformation mesure alors la courbure du contour, qui cherche à entourer les zones d'intérêt de l'image ou du volume.

L'approche précédente fut améliorée l'année suivante en définissant l'objet comme la somme de deux composantes, l'une rigide, dite *de référence*, et l'autre déformable attachée à la première [TW88]. Ce modèle hybride permet de simplifier l'expression de l'énergie potentielle qui n'a plus à gérer les transformations rigides (translation et rotation globales). Dans l'hypothèse des petites déformations, on peut même considérer cette matrice comme constante et l'inverser une fois pour toutes au début. Des forces de pénalité générées par les obstacles viennent ensuite déformer l'objet.

Dans [TF88], la composante de référence n'est plus rigide et peut "absorber" une partie des déformations, conduisant à des objets *plastiques*. Cette composante peut également subir des forces de frottement visqueuses

[5] Au sens physique, propre et mathématique du terme.
[6] Voir annexe B.

et les auteurs décrivent la possibilité de fractures en des points préétablis du maillage en cas de trop forte déformation.

6.2 Conclusion

Ces méthodes, assez anciennes, ont permis les premières animations de matériaux déformables. Leur inconvénient majeur est leur coût de calcul élevé dû à l'inversion matricielle qu'ils requièrent, ce qui les rend incompatibles avec une application temps-réel.

L'utilisation des différences finies dans [TPBF87] limite le champ d'utilisation à des maillages réguliers, les objets ayant forcément une forme liée à la grille utilisée. La méthode des éléments finis que nous décrivons dans la section suivante est en fait une généralisation de ces méthodes, basées sur une vision continue de la matière. On va maintenant être capable d'évaluer les équations différentielles sur un maillage arbitraire composé d'éléments.

7 Les éléments finis

Les éléments finis sont *la* méthode de prédilection des physiciens pour simuler un matériau déformable. Indispensables dans l'industrie et faisant l'objet de publications en mécanique et en mathématiques appliquées depuis des dizaines d'années, ils utilisent également le modèle continu décrit dans la section précédente.

La littérature sur le sujet est souvent trop pointue alors que le résultat visuel recherché en animation ne demande pas un modèle si précis. Elle est de ce fait d'une compréhension difficile, la méthode générale pouvant être expliquée plus simplement.

Cette technique peut également sembler être réservée à l'étude des positions d'équilibre *statiques* et non à l'évolution *dynamique* des objets au cours du temps, les calculs impliqués étant qui plus est souvent très longs. Toutes ces raisons expliquent le relativement faible engouement pour la méthode dans le domaine de l'image de synthèse, alors qu'elle a d'évidents avantages, comme nous allons essayer de le montrer dans cette section.

Cette description est très simplifiée et l'on se reportera à [Seg84, Bat96, Kas97] ou [ZT91] pour plus de détails.

7.1 Principe

On part ici aussi de l'expression de l'énergie potentielle de déformation de l'objet. La position d'équilibre recherchée sera celle qui *minimise* cette énergie, et donc où la dérivée de l'énergie par rapport à la position s'annule. Le but est donc de trouver la valeur de la déformation qui satisfait cette équation différentielle d'équilibre.

Chercher la valeur de la déformation comme une fonction *continue* dans le matériau signifie résoudre analytiquement cette équation, ce qu'on ne sait faire que dans des cas très simples. C'est donc ici qu'intervient la discrétisation en éléments finis.

Discrétisation en éléments

La fonction continue recherchée va être représentée par une approximation discrète. On va découper l'objet en *éléments*, sortes de cellules de matière possédant chacune un faible nombre de variables qui décriront l'allure de la fonction à l'intérieur de l'élément. Ces variables seront en fait assignées à chacun des *sommets* de l'élément, chacun décrivant plus précisément par ses variables ce qui se passe autour de lui dans l'élément. La fonction étant entièrement décrite par ces quelques coefficients, on restreint grandement l'espace des fonctions que l'on peut représenter.

Des raisons de continuité sur la fonction recherchée vont faire que l'on va souvent imposer à deux éléments juxtaposés d'avoir les mêmes variables aux sommets qu'ils partagent. Cela assure que l'on n'aura pas de discontinuité de la fonction quand on l'exprimera d'un côté et de l'autre de la frontière entre les éléments. La fonction est donc maintenant représentée par un ensemble de variables, définies en chaque sommet des éléments créés.

La description de la fonction par ces variables se fait au travers des *fonctions de base*. Elles décrivent pour chaque sommet comment les variables qu'il contient influencent la fonction générée. Ces fonctions de base seront généralement de la même forme pour tous les sommets et de leur complexité dépendra l'espace des fonctions représentables et donc la qualité de la solution.

Soyons concrets. La fonction cherchée sera le plus souvent approximée par une *fonction linéaire par morceaux*, linéaire sur chacun des éléments (plus rarement par un polynôme de plus haut degré). Pour la définir il suffit d'*une* valeur à chaque sommet[7] et les fonctions de base sont alors simplement les fonctions linéaires par morceaux qui valent 1 au sommet considéré et 0 sur tous les autres sommets (des fonctions triangle en 1D). Toute fonction linéaire par morceaux sur les éléments peut en effet être représentée comme une somme pondérée de ces fonctions de base, les poids associés à chaque sommet étant les valeurs prises par la fonction en ces sommets. Ce seront les variables qui décrivent la fonction.

La Figure 1.6 explicite cette représentation en une dimension. Chaque élément est un segment de l'axe des abscisses, sur lequel les fonctions de base ont une forme triangulaire (Fig. 1.6a). La fonction continue est approximée par une fonction linéaire par morceaux, somme pondérée de ces fonctions triangles (Fig. 1.6b).

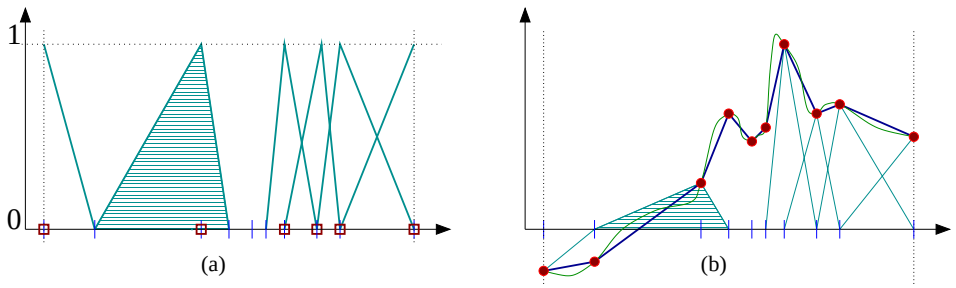


FIG. 1.6: Les fonctions de base ont une forme triangulaire en 1D (seules celles associées aux sommets encadrés sont représentées) (a). Leur somme pondérée crée la fonction linéaire par morceaux qui approxime la fonction continue (b).

Noter que la forme des éléments (leur longueur en 1 dimension) n'a pas à être uniforme et qu'il est intéressant de l'adapter à la courbe approximée comme c'est le cas ici. Cela demande de connaître *a priori* l'allure de la solution cherchée.

En deux dimensions, sur un maillage d'éléments triangulaire, les fonctions de base auront la forme de pyramides s'appuyant sur les frontières du maillage.

En résumé, on a par passage aux éléments finis représenté la fonction cherchée (les déformations de l'objet) par un ensemble de variables aux sommets du maillage.

Résolution du système

L'équation d'équilibre du système doit être vérifiée sur chacun des éléments, ce qui va se traduire par des équations sur chacun des sommets. Elle comporte des dérivées partielles qui s'expriment analytiquement en chaque sommet comme une combinaison linéaire des valeurs prises par la fonction sur les autres sommets.

En chaque sommet, on peut donc écrire une équation linéaire faisant intervenir la valeur de la fonction dans cette zone (définie par la valeur en ce sommet et aux sommets proches). En pratique, chaque sommet satisfait une, deux ou trois équations selon que l'on est en 1, 2 ou 3 dimensions, car chaque sommet comporte alors les 1, 2 ou 3 composantes de la fonction déformation, qui chacune doit avoir une dérivée nulle par rapport à la position du sommet.

En regroupant toutes ces équations linéaires dans une matrice on obtient un système matriciel de n (resp. $2n$, $3n$) équations en les n (resp. $2n$, $3n$) inconnues que sont les valeurs de la fonction aux sommets du maillage :

$$K\,\mathbf{U} = \mathbf{F} \tag{1.8}$$

où K, appelée *matrice de rigidité* du système, et où $\mathbf{U}$ et $\mathbf{F}$ sont les vecteurs rassemblant respectivement les déplacements et les forces externes appliquées aux n sommets.

Il ne reste qu'à résoudre ce système pour obtenir l'approximation de la fonction déformation sur le maillage, dont on pourra ensuite par exemple déduire les contraintes internes grâce à la loi de comportement.

[7] Si l'on est en dimension 1. En dimension 3 la fonction cherchée a 3 composantes et sera donc représentée par 3 fonctions linéaires $\mathbf{U}_x$, $\mathbf{U}_y$ et $\mathbf{U}_z$. On aura alors 3 inconnues en chaque sommet.

En pratique, puisque seules les valeurs prises sur les sommets les plus proches interviennent dans l'expression des dérivées[8], chaque équation n'impliquera qu'un petit nombre de sommets. La matrice K née de leur regroupement aura donc une faible largeur de bande, des méthodes adaptées pouvant être utilisées pour la résolution. Noter aussi que le conditionnement de la matrice[9] dépend directement des coefficients de raideur du matériau : plus il est rigide, plus l'inverse est difficile à calculer.

En chaque sommet la déformation et la force appliquée (généralement nulle) peuvent être connues ou être des variables du système. Rien n'empêche d'imposer une déformation à un sommet (s'il est en contact avec un autre objet par exemple) et d'y calculer la force créée, tout comme on peut lui faire subir une force donnée (nulle à la surface du matériau sauf en quelques points par exemple) et chercher la déformation résultante. On peut donc échanger les inconnues et les données du problème dans les valeurs de $\mathbf{U}$ et $\mathbf{F}$, et modifier alors le système matriciel K pour ne plus avoir que des inconnues d'un côté et des données de l'autre.

Simulation dynamique

En animation, on ne cherche généralement pas à trouver les positions d'équilibre d'un objet mais plutôt ses déformations au cours du temps. Il faut alors prendre en compte les inerties et les forces de frottement, ce qui transforme l'Équation 1.8 en une équation différentielle du second ordre classique :

$$M\frac{\partial^2 \mathbf{U}}{\partial t^2} + D\frac{\partial \mathbf{U}}{\partial t} + K\mathbf{U} = \mathbf{F} \tag{1.9}$$

où M et D sont des matrices liées à la masse et aux coefficients d'amortissement. Elles sont obtenues, tout comme K, en regroupant les équations de chaque sommet. La masse d'un sommet est ainsi calculée comme l'intégrale de la fonction densité au travers des fonctions de base (de même pour D obtenue à partir des coefficients de frottement de chaque élément).

Les différentes matrices M, D et K peuvent dépendre de la géométrie des éléments et doivent donc être recalculées dès que l'objet se déforme, ce qui est coûteux. Cozot désigne par le terme de lagrangien *libre* de telles méthodes dans lesquelles un remaillage de l'objet suit chaque pas de simulation fait par éléments finis [Coz96].

On considère généralement que l'objet se déforme suffisamment peu pour considérer ces matrices comme constantes. Cette approximation signifie que l'on approxime un comportement purement linéaire de l'objet, ce qui n'est généralement vrai que pour les petites déformations. Il faudrait au delà d'une certaine déformation modifier les coefficients pour obtenir un comportement plus réaliste. Considérer ces valeurs comme constantes n'est néanmoins pas absurde, l'interaction entre deux points proches étant conditionnée par leur disposition géométrique (le nombre d'atomes les séparant), qui ne changera pas au cours du temps.

7.2 Applications

Méthode classique

L'une des premières application des éléments finis à l'animation fut faite par Gourret *et al.* dans une simulation de contact entre une main et une balle déformables. La réponse aux collisions est quelque peu empirique et la modélisation des doigts simple, mais la simulation permet une déformation complexe [GMTT89]. Les matrices sont remises-à-jour quand les articulations des doigts ont bougé de plus de dix degrés. La simulation est longue à calculer et la dynamique en a été supprimée pour ne pas la ralentir davantage.

Chronologiquement vinrent ensuite Collier *et al.* avec un modeleur de tissus utilisant des éléments carrés en deux dimensions, inspirés de ceux utilisés pour simuler les membranes dans l'industrie [CCOS91].

Chen et Zeltzer cherchèrent quand à eux à animer les muscles humains [CZ92]. Leurs éléments sont très complexes (20 nœuds chacun) et il n'en utilisent que 2 par muscle. Ceux-ci se voient imposer des forces par l'intermédiaire des tendons qui les relient aux os. Bien qu'utilisant les principes de la dynamique modale décrits dans la Section 2.1, leur simulation de quelques dizaines de noeuds est très lente.

Bro-Nielsen et Cotin utilisèrent une idée intéressante en 1996 en remarquant tout simplement que seul le déplacement des nœuds de la surface est pertinent visuellement [BNC96]. Le système matriciel de l'Équation

[8] Les sommets ont une influence sur la fonction limitée au support de leur fonction de base. Avec des fonctions linéaires, l'expression des dérivées en un sommet ne fera intervenir que les sommets voisins qui lui sont directement liés par un élément.

[9] Sa facilité à être inversée.

1.8 peut être réécrit en séparant ces nœuds de ceux de l'intérieur. Cette méthode, dénommée *condensation* par les auteurs, aboutit à un système matriciel qui ne concerne plus que les nœuds de la surface, donc de taille bien moindre, mais qui a en contrepartie perdu sa structure de matrice creuse. À noter que cette manipulation, classique chez les physiciens, avait déjà été appliquée dans [GMTT89].

Le nouveau système, de type $K\tilde{\mathbf{U}} = \tilde{\mathbf{F}}$ est alors pré-inversé, afin de pouvoir déterminer rapidement les déformations en fonction des forces par

$$\tilde{\mathbf{U}} = K^{-1}\tilde{\mathbf{F}}$$

On déformera l'objet en appliquant des forces non nulles à quelques endroits (seules quelques colonnes de K^{-1} seront donc à prendre en compte, ce qui accélère beaucoup le produit), et en en déduisant très rapidement les déformations générées. On obtient ainsi une position d'équilibre *statique* de l'objet. L'inconvénient de piloter les déformations en appliquant des forces et non des déplacements aux nœuds est qu'on ne peut alors garantir la forme des déformations à l'endroit où l'on voudrait appliquer un outil. Celui-ci peut ne pas repousser assez l'objet (si les forces appliquées sont trop faibles) ou au contraire sembler agir à distance (forces trop fortes), ce qui pose évidement problème. On doit donc ici se mettre *à la place* du modèle physique et appliquer exactement les forces que celui-ci aurait générées pour avoir une simulation correcte, ce qui est impossible. On gagne par contre énormément en temps de calcul en imposant ainsi des forces arbitraires car la matrice peut-être pré-inversée une fois pour toutes.

Dans les applications classiques, l'outil va imposer une déformation à l'objet, les forces générées à ces nœuds devenant alors les inconnues du système. Si les points sur lesquels on agit ne sont pas toujours les mêmes au cours de la simulation, ce qui est le cas, le système matriciel change tout le temps et ne peut donc être ainsi pré-inversé.

On peut par contre effectuer une simulation dynamique, l'équation $M\frac{\partial^2\mathbf{U}}{\partial t^2} + D\frac{\partial\mathbf{U}}{\partial t} + K\mathbf{U} = \mathbf{F}$ pouvant se réécrire à l'aide de différences finies pour calculer les dérivées temporelles. On obtient alors un système de la forme

$$K'\tilde{\mathbf{u}}_t = \tilde{\mathbf{f}}_{(t,\,t-1,\,t-2)}$$

où le vecteur $\tilde{\mathbf{f}}$ dépend maintenant des déformations aux deux instants précédents. Il est plus cher à calculer et n'a plus comme précédemment quasiment toutes ses valeurs nulles ce qui ralentit beaucoup le produit matriciel avec $(K')^{-1}$.

Dans le même ordre d'idée, James et Pai [JP99] ont tout récemment proposé une méthode basée sur le même principe, mais dans laquelle ils *mettaient à jour* l'inverse de la matrice K. Lorsque l'utilisateur manipule un outil virtuel pour déformer l'objet, peu de nœuds passent du statut de déplacé à celui de libre, et inversement. Les auteurs ont alors fait remarquer que la matrice inverse n'était en fait que peu modifiée dans ces cas. L'influence d'un point étant limitée à une colonne de la matrice, une astuce mathématique permet de la remettre à jour rapidement à l'aide de lourds précalculs, l'animation restant temps-réel.

Dans sa thèse [Cot97], Cotin a proposé une méthode imposant cette fois ci une contrainte en *déplacement* sur l'objet. Il utilise l'intéressante propriété de *superposition* des éléments finis *en élasticité linéaire*, qui dit que la déformation de l'objet résultant du déplacement **d** d'*un* point est la somme de celles qui seraient produites par des déplacements d_x, d_y et d_z appliqués séparément dans chacune des 3 directions x, y et z. Ce résultat est dû au caractère linéaire du système d'équations. Qui plus est, un déplacement double va simplement créer une déformation double[10].

Il suffit alors de précalculer, pour chaque point séparément, quelle va être la déformation de tout l'objet lorsque ce point est déplacé dans les trois directions. Ces informations, filtrées pour supprimer les déplacements trop faibles et économiser ainsi la mémoire, sont stockées dans chaque point *de la surface*, conduisant à une animation temps-réel [CDA99, CDA96, CDA98].

Le problème est qu'on ne peut connaître la déformation créée lorsqu'on manipule plus d'un point à la fois (ce qui est généralement le cas). Appliquer en les sommant chacune des déformations créées par les points déplacés va conduire à une déformation exagérée. Cotin n'en applique alors qu'une partie, en représentant dans une matrice l'influence mutuelle des nœuds déplacés, dont l'inverse va déterminer les déplacements à appliquer aux différents nœuds. On peut ainsi déplacer plusieurs sommets, mais au prix de l'inverse d'une matrice $3m \times 3m$ où m est le nombre de noeuds déplacés.

[10] Ce comportement n'est pas très réaliste, mais la linéarité est une première approximation de la complexité des mouvements réels.

L'un des défauts de ces trois dernières méthodes ([BNC96, JP99, Cot97]) est que toute modification de la topologie de l'objet (une simple entaille ou une découpe plus profonde) est impossible car elle remettrait en cause toutes les matrices précalculées qui ne seraient plus adaptées. Mais leur inconvénient majeur est qu'elles sont *statiques*, ne donnant qu'une suite d'états d'équilibre de l'objet. C'est en particulier sensible lorsque l'outil quitte brusquement l'objet et que celui-ci revient alors immédiatement dans sa position d'équilibre sans aucune oscillation dynamique.

Éléments finis explicites

Toutes les méthodes décrites jusqu'à maintenant font usage d'éléments finis *implicites* et nécessitent l'inversion d'un système global. Les éléments finis *explicites*, au contraire s'apparentent plus à des systèmes de particules et profitent de leur flexibilité.

Le principe est le suivant : lorsque, sur chacun des éléments du système, on a écrit les équations d'équilibre, on est capable d'exprimer la force que subit chaque sommet en fonction des déplacements des sommets voisins. Plutôt que de chercher à trouver la position d'équilibre que ces forces vont créer en résolvant le système matriciel, on va simplement intégrer ces forces pour chaque point. On obtient ainsi de nouvelles positions dans lesquelles on va de nouveau calculer des forces, etc. Il suffit d'utiliser les lois de la dynamique du point pour créer l'animation, ce qui est plus simple que d'essayer de résoudre l'équation du second ordre 1.9.

On perd ici en précision ce que l'on va gagner en vitesse. La résolution globale du système garantissait que les positions trouvées étaient toutes compatibles entre elles, satisfaisant globalement toutes les équations. On n'a plus ici cette cohérence, mais plutôt une juxtaposition de solutions locales, bien plus rapides à calculer. On introduit par contre un retard dans la propagation, chaque point ne réagissant qu'à la position de ses voisins à l'instant *précédent*.

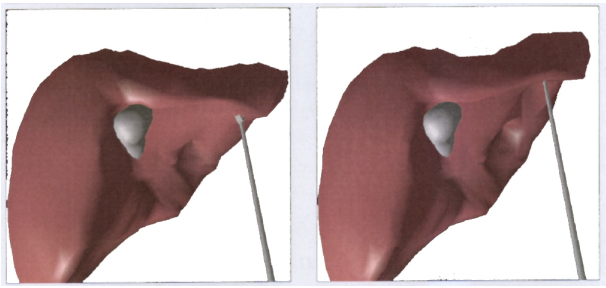


FIG. 1.7: Simulation de laparoscopie hépatique utilisant les masses-tenseurs.

Cette méthode a été utilisée dans deux contributions récentes de la littérature. De part sa ressemblance avec les masses-ressorts (on calcule une force en chaque point d'après ses voisins puis on l'intègre), Cotin a dans sa thèse dénommé cette méthode les *masses-tenseurs*, le calcul de la force pouvant s'exprimer par un tenseur [11] (voir Fig 1.7) [Cot97, DCA99].

Plus récemment, O'Brien et Hodgins ont proposé une méthode simulant les fractures et leur propagation à l'intérieur d'un matériau en utilisant également eux aussi une méthode explicite [OH99]. La discrétisation du modèle en éléments finis évoluait au cours du temps pour suivre l'apparition des craquelures dans le matériau. Celles-ci apparaissaient à l'endroit où les contraintes internes dépassaient un seuil donné.

Ces deux approches sont bien plus rapides que celles nécessitant une inversion globale, même si dans le cas de [OH99] les très fortes raideurs des matériaux employés demandaient un très faible pas de temps, nécessaire également au suivi de l'évolution des fractures. Cotin propose ainsi un simulateur chirurgical interactif.

7.3 Conclusion

Une fois le formalisme compris (on n'a en particulier pas décrit ici la façon dont sont calculées les dérivées partielles sur les éléments), utiliser les éléments finis revient dans la méthode classique à résoudre un gros système matriciel. Trop lente, cette opération interdit les applications temps-réel.

Pré-inverser les matrices en les supposant constantes au cours du temps limite les applications à de faibles déformations. Cela demande de fixer une fois pour toutes le statut (déplacé par l'outil ou libre) de chacun des

[11] On peut voir un tenseur comme une matrice symétrique.

points. Il faut employer des mathématiques subtiles pour pouvoir comme dans [JP99] mettre à jour cette inverse en fonction du statut de chaque point et pouvoir ainsi déplacer des points arbitraires avec un outil. Encore dans ce cas là n'obtient-on que des états d'*équilibre* du système, la dynamique de l'animation faisant cruellement défaut.

Les éléments finis explicites sont quand à eux bien plus rapides, et devraient permettre le temps-réel. Ils procurent de meilleures conditions que les masses-ressorts, le temps de calcul de la force en chaque point pouvant en effet être du même ordre de grandeur alors que moins de points simulés suffisent à obtenir le même résultat visuel.

Les changements de topologie sont impossibles avec les éléments finis implicites, mais mis en œuvre par Cotin avec une méthode explicite. Le système matriciel doit en effet alors être complètement recalculé, les interactions entre points étant modifiées. Ce formalisme n'est en outre pas approprié à la simulation de matériaux trop déformables. Les modèles physiques décrivant les matériaux sont faits pour des métaux et tolèrent une déformation de l'ordre de 1%, incompatible avec les tissus biologiques.

On peut considérer les matrices comme constantes pour de très faibles déformations, mais il faudrait pour des matériaux biologiques les remettre à jour régulièrement. Les résultats de Cotin avec une méthode explicite semblent néanmoins montrer que l'on doit pouvoir dépasser ces limites, aux dépends du réalisme physique de la simulation.

8 Modèles à couches

8.1 Principe

Les modèles que nous allons maintenant décrire vont essayer de mélanger plusieurs des approches présentées pour s'en enrichir en tirant parti de leurs avantages respectifs. Tout comme les modèles de Terzopoulos vu précédemment [TW88, TPF89], c'est généralement des *couches* constituées de différents modèles qui vont être superposées.

Dès 1976, Burtnik [BW76] animait le squelette d'un personnage et en déduisait le mouvement de la couche de muscles et de peau qui y était attaché, le travail de l'animateur s'en trouvant simplifié. La méthode fut améliorée par Chadwick *et al.* [CP88, CHP89] par la séparation des muscles, animés par une FFD contrôlée par un réseau masses-ressorts, et de la peau purement géométrique. Ce type de modèle est repris par Cani-Gascuel *et al.* dans [GVP90, GVP91, Gas90], les interactions entre les couches étant plus complexes. [CGD97] reprendra certaines des idées du modèle de [GVP91] en remplaçant la surface spline par une surface implicite.

Le modèle de Turner [TT93, Tur95] d'animation de personnages mélange une animation cinématique du squelette par cinématique inverse, des muscles modélisés par objets implicites attachés au squelette par des ressorts, et enfin une peau sur laquelle se concentrent les calculs et qui utilise la méthode de [TPBF87]. Shen et Thalmann [ST95] se concentreront sur la modélisation des muscles par *metaballs*, la peau purement géométrique étant constituée de B-splines.

Citons également en 1992 les travaux de Metaxas et Terzopoulos [MT92] qui reprennent ceux de [TF88] et [WW90] dans des simulations utilisant plusieurs couches.

8.2 Conclusion

Ces modèles sont souvent rapides, car ils décomposent le problème et simulent avec une méthode adaptée chaque partie. Ils offrent ainsi un paramétrage séparé de chaque couche, ce qui peut être intéressant car parfois plus intuitif.

Ce genre d'approches se rapproche de la multirésolution. Celle-ci ne consiste néanmoins pas simplement en un empilement de couches. On doit plutôt la voir comme une façon de générer automatiquement des couches pouvant interagir et se modifier. Or les méthodes proposées offrent très rarement une interaction réelle entre les couches, chacune se contentant plutôt de contrôler seulement la suivante. Dans [GVP91, CGD97] ou [TW88] l'interaction se fait dans les deux sens, les déformations de la peau pouvant à leur tour influencer le modèle interne.

Les sections précédentes classaient les modèles par grandes familles de méthode. Nous allons maintenant dépasser ce classement et nous intéresser à deux types d'algorithmes, plus spécifiquement visés par cette thèse : ceux autorisant une simulation interactive et ceux tirant parti de la multirésolution.

9 Applications interactives

Cette section détaille quelles sont, parmi les méthodes présentées, celles qui permettent une véritable animation interactive. On ne parle pas ici de *temps-réel* car nous réserverons ce terme à des applications bien précises (voir Sec. 11.2). Les applications présentées permettent néanmoins une réaction interactive du matériau aux sollicitations de l'utilisateur, accompagnées d'un affichage de plusieurs images par seconde.

L'application typique des modèles déformables temps-réel est le simulateur médical. Très utile pour la formation des chirurgiens, il est actuellement sujet de recherche pour de grands groupes industriels :

MUSE	`www.musetech.com/html/muse2000/apps/medical.html`
HT Medical System	`www.ht.com`
Teneo	`www.teneocomp.com`

et universitaires :

MIT	`touchlab.mit.edu`
Penn State	`cs.millersv.edu/haptics`
EPFL	`imt.dmt.epfl.ch`
Karlsruhe	`www-kismet.iai.fzk.de/TRAINER/mic_trainer1.html`
LIFL	`www.lifl.fr/GRAPHIX/`
AISIM	`www-sop.inria.fr/epidaure/AISIM/` [INR]

Les simulateurs existant sur le marché se contentent néanmoins généralement d'animer des objets *rigides* et apprennent principalement au chirurgien le maniement des outils dans l'espace.

De par leur simplicité et leur rapidité, les masses-ressorts ont été largement utilisés, notamment par Meseure [MC96a, MC96b, Mes97] et Cover *et al.* [CEO93]. Ces approches utilisent peu de ressorts pour assurer l'interactivité et ajoutent des ressorts de longueur à vide nulle entre chaque point et la position qu'il occupait dans sa position de repos. Il définissent ainsi une forme au repos vers laquelle la surface va chercher à revenir. Sans cette surface, le maillage purement surfacique de l'objet n'aurait aucune consistance *volumique* et pourrait être aplati sans résister.

Basdogan *et al.* [BHS+98] fournissent un retour d'effort à l'utilisateur au travers de l'utilisation d'un *Phantom* [Tec] en appliquant le même genre de méthode. La force calculée est celle subie par quelques ressorts attachés à un objet fixe et simulant le comportement élastique très local de l'objet, autorisant ainsi des calculs à 1000Hz nécessaires au retour d'effort. L'affichage du modèle utilise localement des courbes splines et globalement une méthode FFD.

Citons également l'utilisation temps-réel des masses-ressorts par Desbrun dans [DSB99] qui grâce à une méthode d'intégration semi-implicite, interprétation simplifiée de [BW98], propose une animation de tissus très rapide[12].

L'algorithme de *ChainMail* de Gibson [Gib97] modélise l'objet comme un ensemble de maillons ne pouvant ni être dissociés ni être trop imbriqués (Fig. 1.8). À chaque déplacement imposé, un algorithme itératif déplace tous les maillons de façon à satisfaire ces contraintes. Purement géométrique, cette déformation est très rapide : elle est accompagnée d'une relaxation par masses-ressorts, faite uniquement lorsque l'on en a le temps, c'est-à-dire quand on ne manipule plus l'objet. L'intêret de cette combinaison est que les déformations purement plastiques du réseau de maillons sont très rapides (125000 maillons déplacés à 3Hz sur une SGI Indy).

Parmi les méthodes utilisant les éléments finis, les travaux de Bro-Nielsen [BNC96] et Cotin [Cot97] ainsi que ceux de James et Pai [JP99] offrent une interface interactive. Notons qu'aucun de ces modèles n'est réellement dynamique, les masses-tenseurs de Cotin [Cot97] réinitialisant la vitesse à 0 avant chaque pas de simulation et les deux autres ne proposant que des états d'équilibre statiques.

[12] Voir Annexe B

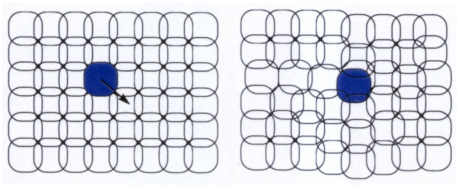


FIG. 1.8: Une juxtaposition de maillons imbriqués forme le *ChainMail*.

10 La multirésolution en animation

10.1 Plusieurs façons de faire de la multirésolution

Définissons plus précisément ce que l'on entend par multirésolution en animation. Il ne s'agit pas simplement de pouvoir avoir une résolution variable au cours de l'animation, comme c'est le cas dans [HPH96] ou [OH99] qui font évoluer la discrétisation de leur système en fonction du lieu des plis d'un tissus ou de la propagation de fractures.

Multirésolution signifie davantage avoir *à tout instant* à disposition différentes représen-tations du même mouvement et choisir en permanence celle qui convient le mieux, ce choix pouvant éventuellement dépendre de la partie de l'objet concernée. On se donne la possibilité de générer plusieurs mouvements, à des qualités et des vitesses de calcul différentes, pour n'utiliser à chaque instant que celui offrant le meilleur compromis.

Ces différentes représentations doivent néanmoins cohabiter suffisamment pour que l'utilisateur n'ait pas conscience de leur présence. Il doit avoir l'impression que le comportement reste identique au cours de la simulation, indépendant des résolutions qui cohabitent et de leur possibles modifications.

Les différents niveaux de détail peuvent être établis au départ ou être calculés lors de l'animation en fonction de ce qui s'y déroule. Ce dernier cas est plus complexe et risque, s'il prend trop de temps, de figer l'animation, ce qui est gênant dans un contexte interactif.

Ces différentes résolutions peuvent également être du même type de simulation, vues à des résolutions différentes, ou mélanger différentes techniques d'animation (dynamique, cinématique et ponctuelle par exemple dans [CH97]).

Les divers seuils qui décideront quelle résolution doit être utilisée à chaque endroit permettront de régler la vitesse de la simulation en autorisant très ponctuellement ou au contraire assez largement l'utilisation des résolutions les plus fines. Dans une approche de temps-réel, on pourra également *garantir* un délai fixe entre deux affichages en limitant l'utilisation des résolutions les plus coûteuses en fonction du temps de calcul disponible.

10.2 Difficultés

Il y a toujours deux problèmes principaux dans la mise en place des approches multirésolution :

– Les différentes résolutions que l'on veut mélanger doivent avoir, dans une certaine mesure, le *même comportement dynamique*, c'est-à-dire approximer le même mouvement au cours du temps. Si tel n'était pas le cas, la simulation serait chaotique, son comportement dépendant des résolutions employées alors que c'est précisément ce que l'on veut cacher à l'utilisateur.
– Les différentes résolutions doivent *pouvoir cohabiter*, c'est-à-dire que les *transitions* entre les niveaux ne doivent pas être visibles. Ces transitions sont d'une grande importantes et de leur discrétion dépend le résultat. Cette notion est à différencier de la précédente, car même avec un comportement dynamique identique, deux résolutions ne peuvent pas donner un bon résultat si elles ne cohabitent pas. Deux mouvements, même similaires, doivent par exemple être *en phase* à l'endroit ou l'on passe de l'un à l'autre. On devra aussi veiller à éviter les clignotements dues à une oscillation entre deux résolutions ainsi que les apparitions soudaines de détails qui ne faisaient pas partie de la résolution grossière précédente, ou tout autre indice visible dépendant de l'application.
  Dans notre cas, simuler une série de points, chacun à sa propre échelle est aisé, mais encore faut-il que ces points puissent interagir les uns avec les autres lorsqu'ils n'appartiennent pas à la même échelle, afin que l'objet forme un tout et non une simple juxtaposition de parties à différentes résolutions.

10.3 Adaptatif *versus* Hiérarchique

Pour une même zone de l'objet, les différentes résolutions qui peuvent y être employées vont l'être séparément ou collectivement :

- Ou bien une résolution fine sera la seule réellement active dans cette zone et elle devra alors être capable de cohabiter avec les zones voisines qui n'ont pas forcément la même résolution.
- Ou bien cette résolution cohabitera avec les autres résolutions de cette même zone, en leur redonnant par exemple par moyenne les informations dont elle dispose. Un certain nombre de résolutions seront alors actives en même temps dans cette zone pour représenter le phénomène.

Nous désignerons les premières méthodes par le terme d'*adaptatives* (Fig. 1.9a) et les secondes par celui de *hiérarchiques* (Fig. 1.9b). L'adaptatif n'utilisera qu'*une* résolution à un endroit donné, devant l'interfacer avec les zones voisines. Le but sera alors d'*adapter* en chaque zone la résolution en fonction de la simulation.

Le hiérarchique devra par contre gérer plusieurs résolutions simultanément à chaque endroit et être capable de les faire communiquer. On gagne en généricité (les différentes zones peuvent communiquer directement au travers de la même résolution) au prix d'un coût de calcul légèrement plus élevé dû à la simulation de plusieurs résolutions à chaque endroit.

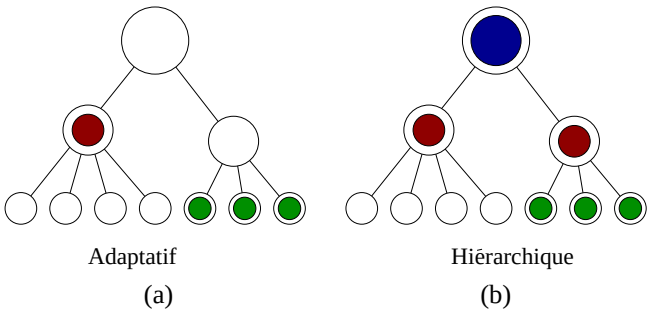


FIG. 1.9: Parmi toutes les résolutions possibles, l'adaptatif (a) ne simule que le niveau le plus fin (points) alors qu'en hiérarchique (b), les niveaux inférieurs continuent à être actifs et cohabitent avec le niveau fin.

Ces définitions ne sont pas académiques et sont introduites ici par commodité afin de pouvoir s'y référer dans la suite de ce document lorsque nous décrirons nos contributions.

10.4 Utilisations de la multirésolution

Différents articles ont tout d'abord cherché à formaliser quels étaient les types de mouvements que l'on pouvait générer et les critères permettant de choisir lequel est le mieux approprié.

Berka introduit dans [Ber97] la notion de niveau de détail *dynamique* (DLOD). Il énumère quelques critères de décision quand au choix du niveau de détail : l'amplitude observable à l'écran du mouvement, sa vitesse et le nombre des objets qui l'adoptent, permettent éventuellement de les regrouper. Il soulève, sans toutefois y répondre, les problèmes qui apparaissent alors : comment mesurer ces paramètres, quel sera le mouvement simplifié et comment gérer les transitions entre les niveaux ?

Chenney et Forsyth [CF97] vont eux étudier à quel point on peut simplifier le calcul de l'animation des objets temporairement non visibles. Il faut pour cela que l'état (position, posture) dans lequel les objets réapparaissent soit cohérent avec ce dont l'utilisateur peut se souvenir. Le but va donc être de reproduire la *statistique* décrivant les états de l'objet ; plus celle-ci est aléatoire, plus on pourra simplifier les calculs consacrés aux objets non visibles. Les gains de temps annoncés par les auteurs varient entre 1.5 et 5.

Dans sa thèse [Val99], Valton décrit un mécanisme de décomposition en niveaux de détail de modèles d'animation. Une structure d'arbre comportant différents modules découplés, pouvant être désactivés lorsqu'il faut dégrader le mouvement, permettent de régler la qualité de l'animation. Une grammaire permet de corréler les changements de niveaux de détail dans l'animation et dans le modèle géométrique affiché.

Carlson et Hodgins ont proposé un modèle utilisant trois niveaux de détail dans l'animation par modèle physique de "pistons" bondissants [CH97]. Selon que chaque objet est proche ou lointain, ou utilisera une simulation dynamique, cinématique, voire simplement basée sur des particules en ne considérant l'objet que comme un point. Même si chaque niveau n'a pas tout à fait le même comportement, on obtient quand même ainsi un gain sur le temps de calcul.

Rappelons dans l'approche modale de Pentland et Williams [PW89] la possible utilisation multirésolution d'un nombre donné de modes vibratoires permettant de modifier le compromis vitesse-qualité.

En utilisant cette fois-ci un réseau masses-ressorts, Hutchinson *et al.* proposent une méthode adaptative de simulation de tissus [HPH96]. Celui-ci est tout d'abord maillé grossièrement, mais de nouvelles masses sont ajoutées en fonction de l'animation lorsque l'angle entre deux ressorts devient trop important pour affiner le résultat. On accélère ainsi le début de la simulation dans leur exemple d'une nappe tombant sur une table et se pliant au bord. Le problème principal de la méthode est que l'adjonction de ressorts modifie sensiblement le comportement du tissus, chaque point ajouté apportant par exemple dans leur implémentation une masse supplémentaire. Rien n'est de plus prévu pour resimplifier localement le maillage si l'objet change de position.

Ganovelli a lui aussi développé des approches multirésolution à base de masses-ressorts. Ses premiers travaux [GCS99] cherchaient à déterminer la raideur équivalente à plusieurs ressorts en calculant leur résistance à un fort étirement (tous les ressorts ayant atteint leur longueur limite). Il a ensuite utilisé les raideurs de Van Gelder [Gel98] dans un modèle autorisant les découpes [GCMS00]. La faible indépendance à la résolution de ce modèle (voir Chapitre 4, Section 7) grève néanmoins cette méthode.

Desbrun développe lui aussi un modèle multirésolution (voir Sec. 5.3) en temps (le pas de temps d'intégration de chaque particule peut varier) et en espace (les particules peuvent se diviser et se regrouper) basé sur le formalisme SPH [DCG96, Des97, GCD$^+$98, DC99b]. Cette méthode a été développée pour des matériaux hautement déformables, et les essais que nous avons faits d'application à des matériaux plus élastiques que plastiques n'ont pas été concluants. C'est néanmoins ce genre d'approche qui nous inspirera et que nous chercherons à appliquer.

Parmi les méthodes utilisant des éléments finis, citons les travaux de Hayward [AH97] qui, par une analogie avec les impédances équivalentes de circuits électriques, pouvait séparer l'objet simulé en 2 parties, l'une fine et l'autre plus grossière, chacune ne voyant l'autre que par un modèle équivalent simplifié au travers des noeuds situés à l'interface.

Zhuang décrit dans sa thèse [Zhu00] un formalisme à base d'éléments finis permettant de simuler en temps-réel les déformations d'objets déformables. La discrétisation, fixe au cours du temps, n'est pas uniforme : plus fine à proximité de la surface, elle permet d'obtenir plus de précision dans la zone d'interaction.

Nous avons déjà décrit le maillage adaptatif généré par les fractures dans [OH99]. Même si le maillage évolue pour s'adapter à la simulation, on ne peut toutefois ici, pas plus que dans les masses-ressorts de Hutchinson [HPH96], parler de multirésolution. Il faudrait pour cela que l'on puisse par exemple pour [OH99] à nouveau simplifier le maillage des morceaux qui se détachent de l'objet et ont un simple vol balistique ne nécessitant pas l'animation de tous les tétraèdres dont ils sont constitués.

11 Choix d'un modèle

11.1 Méthodes interactives

Il existe déjà des méthodes permettant de simuler interactivement des matériaux déformables, dans un contexte chirurgical par exemple. On pourrait dire que plus rien n'est à faire, certaines de ces méthodes "permettant une vitesse de rafraîchissement de 15 Hz". Cet argument n'a cependant aucune valeur, le taux de rafraîchissement ne signifiant strictement rien si l'on ne sait pas à quelle durée d'animation physique correspond l'intervalle entre deux affichages. Si les durées ne sont pas les mêmes, le mouvement apparaîtra simplement dans ces cas comme très ralenti, bien qu'affiché à 25 images par secondes.

11.2 Le temps réel vrai

Sont généralement désignées comme temps-réel toutes les méthodes permettant l'affichage de plusieurs images par seconde, définition bien trop imprécise que nous préférons restreindre. Une application temps-réel *vrai* synchronise à chaque image affichée le temps ressenti par la personne qui la regarde à celui correspondant à l'étape de simulation affichée. Il faut en d'autres termes que lorsque deux images sont affichées à une seconde d'intervalle, la machine ait effectivement calculé le comportement du matériau durant une seconde, ni plus, ni moins. Dans le cas contraire, le mouvement apparaîtra respectivement comme joué en accéléré ou au ralenti.

Ce genre de considération n'est que très rarement pris en compte [Luc85]. On choisit généralement de simuler le nombre maximal de points d'échantillonnage qui permette de conserver un affichage de plusieurs images par secondes. Dès que le matériau est un peu rigide, le pas de temps d'intégration doit être baissé pour limiter la divergence de l'intégration [13]. Les dix à cinquante images affichées par seconde (après chaque pas de simulation) n'ont alors plus rien à voir avec l'intervalle de un millième de seconde ou moins qui est supposé les séparer.

Que l'animation ne soit pas affichée à la bonne vitesse ne gène pas trop si on ne cherche qu'un réalisme visuel. On pourra chercher à jouer sur les coefficients pour obtenir l'effet recherché (on aura tout de même du mal à créer un corps rigide et ayant beaucoup de particules). Il faut également garantir que chaque pas de simulation prendra le même temps machine à être calculé pour être sûr que l'animation se fera à une vitesse éventuellement fausse, mais au moins constante.

11.3 Nécessité de la multirésolution

Le temps de calcul d'une seconde d'animation dépend linéairement du nombre n de points simulés et du temps mis pour calculer une force en chaque point [14]. Si l'on admet que le temps de calcul d'une force en un point est une constante, on est limité par le nombre de points que l'on peut simuler. Il faut donc employer des méthodes multirésolution si on veut accroître le réalisme visuel et *optimiser* l'utilisation de ces n points, en les plaçant à chaque instant au bon endroit dans le matériau.

Dans des animations multirésolution encore plus que dans les autres méthodes, la charge de calcul va pouvoir varier au cours de l'animation, même si on cherchera à l'utiliser à son maximum. Il conviendra donc de bien synchroniser calcul et affichage pour obtenir ce que nous avons appelé le temps-réel vrai.

Mettre en place de telles méthodes nécessite un formalisme suffisamment indépendant de la résolution employée pour que l'on puisse espérer simuler des discrétisations différentes sans que cela n'apparaisse trop.

11.4 Modèle indépendant de la résolution.

Nous allons ici chercher, à la lumière de l'état de l'art qui précède, quelle sont les méthodes permettant une simulation indépendante de la résolution.

Le comportement des réseaux masses-ressorts est directement lié au maillage utilisé. On ne peut espérer faire facilement cohabiter des maillages trop différents ayant le même comportement dynamique. Van Gelder a démontré dans un article [Gel98] *l'impossibilité* de régler les raideurs d'un réseau masses-ressorts pour le faire coïncider avec un modèle continu de la matière et de le rendre indépendant de la résolution.

Van Gelder présente les deux méthodes dans un même formalisme matriciel (sur un réseau à trois points fait de masses-ressorts ou d'éléments finis) et cherche à déterminer les raideurs à donner aux ressorts pour les faire coïncider. Il fournit un contre exemple simple prouvant que c'est impossible. En simplifiant considérablement le modèle continu de la matière [15], Van Gelder propose de choisir la raideur k de *chaque* ressort selon :

$$k = \frac{E\ V}{l^2}$$

pour minimiser la différence entre les deux méthodes. E est le module d'Young [16], représentant la raideur globale que l'on souhaite donner au matériau. V est la somme des volumes des tétraèdres (des aires des triangles en 2D) adjacents à l'arête où est le ressort, alors que l est sa longueur au repos.

Ces considérations ne sauraient totalement supprimer le trop fort lien existant entre le maillage et le comportement d'un réseau masses-ressorts, ce qui les rend inutilisables en multirésolution.

Parmi les méthodes présentées, seules celles basées sur l'utilisation d'une équation d'état caractérisant le matériau garantissent un comportement indépendant (dans une certaine mesure) de la résolution employée. On se tournera donc vers les SPH et les méthodes de type élément finis.

[13] Voir annexe B

[14] On exclut donc d'emblée les méthodes matricielles de type éléments finis implicites dont la résolution est en $O(\hat{n})$ au mieux.

[15] On supprime toute notion d'incompressibilité dans le matériau, ce qui revient à prendre $\nu = 0$ avec les notations introduites au chapitre suivant.

[16] Voir le chapitre 2.

Nos premiers essais ont été réalisés avec une méthode à base de SPH. Ils n'ont pas permis la simulation de matériaux trop rigides, le modèle étant mieux adapté aux interactions inter-particules faibles. En effet, l'approximation d'une intégrale continue par une somme discrète telle qu'on le fait en Section 5.1 n'est possible que si le volume sur lequel on intègre est suffisamment bien échantillonné. Les simulations astrophysiques gèrent plusieurs millions de points qui font que cette approximation est valable. Dans notre cas, les contraintes de temps-réel font que l'on doit se contenter de quelques centaines de points au plus, les noyaux de convolution n'en englobant qu'une partie dans chaque intégrale, ce qui rend l'approximation bien plus grossière.

Libersky et Petschek simulent bien un matériau très rigide [DP90], mais là encore il utilisent beaucoup de points et les temps de calcul sont prohibitifs, même pour leur exemple 2D.

Les résultats visuels que nous avons constatés font état d'un matériau très mou, tendant plus vers le liquide ou le boueux que vers le solide, ce qui était bien adapté aux objectifs de Desbrun. Obtenir des matériaux rigides en augmentant la cohésion demande un grand nombre de points simulés et des pas de temps très faibles pour que l'animation ne diverge pas, ce qui rend cette méthode difficilement compatible avec notre objectif de temps-réel.

C'est vers des méthodes utilisant une loi de comportement globale, basées sur une vision continue de la matière que nous allons nous tourner. La solution semblait alors devoir passer par l'utilisation de méthodes de type éléments finis, assez complexes et très coûteuses puisque impliquant des résolutions matricielles. Les méthodes explicites nous étaient inconnues aux débuts de nos travaux et nous avons cherché à développer des méthodes similaires, impliquant le calcul direct d'une force en chaque point, intégrée séparément par la suite comme dans un système de particules. Nous comparerons par la suite les résultats obtenus avec ceux de certaines méthodes d'éléments finis. Seront alors élaborés des algorithmes permettant une utilisation multirésolution de ces calculs.

12 Conclusion générale

Ce chapitre a présenté les grandes classes de méthodes utilisées dans la simulation d'objets déformables ainsi que les principaux articles s'y rapportant. Des conclusions partielles explicitaient leurs avantages et leurs inconvénients, ainsi que leur adéquation avec nos buts de simulation temps-réel tirant parti de la multirésolution.

Nous avons vu qu'une solution pouvait être le mélange des méthodes à base de particules de type masses-ressorts (pour leur rapidité), et de forces issues de la description continue de la matière faite par les physiciens (pour leur indépendance à la résolution).

Nous allons en quelque sorte mettre au point un système de particules amélioré en cherchant une expression de la force entre deux points qui soit liée aux caractéristiques physiques du matériau. Nous allons pour cela étudier dans le chapitre suivant le formalisme utilisé par les physiciens pour décrire la matière comme un milieu continu. Nous verrons par la suite comment calculer ces forces et mettre alors en place les méthodes multirésolution qui en tireront parti.

CHAPITRE 2

Notions d'élasticité linéaire

A beautiful story needs a beautiful language to tell.
Tensor is the language of mechanics.

Y. C. FUNG, *A first Course in Continuum Mechanics*

Ut tensio sic vis.
The power of any springy body is in the same proportion with the extension.

Robert HOOKE, 1635-1703

LES PHYSICIENS ont depuis des siècles tenté de mettre en équations le comportement de la matière. Plus ou moins générales, précises ou robustes, ces modélisations s'appuient sur la représentation du phénomène de déformation à l'aide de champs vectoriels et de tenseurs. Ceux-ci décrivent en particulier la *déformation* de l'objet ainsi que les *contraintes internes* qu'il subit. Des *lois de comportement* viennent ensuite lier les contraintes et la déformation qui en résulte. Ce chapitre présente succinctement cette modélisation physique et les équations principales que nous utiliserons dans les chapitres suivants. Le lecteur non familier avec les notations différentielles peut se reporter à l'Annexe A pour une brève introduction.

Les modèles physiques présentés sont parmi les plus simples. Il ne prennent pas en compte des phénomènes tels que l'influence et la variation de la température à l'intérieur du matériau, la possible anisotropie de ce dernier, etc. Ils ne sont donc pas forcément les mieux adaptés à une simulation spécifique, mais ont l'avantage de la simplicité et de la polyvalence. Pour une présentation plus complète, le lecteur pourra se référer à [LL59, TG70] (académiques), [Ger62, Cia85] (en français), [Dar95] (clair et simple) ou [Fun65, CMP89, SBG96] (plus complexe). Citons aussi [MWTT98] qui détaille plus précisément les modèles de matériaux dédiés aux applications bio-médicales.

Nous détaillons tout d'abord la modélisation des déformations en présentant deux modèles de déformation distincts. Nous présentons ensuite la modélisation des contraintes que subit le matériau puis la loi de comportement qui relie contraintes et déformations. Nous présentons enfin la façon dont sont exprimées les forces de frottement.

1 Le tenseur des déformations

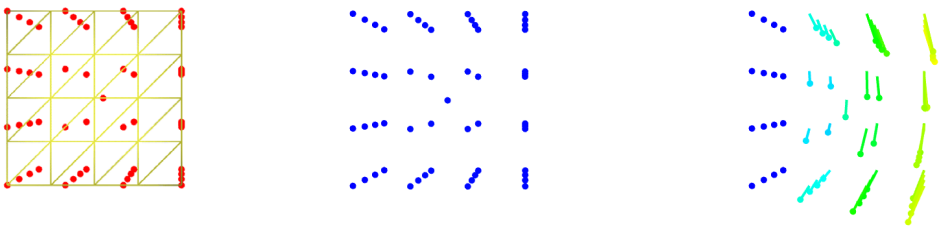
Nous présentons ici deux façons de représenter les déformations que subit un matériau, l'une étant la version simplifiée de l'autre. Chacune va, pour représenter les déformations du matériau, définir en chaque point un *tenseur des déformations*, noté ε, plus ou moins complexe. Les deux modèles ont leurs intérêts et leurs inconvénients, aussi les comparons-nous par la suite.

1.1 Le tenseur de Cauchy

Le champ de déplacement

Les déformations d'un objet sont mesurées à partir d'une position initiale, qui est généralement la position de repos de l'objet dans laquelle aucune force n'est appliquée à l'objet.

On définit alors le champ vectoriel *déplacement*, généralement noté $\mathbf{u}$, qui est simplement pour chaque point le vecteur reliant sa position au repos à sa position actuelle dans la configuration déformée (voir Fig. 2.1). En chaque point, et à chaque instant, on définit donc : $\mathbf{u}_t = \mathbf{p}_t - \mathbf{p}_0$ où l'indice désigne ici le temps et $\mathbf{p}$ la position du point dans un repère galiléen fixe.



Position de repos — Points d'échantillonnage — Champ de déplacement

FIG. 2.1: Un champ de déplacement dans un cube en 3D, représenté en chaque point par un vecteur.

Décomposition du champ

On n'utilise plus ensuite que la première dérivée spatiale de ce champ $\mathbf{u}$ au travers de la matrice A qui est son gradient[1] :

$$A = \mathbf{grad}\,\mathbf{u} = \begin{pmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_x}{\partial y} & \frac{\partial u_x}{\partial z} \\ \frac{\partial u_y}{\partial x} & \frac{\partial u_y}{\partial y} & \frac{\partial u_y}{\partial z} \\ \frac{\partial u_z}{\partial x} & \frac{\partial u_z}{\partial y} & \frac{\partial u_z}{\partial z} \end{pmatrix} \tag{2.1}$$

Cette matrice A peut être séparée en la somme de deux matrices 3×3 :

$$A = \mathbf{grad}\,\mathbf{u} = \underbrace{\frac{1}{2}\left(\mathbf{grad}\,\mathbf{u} + \mathbf{grad}\,\mathbf{u}^T\right)}_{E} + \underbrace{\frac{1}{2}\left(\mathbf{grad}\,\mathbf{u} - \mathbf{grad}\,\mathbf{u}^T\right)}_{R} \tag{2.2}$$

E est une matrice symétrique tandis que R est antisymétrique et est liée au rotationnel $\mathbf{rot}\,\mathbf{u}$ du champ de déplacement.

Définition

Dans le *modèle de Cauchy* que nous décrivons ici, c'est la matrice E qui va représenter les déformations du matériau et va donc être le *tenseur des déformations*, ε_{Cauchy}. R, que nous n'utiliserons pas dans les modèles simplifiés présentés, est appelé *tenseur de rotation*.

[1] Cette matrice est formée, sur chacune de ses lignes, du gradient de la composante associée du vecteur $\mathbf{u}$. Voir l'Annexe A pour des définitions plus complètes.

On a donc :

$$\varepsilon_{Cauchy} = \begin{pmatrix} \frac{\partial u_x}{\partial x} & \frac{1}{2}(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}) & \frac{1}{2}(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x}) \\ \frac{1}{2}(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}) & \frac{\partial u_y}{\partial y} & \frac{1}{2}(\frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y}) \\ \frac{1}{2}(\frac{\partial u_z}{\partial x} + \frac{\partial u_x}{\partial z}) & \frac{1}{2}(\frac{\partial u_z}{\partial y} + \frac{\partial u_y}{\partial z}) & \frac{\partial u_z}{\partial z} \end{pmatrix} \tag{2.3}$$

En désignant par x_i la $i^{ème}$ direction du *repère du monde* (fixe et galiléen) dans lequel on dérive ($x_1 = x, x_2 = y, x_3 = z$), les termes de la matrice peuvent s'écrire :

$$(\varepsilon_{Cauchy})_{ij} = \frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}) \tag{2.4}$$

Propriétés

L'opération de dérivée spatiale que l'on applique pour obtenir A *filtre* le champ **u** pour n'en plus considérer que son *évolution* au travers du matériau. Une *translation* appliquée à tout l'objet se traduit par l'ajout d'une constante au champ **u**, qui sera supprimée par la dérivée. Ne modifiant pas le tenseur des déformations, cette translation ne créera pas de force additionnelle. Si l'objet est donc insensible aux translations, il ne l'est par contre pas à l'autre mouvement rigide qu'est la rotation qui elle entraînera des forces internes et donc une déformation.

Ce comportement est irréaliste, mais il s'avère suffisant dans la plupart des cas, lorsqu'on se limite à de faibles déplacements et que l'objet reste donc proche de sa position d'équilibre.

Bien évidemment, lorsque l'objet est dans sa position de repos, le champ **u** est uniformément nul et le tenseur ε l'est aussi, ce qui ne créera aucune force comme on le verra dans une section suivante. La position de repos est donc une position stable.

1.2 Le tenseur de Green-Lagrange

Le repère lié au matériau

Pour le tenseur de Green-Lagrange, on définit deux systèmes de coordonnées, liés à deux repères différents : le repère du monde, fixe, et dans lequel seront mesurées les positions des points, et celui, local et lié au matériau, défini par la position de repos de l'objet. En l'absence de déformation, ces deux référentiels coïncident à une transformation rigide près. C'est leur "écartement", encore une fois filtré par une dérivée spatiale, que va mesurer le tenseur des déformations de Green-Lagrange.

On désignera par Ω le système de coordonnées lié au repère du matériau. Les dérivées par rapport à la $i^{ème}$ coordonnée de ce repère seront donc notées $\frac{\partial}{\partial \Omega_i}$.

Définition

Le tenseur de Green-Lagrange est défini par :

$$\varepsilon_{Green-Lagrange} = \mathbf{grad}_\Omega\, \mathbf{p} \times \mathbf{grad}_\Omega\, \mathbf{p} - I_3 \tag{2.5}$$

où les termes $\mathbf{grad}_\Omega\, \mathbf{p}$ sont des matrices 3×3 de gradient de la position, définies dans le repère lié au matériau :

$$(\mathbf{grad}_\Omega \mathbf{p})_{ij} = \frac{\partial p_i}{\partial \Omega_j}$$

dont on fait le produit matriciel et auxquels on soustrait la matrice identité de taille 3, I_3.

Chaque terme de la matrice ε peut donc s'écrire :

$$(\varepsilon_{Green-Lagrange})_{ij} = \left(\frac{\partial \mathbf{p}}{\partial \Omega_i} \cdot \frac{\partial \mathbf{p}}{\partial \Omega_j}\right) - \delta_{ij} \tag{2.6}$$

où δ_{ij} est le symbole de Kronecker ($\delta_{ij} = 1$ si $i = j$ et 0 sinon). Chaque terme de ε_{ij} est ainsi le produit scalaire de deux vecteurs, auquel on soustrait éventuellement 1.

Propriétés

Cette définition du tenseur provient de l'étude de la variation du *carré de la distance* entre deux points voisins, avant et après déformation. On n'a pas de racine carrée inhérente à une distance, mais restent des termes quadratiques que l'on retrouve dans l'expression du tenseur, qui *n'est donc pas linéaire*.

Le tenseur de Green-Lagrange est une matrice symétrique (puisque c'est un tenseur) et a plusieurs propriétés intéressantes : la première est que, tout comme pour le tenseur de Cauchy, le tenseur de Green-Lagrange est nul lorsque l'objet est dans sa position de repos. En effet, supposons dans un premier temps et pour simplifier que l'on a pris comme référentiel lié au matériau le référentiel global[2]. Alors la matrice $\mathbf{grad}_\Omega\ \mathbf{p}$ n'est autre que l'identité puisqu'elle est le gradient de la position par rapport à la position. La matrice ε est donc nulle (après soustraction de I_3, voir Eq. 2.5).

Ce tenseur est, tout comme Cauchy, également insensible aux translations globales de l'objet (ajout d'une constante au champ $\mathbf{p}$ qui disparaît dans la dérivée). Il a de plus l'importante propriété de ne pas être modifié par des *rotations globales* de l'objet *dans sa forme de repos*. Si l'on se trouve dans la position de repos, les dérivées $\frac{\partial \mathbf{p}}{\partial \Omega_i}$ qui servent à calculer les ε_{ij} sont les lignes de la matrice identité, c'est-à-dire des vecteurs de la forme $(\delta_{1i}, \delta_{2i}, \delta_{3i})^T$. Faire tourner rigidement l'objet revient à appliquer à la position de ses points une matrice de rotation O. Les nouvelles positions $\mathbf{p}'$ sont alors $\mathbf{p}' = O\ \mathbf{p}$. On a donc, pour chaque terme du tenseur (Eq. 2.6) :

$$\begin{aligned}
\varepsilon_{ij} &= \left(\frac{\partial \mathbf{p}'}{\partial \Omega_i} \cdot \frac{\partial \mathbf{p}'}{\partial \Omega_j}\right) - \delta_{ij} \\
&= \left(\frac{\partial O\ \mathbf{p}}{\partial \Omega_i} \cdot \frac{\partial O\ \mathbf{p}}{\partial \Omega_j}\right) - \delta_{ij} \\
&= \left(O\ (\delta_{1i}, \delta_{2i}, \delta_{3i})^T \cdot O\ (\delta_{1j}, \delta_{2j}, \delta_{3j})^T\right) - \delta_{ij} \\
&= \left(\left(i^{ème}\ ligne\ de\ O\right) \cdot \left(j^{ème}\ ligne\ de\ O\right)\right) - \delta_{ij} \qquad (2.7)
\end{aligned}$$

Mais puisque O est une matrice de rotation, le produit scalaire de ses lignes i et j vaut précisément δ_{ij}. Le tenseur ε est donc bien nul après que l'on a appliqué la rotation. Il est à noter que dans le cas général, la multiplication par la matrice O ne préserve pas le tenseur et que c'est parce qu'on est dans la forme de repos et que les vecteurs sont alors composés de 1 et de 0 que l'on peut ainsi simplifier. Une rotation rigide d'un objet *déformé* modifiera donc le tenseur.

L'annexe C décrit en détails l'implémentation de ce formalisme dans un cadre d'éléments finis *explicites* (*cf* Section 7.2 du chapitre précédent).

1.3 Comparaison des deux modèles

Différences mathématiques

Les deux modèles se différencient par le repère par rapport auquel les dérivées spatiales sont calculées. Alors que Cauchy dérive par rapport au repère du monde, Green-Lagrange dérive dans le repère lié au matériau. Cauchy est désigné comme *Eulérien* (on mesure tout dans un repère fixe), alors que Green-Lagrange est *Lagrangien* (on "suit" les points dans leur mouvement, en mesurant par rapport à un repère lié au matériau). Ces deux façons de mesurer les choses sont équivalentes, et l'on peut passer de l'une à l'autre en prenant en compte le mouvement d'un repère par rapport à l'autre.

Outre cette différence, le modèle de Cauchy, plus simple, peut être vu comme une approximation au premier ordre de celui de Green-Lagrange. Alors que Cauchy est une combinaison linéaire de dérivées premières, Green-Lagrange introduit en plus un terme quadratique. Si l'on réécrit en effet les équations de Green-Lagrange en fonction du déplacement et non plus de la position, on obtient :

$$(\varepsilon_{Green-Lagrange})_{ij} = \frac{1}{2}\left[\frac{\partial u_i}{\partial \Omega_j} + \frac{\partial u_j}{\partial \Omega_i} + \left(\frac{\partial u_1}{\partial \Omega_i}\frac{\partial u_1}{\partial \Omega_j} + \frac{\partial u_2}{\partial \Omega_i}\frac{\partial u_2}{\partial \Omega_j} + \frac{\partial u_3}{\partial \Omega_i}\frac{\partial u_3}{\partial \Omega_j}\right)\right] \qquad (2.8)$$

Si l'on remplace alors les dérivées par rapport au repère du matériau par des dérivées dans le repère du monde (Ω_1 devenant x ; Ω_2, y et Ω_3, z), on reconnaît dans les deux premiers termes de $\varepsilon_{Green-Lagrange}$ ceux qui

[2] On va montrer par la suite que la transformation rigide qui peut exister entre ces deux repères *quand on est dans la position d'équilibre* ne change pas le tenseur, qui sera donc nul quel que soit le choix du repère du matériau.

forment ε_{Cauchy} (Eq. 2.4). Ce remplacement de dérivées, cette assimilation de l'Eulérien et du Lagrangien, n'est possible que lorsque les deux repères sont très proches, ce qui est le cas uniquement dans la position de repos de l'objet. Pour de très faibles déformations, les termes quadratiques de Green-Lagrange devenant négligeables, les deux modèles sont donc équivalents. Le tenseur de Cauchy est d'ailleurs appelé tenseur *infinitésimal* de Cauchy. Ils se différencient néanmoins assez rapidement pour des déplacements plus importants.

On peut en fait définir quatre tenseurs en combinant l'utilisation ou non des termes quadratiques additionnels et la dérivation dans le repère du monde ou dans celui du matériau. Le tenseur utilisant les termes quadratiques, mais défini dans le repère du monde s'appelle d'ailleurs *tenseur d'Almansi (et Hamel)*.

Différences comportementales

Pour illustrer la différence entre les deux tenseurs, nous présentons quelques images extraites de deux animations. Nous simulons le mouvement d'un cube sous l'action de la gravité, sa face de gauche étant fixée à un mur fixe (non représenté). Au départ, le cube est placé, sans vitesse initiale dans sa position de repos (Fig. 2.2) et nous présentons quelques images de sa première oscillation.

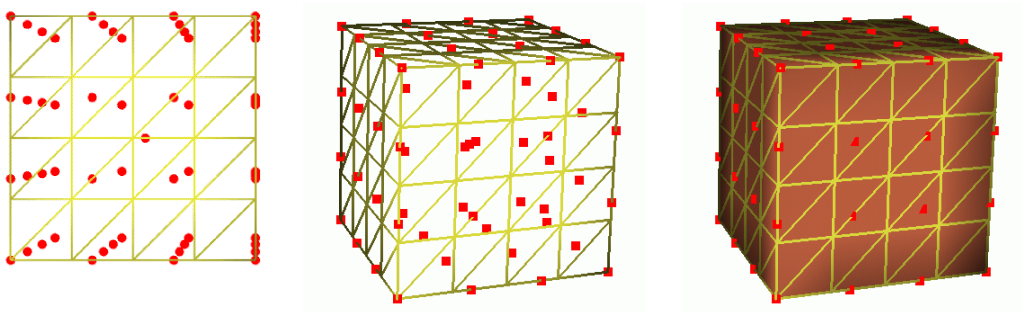
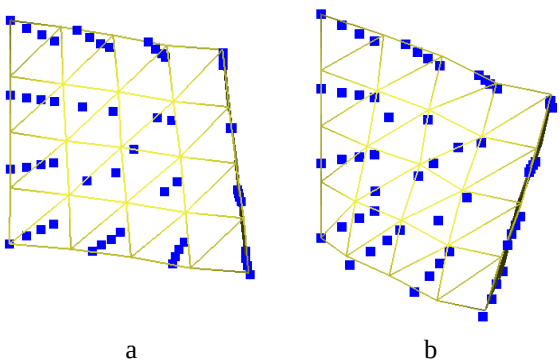


FIG. 2.2: Le cube dans sa position de repos. Vues de face et de profil des points qui vont être simulés.

Tous paramètres par ailleurs égaux[3], la méthode d'intégration étant la même, seul le mode de calcul du tenseur des déformations différencie les deux animations. Les images des Figures 2.4 et 2.5 montrent en chaque point le déplacement, la vitesse et la force subie à des intervalles réguliers de la simulation.



a b

FIG. 2.3: La position d'équilibre finale du cube lorsqu'on ajoute du frottement interne est différente en utilisant Cauchy (a) et Green-Lagrange (b).

Comme on peut le voir, le tenseur de Green-Lagrange donne une animation plus naturelle, le cube s'arquant, alors qu'il a plus tendance à s'effondrer avec Cauchy (voir les positions d'équilibre des deux animations lorsqu'on ajoute un frottement interne pour les stabiliser en Figure 2.3). Le tenseur de Cauchy montre là sa limitation aux très faibles déplacements. Plus complexe, Green-Lagrange donne donc une animation plus réaliste, mais par contre plus lente à calculer (environ 2.3 fois), car nécessitant plus de calculs. L'autre défaut du tenseur de Green-Lagrange est sa plus grande instabilité numérique, contrepartie d'un mouvement plus complexe.

[3] $\mu = 7000, \lambda = 20000$, aucun frottement, pas de temps de 8.3 10^{-4} secondes

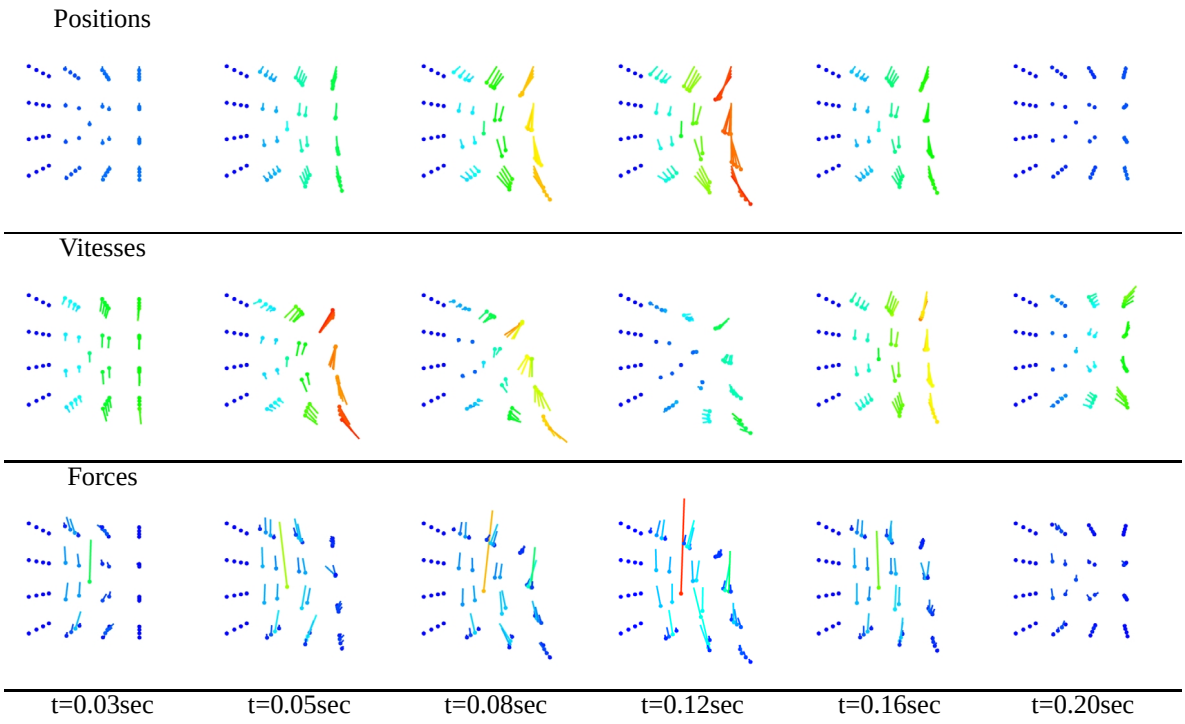


FIG. 2.4: Images de l'animation d'un cube sous l'action de la gravité avec le tenseur de Cauchy.

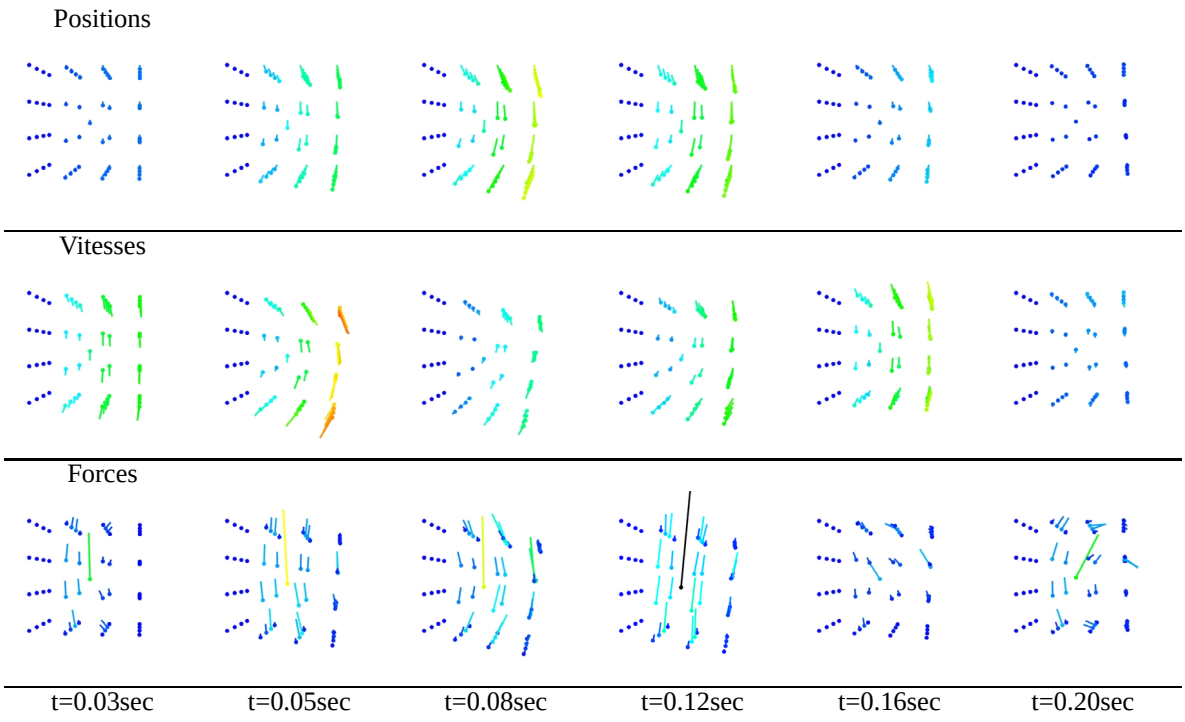


FIG. 2.5: Images de l'animation d'un cube sous l'action de la gravité avec le tenseur de Green-Lagrange.

2 Le tenseur des contraintes

Après la description des déformations, nous introduisons ici le *tenseur des contraintes* qui va décrire en chaque point la répartition des forces internes. Nous verrons ensuite comment relier ce tenseur aux forces et accélérations réellement subies par les points.

2.1 Définition

Le tenseur des contraintes, noté σ, est également une matrice 3×3 symétrique. C'est tout comme le tenseur des déformations, une approximation assez grossière de ce qui se passe en chaque point du matériau.

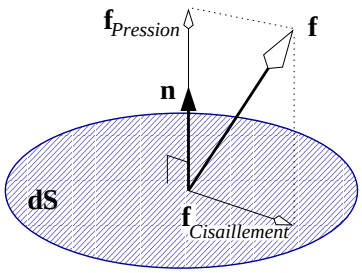


FIG. 2.6: Décomposition de la force agissant sur un élément de surface dS défini par sa normale $\mathbf{n}$.

Considérons, en un point donné de l'objet, un élément infinitésimal de surface dS et de normale $\mathbf{n}$. Ce petit élément de surface reçoit une force $\mathbf{f}$ qui dépend de l'orientation de sa normale. On peut idéalement mesurer l'évolution de $\mathbf{f}$ en fonction de celle de $\mathbf{n}$ et l'écrire sous la forme :

$$\mathbf{f} = \Theta(\mathbf{n}) \;\; dS \tag{2.9}$$

Proportionnelle à l'aire, la force (intensité et direction) dépend au travers de la fonction Θ de la normale $\mathbf{n}$.

La fonction Θ peut être arbitrairement complexe. On pourra tout de même lui imposer d'être symétrique ($\Theta(-\mathbf{n}) = \Theta(\mathbf{n})$), indiquant par là que la force reçue par le petit élément de surface sera la même qu'on le considère retourné ($\mathbf{n}' = -\mathbf{n}$) ou non.

Si la fonction Θ est une simple multiplication par un scalaire, la force est une *pression uniforme* : elle s'applique le long de la normale à la surface et a une valeur constante dans toutes les directions.

Si Θ peut être représentée par une matrice 3×3 de la forme $\begin{pmatrix} P & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, on a cette fois affaire à un champ de pression directif (ici selon l'axe x, mais ce pourrait être dans n'importe quelle direction en multipliant par une matrice de rotation). L'élément de surface reçoit une pression P lorsqu'il est orienté le long de la direction x, et une force nulle quand il est orienté dans une direction orthogonale.

Un peu plus complexe, la classe des tenseurs de contrainte que nous allons utiliser est celle où la fonction Θ peut-être représentée par une matrice 3×3 quelconque. Cette matrice, qui sera dénotée σ, sera symétrique pour les raisons vues plus haut. L'application linéaire σ pourra donc être considérée comme *l'approximation au premier ordre* de Θ, qui elle peut être quelconque. Une force linéairement dépendante de la normale est évidemment une approximation, mais elle se révèle suffisante pour des simulations générales. On a donc :

$$\mathbf{f} = \sigma \,.\, \mathbf{n} \;\; dS \tag{2.10}$$

2.2 Force et accélération

Ce tenseur des contraintes σ est une notion assez abstraite des contraintes internes. On peut le relier simplement à l'*accélération* $\mathbf{a}$ que subit chaque point par l'équation :

$$\rho \mathbf{a} = \mathbf{div}\, \sigma \tag{2.11}$$

où ρ est la masse volumique du matériau au point considéré et **div** est le vecteur composé des divergences de chacune des lignes de la matrice σ (voir Annexe A).

L'accélération est inversement proportionnelle à la densité, ce qui se comprend intuitivement : pour une contrainte σ donnée, l'accélération subie sera d'autant moins forte que le matériau est lourd et donc difficile à déplacer, et inversement s'il est léger. L'opérateur divergence va ici indiquer la direction que doit suivre le point pour tenter d'uniformiser le champ σ (une contrainte d'un côté va, si elle n'est pas compensée de l'autre côté, avoir tendance à "pousser" le point dans cette direction).

La notion de force est quant à elle liée à celle de masse. La loi fondamentale de la dynamique stipule que $\mathbf{f} = m\,\mathbf{a}$ où m est la masse du point considéré. Depuis le début de ce chapitre, nous nommons 'point' une position

quelconque à l'intérieur du matériau, qui est supposé continu. Le point, tel que nous l'entendons n'a donc pas de volume donc pas de masse. Pour pouvoir parler de force, il faut isoler un volume V, même infinitésimal, et considérer le tenseur des contraintes comme constant sur ce volume pour obtenir une force. On parlera alors plutôt de *force volumique* en écrivant :

$$\frac{\mathbf{f}}{V} = \frac{m\mathbf{a}}{V} = \frac{\rho V\mathbf{a}}{V} = \rho\mathbf{a} = \mathbf{div}\,\sigma$$

3 La loi de comportement

La loi de comportement va relier contraintes et déformation, en introduisant des propriétés propres au matériau considéré. Ces lois peuvent encore une fois être bien plus complexes que celle présentée ici, en faisant par exemple intervenir la température du matériau. Nous présentons la loi de comportement que nous utiliserons, ainsi que l'équation de Navier qui en résulte dans le cadre des petits déplacements et son interprétation comme la superposition de deux phénomènes simples.

3.1 Définition

La loi de comportement va relier les deux tenseurs définis précédemment (déformations et contraintes). Comme cela se passe dans un ressort, on va généralement imposer que la contrainte soit linéairement liée à la déformation. On pourrait, mais c'est très rare, introduire des termes de plus haut degré. Pour simuler des comportements plus complexes et en particulier non linéaires, on préfère utiliser une fonction linéaire par morceaux pour approximer la courbe non linéaire qui relie contraintes et déformation.

On voit sur la Figure 2.7 une illustration de ce qui se passe *au delà* du comportement élastique : entre les états A et B, le matériau est élastique ; il devient ensuite plastique si l'on augmente la déformation jusqu'en C. Relâcher la contrainte en C fera revenir le matériau dans une nouvelle position d'équilibre D, selon une pente égale à celle de son comportement élastique.

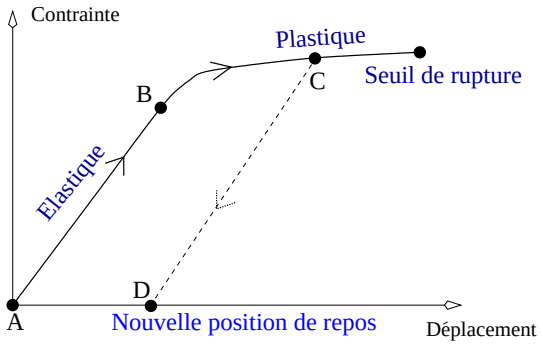


FIG. 2.7: Type de comportement d'un matériau au delà des hypothèses de linéarité.

La loi linéaire que nous allons considérer est de la forme $\sigma = K\varepsilon$ où K est une fonction linéaire [4]. Les deux tenseurs σ et ε sont des matrices 3×3 symétriques et ont donc chacun 6 coefficients indépendants. K comporte donc 36 coefficients indépendants qui lient linéairement ceux des deux tenseurs. Un matériau, dans le cadre des approximations linéaires qui ont été faites depuis le début de ce chapitre, va donc être représenté par 36 coefficients qui vont entièrement définir son comportement et peuvent être mesurés sur des matériaux réels par des séries de tests, qui sont dits *rhéologiques*.

3.2 La loi de Hooke

En pratique, la loi linéaire introduite précédemment est souvent trop générale et peut être simplifiée. Si l'on considère que la matériau est *isotrope*, c'est-à-dire qu'il a le même comportement dans toutes les directions (c'est très souvent le cas, sauf pour des matériaux fibreux à direction privilégiée, comme le bois ou les muscles),

[4] K est en fait un tenseur d'ordre 4

alors des raisons de symétrie font que parmi les 36 coefficients, deux seulement sont indépendants. La loi de comportement se simplifie alors grandement et peut s'écrire :

$$\sigma \;=\; \lambda\,\mathrm{tr}\,(\varepsilon) I_3 \;+\; 2\,\mu\,\varepsilon \tag{2.12}$$

où I_3 est la matrice identité et tr la trace de la matrice. λ et μ sont les deux coefficients indépendants et s'appellent les *constantes de Lamé*. Elles sont homogènes à des pressions.

Des tests rhéologiques permettent de déterminer ces constantes, spécifiques à un matériau donné. Ces tests mesurent en fait directement deux autres valeurs : le module d'Young E (homogène à une pression) et le coefficient de Poisson ν (sans unité). Ces valeurs sont liées aux coefficients de Lamé par les équations :

$$\lambda \;=\; \frac{E\nu}{(1-2\nu)(1+\nu)} \quad , \quad \mu = \frac{E}{2(1+\nu)} \tag{2.13}$$

E correspond intuitivement à la rigidité du matériau et varie entre 10^5 et 10^{10}. ν mesure son incompressibilité et varie entre 0.0 (aucune préservation de volume, valeur théorique, en pratique ν est supérieur à 0.25) et 0.5 (matériaux parfaitement incompressibles).

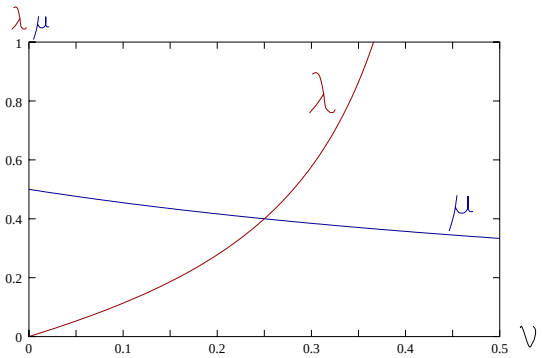


FIG. 2.8: Évolution de λ et μ lorsque ν varie entre 0 et 0.5, pour un E donné. Noter que λ tend vers l'infini pour des matériaux incompressibles ($\nu = 0.5$).

3.3 Équation de Navier

On se place ici dans le cadre des petits déplacements et des petites vitesses, de sorte que l'on peut linéariser les équations. Cela revient pour nous à considérer que le tenseur de Green-Lagrange peut être assimilé à celui de Cauchy, comme on l'a vu précédemment. On a donc (Eq. 2.4) :

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{2.14}$$

Vitesse et accélération sont liés par :

$$v_i = \frac{\partial u_i}{\partial t} \quad , \quad a_i = \frac{\partial v_i}{\partial t} \tag{2.15}$$

La loi de conservation de la masse s'écrit :

$$\frac{\partial \rho}{\partial t} + \mathrm{div}\,(\rho \mathbf{v}) = 0 \tag{2.16}$$

Celle de la conservation du moment, en omettant l'action des forces extérieures (Eq. 2.11) :

$$\rho \mathbf{a} = \mathbf{div}\,\sigma \tag{2.17}$$

Et enfin la loi de comportement de Hooke s'écrit (Eq. 2.12) :

$$\sigma \;=\; \lambda\,\mathrm{tr}\,(\varepsilon) I_3 \;+\; 2\,\mu\,\varepsilon \tag{2.18}$$

L'ensemble de ces équations décrit la théorie de l'élasticité. On a en tout 22 équations (6 + 6 + 1 + 3 + 6) pour les 22 variables du système en chaque point : $\rho, \mathbf{u}, \mathbf{v}, \mathbf{a}, \varepsilon$ et σ. En substituant les termes en σ_{ij} dans ces équations, on obtient l'*équation de Navier* :

$$\boxed{\rho\,\mathbf{a} = \mu\,\mathbf{div}\,(\mathbf{grad}\,\mathbf{u}) \;+\; (\mu+\lambda)\,\mathbf{grad}(\mathrm{div}\,\mathbf{u})} \tag{2.19}$$

3.4 Interprétation

L'équation de Navier, qui, rappelons-le, n'est valable que dans l'hypothèse des très faibles déplacements a une interprétation intéressante. L'accélération est tout d'abord proportionnelle à l'inverse de la densité, ce qui se comprend intuitivement, car plus le matériau est lourd, moins une contrainte donnée parviendra à le déformer. L'accélération est ensuite la somme de deux termes, qui ont chacun une interprétation physique simple :

- Le premier terme, $\rho\,\mathbf{a} = \mu\,\mathbf{div}(\mathbf{grad}\,\mathbf{u}) = \mu\,\Delta\mathbf{u}$, est une équation hyperbolique qui simule la *propagation d'une onde* à l'intérieur du matériau. Une déformation provoquée à un endroit va se propager dans toutes les directions, à une vitesse $\sqrt{\frac{\mu}{\rho}}$.
- Le second terme, $\rho\,\mathbf{a} = (\mu+\lambda)\,\mathbf{grad}(\mathrm{div}\,\mathbf{u})$, simule une *préservation du volume*. div $\mathbf{u}$ mesure l'*expansion volumique*, et une accélération dirigée selon le gradient de cette valeur va chercher à déplacer le point de telle sorte que la densité soit localement préservée.

Les constantes de Lamé viennent pondérer ces deux comportements et ainsi créer des comportements plus complexes d'onde de déformation avec plus ou moins de compressibilité. À noter que lorsque le coefficient de Poisson, ν, est proche de 0.5, ce qui correspond à un matériau parfaitement incompressible, le second terme (préservation du volume) devient prépondérant, le coefficient λ augmentant jusqu'à devenir infini pour $\nu = 0.5$. En pratique on considérera le matériau comme raisonnablement incompressible dès lors que $\lambda > 100\mu$.

Cette interprétation du comportement, exacte si l'on utilise le tenseur de Cauchy, devient trop réductrice si l'on utilise le tenseur de Green-Lagrange. Néanmoins, en tant qu'approximation au premier ordre de celui-ci, ce comportement se retrouvera avec Green-Lagrange, même si les termes non linéaires feront apparaître des réactions plus complexes.

4 Frottements

Les équations précédentes ne font pas intervenir les dérivées temporelles des déformations. Le modèle de *Kelvin-Voigt* présenté ici va les utiliser très naturellement, permettant ainsi de modéliser les forces de dissipation, aussi appelées par analogie aux fluides, forces visqueuses. La démarche mathématique est élégamment exactement la même que celle présentée plus haut et nous ne la détaillerons donc pas trop.

4.1 Le tenseur des taux de déformation

Ce tenseur, noté κ, va mesurer la vitesse à laquelle les déformations se produisent. Il est tout simplement défini comme le tenseur composé des dérivées temporelles des termes du tenseur des déformations, $\kappa = \frac{\partial\varepsilon}{\partial t}$. Ses termes s'expriment donc ainsi :

- Pour le tenseur de Cauchy (voir Eq. 2.4),

$$\kappa_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \tag{2.20}$$

- Pour le tenseur de Green-Lagrange (voir Eq. 2.6),

$$\kappa_{ij} = \left(\frac{\partial \mathbf{v}}{\partial \Omega_i}\cdot\frac{\partial \mathbf{p}}{\partial \Omega_j}\right) + \left(\frac{\partial \mathbf{p}}{\partial \Omega_i}\cdot\frac{\partial \mathbf{v}}{\partial \Omega_j}\right) \tag{2.21}$$

Rien n'interdit d'utiliser un tenseur des taux de déformations de Green-Lagrange avec un tenseur des déformations de Cauchy, et inversement, mais le comportement différent de ces deux formulations risque d'entraîner un comportement chaotique.

Puisque le tenseur des taux de déformations s'écrit de manière similaire à celui des déformations, il hérite de ses propriétés. En particulier, une translation rigide de l'objet, qui n'affecte pas ε, ne sera pas prise en compte dans κ. Si l'on utilise le tenseur dérivé de Green-Lagrange, il en sera de même des rotations globales et donc de tout mouvement rigide. Le tenseur des taux de déformations ne mesurera et n'atténuera donc que les vibrations *internes* au matériau.

4.2 Le tenseur des contraintes

Le tenseur des taux de déformations va lui aussi engendrer des contraintes à l'intérieur du matériau, elles aussi modélisées par un tenseur de taille 3. Pour différencier ces deux influences, nous notons ici $\sigma^{(\varepsilon)}$ les

contraintes dues aux déformations (décrites précédemment) et $\sigma^{(\kappa)}$ celles dues aux forces visqueuses. Les contraintes vraiment exercées sur un point seront tout simplement la somme de ces deux influences $\sigma = \sigma^{(\varepsilon)} + \sigma^{(\kappa)}$.

4.3 La loi de comportement

Les considérations d'isotropie évoquées plus haut font que la loi de comportement qui relie κ et $\sigma^{(\kappa)}$ se définit, tout comme la loi de Hooke (Eq. 2.12), par deux variables indépendantes :

$$\sigma^{(\kappa)} = \phi \, \mathrm{tr}\,(\varepsilon) I_3 + 2\,\psi\,\varepsilon \tag{2.22}$$

Les constantes ϕ et ψ sont encore une fois dépendantes du matériau simulé. Leur interprétation découle de celle faite de l'équation de Navier, pour de faibles déformations. ϕ va atténuer la propagation des ondes dans le matériau, alors que $\phi + \psi$ influencera la préservation du volume.

5 Conclusion

Ce chapitre a présenté les notions élémentaires de physique des milieux continus qui vont être utilisées dans les chapitres suivants. Les formules utilisées par la suite ne reprendront que les résultats finaux de cette modélisation, qui donnent l'expression de la force appliquée en fonction des déformations subies. Nous avons néanmoins tenu, même si nous nous sommes restreint à leur expression la plus simple, à présenter les différents tenseurs qui interviennent dans cette modélisation, pour ainsi introduire les hypothèses qui sont faites sur la classe de déformations et de matériaux qu'ils permettent de simuler.

Les équations impliquées demandent toutes le calcul de dérivées partielles, par rapport à un repère fixe ou à celui lié au matériau. Les approches présentées par la suite ont principalement pour but de trouver l'expression de ces dérivées, dans un matériau qui n'est plus continu comme on l'a supposé ici, mais qui a été *discrétisé* en un ensemble finis de points d'échantillonnage. Une fois ces formules établies et après avoir vérifié leur indépendance à la discrétisation, il restera à mettre en place le processus de multirésolution qui tirera parti du caractère intrinsèque des modèles décrits ici, qui fait que le comportement du matériau ne dépendra pas, dans une certaine mesure, de la résolution utilisée.

CHAPITRE 3

Premier modèle multirésolution

CALCULER le mouvement d'un objet à plusieurs résolutions est indispensable pour optimiser les calculs et accélérer la simulation. Nous avons décrit dans le chapitre précédent la théorie de l'élasticité qui permet de s'abstraire, au moins partiellement, de la résolution utilisée.

Ce chapitre va présenter un premier modèle d'algorithme multirésolution mettant en œuvre les équations de la mécanique des milieux continus vues dans le précédent chapitre. Il a fait l'objet d'une publication au 10 *ème* Workshop Eurographics sur l'animation et la simulation [DDBC99].

Plus précisément, nous sommes ici partis de l'équation de Navier, qui donne directement l'accélération subie par un point en fonction des déplacements des points voisins (Eq. 2.19) :

$$\rho\ \mathbf{a} = \mu\ \mathbf{div}(\mathbf{grad}\ \mathbf{u})\ +\ (\mu+\lambda)\ \mathbf{grad}(\mathrm{div}\ \mathbf{u}) \tag{3.1}$$

Le chapitre précédent a détaillé les hypothèses faites et l'interprétation de cette équation (voir Section 3.3). L'algorithme que nous allons développer (sans prendre en compte dans un premier temps l'aspect multirésolution) a la simplicité de ceux des modèles de particules.

Partant d'une discrétisation de l'objet, à chaque pas de temps on va :
- calculer en chaque point **div** (**grad u**) et **grad**(div **u**) ;
- en déduire l'accélération en utilisant l'équation de Navier ;
- intégrer cette accélération et les éventuelles forces extérieures (gravité) ;
- en déduire les nouvelles positions des points et donc leurs déplacements respectifs ;
- et ainsi de suite...

La seule difficulté réside dans le calcul des deux opérateurs différentiels. Des méthodes comme les *différences finies* permettent de calculer, grâce à des développements de Taylor, l'expression de dérivées partielles arbitraires. Elles nécessitent par contre de connaître la valeur du champ dont on veut calculer la différentielle en des points *équirépartis* sur une *grille régulière*. On veut ici pouvoir discrétiser un objet de topologie arbitraire. L'introduction de nouveaux points pour la multirésolution va de plus compliquer ensuite la discrétisation, pouvant amener à une grille quelconque, ce qui fait que cette méthode est mal adaptée.

C'est la raison pour laquelle nous avons cherché à définir de nouveaux opérateurs différentiels capables d'approximer les différentielles sur des points d'échantillonnage arbitrairement répartis. Le laplacien **div** (**grad u**)

est le premier des deux termes que nous avons calculés. Son expression a ensuite été reprise pour formuler **grad**(div **u**).

Une fois ces opérateurs exprimés, nous pourrons les utiliser dans un algorithme multirésolution, profitant de ce que l'échantillonnage puisse être arbitraire. Nous détaillerons la façon dont nous adaptons la résolution locale au cours de la simulation, élaborant ainsi l'une des premières méthodes de simulation multirésolution pour un matériau élastique.

1 Calcul du laplacien

Le laplacien $\Delta\mathbf{u}$ du champ **u**, qui est égal au **div** (**grad u**) est pour chacune de ses composantes la somme des dérivées secondes dans les trois directions :

$$(\Delta\mathbf{u})_i = u_{i,xx} + u_{i,yy} + u_{i,zz} \tag{3.2}$$

1.1 Dérivée seconde scalaire

Le but est donc ici de savoir calculer un des ces termes $u_{i,jj} = \frac{\partial^2 u_i}{\partial x_j^2}$, dérivée seconde du champ scalaire u_i dans une direction j quelconque.

S'inspirant des différences finies, on peut exprimer classiquement la dérivée seconde de f au point p comme :

$$f''(p) = \frac{f(p+h) \ - \ 2f(p) \ + \ f(p-h)}{h^2}$$

où h est la distance entre p et ses deux voisins situés en $p+h$ et $p-h$. Cette expression est valide à l'ordre 2, les termes négligés étant de l'ordre de $O(h^2)$.

Milne avait dans sa thèse [Mil95] montré combien la dérivée était sensible au bruit dès lors que l'on n'a plus affaire à une dérivée centrée. Fornberg [For88] avait proposé une extension de la formule précédente pour traiter le cas où les deux voisins de p ne sont plus situés à la même distance h, mais à des distances δ et ε. Son calcul revient à chercher la courbure du polynôme de degré deux passant par les trois points. La dérivée s'exprime alors par :

$$f''(p) = \frac{2}{\delta+\varepsilon}\left(\frac{f(p+\delta)-f(p)}{\delta} + \frac{f(p+\varepsilon)-f(p)}{\varepsilon}\right)$$

On notera que dans le cas où $\delta = \varepsilon = h$ on retrouve l'expression précédente, ce qui est normal puisque toutes deux s'obtiennent en regroupant les développements de Taylor de f faits en p avec chacun de ses voisins. Si les points ne sont pas à la même distance, on n'obtient par contre plus qu'une formule valable à l'ordre 1, les termes négligés étant de l'ordre de $O(\delta,\varepsilon)$.

1.2 Passage en trois dimensions

L'expression du laplacien que nous allons utiliser en 3D est une simple extrapolation de la formule précédente. En lieu et place des deux voisins précédents, nous avons maintenant n voisins répartis autour du point. Nous ne calculons plus la dérivée seconde le long d'un axe donné (celui sur lequel sont les points dans l'exemple 1D précédent), mais la somme de toutes ces dérivées secondes dans *toutes* les directions. On obtient donc en un point i une approximation du laplacien, somme de ces dérivées secondes :

$$(\Delta f)^i = \frac{2}{\sum\limits_{\text{voisins } j} l^{ij}} \sum_{\text{voisins } j} \frac{f^j - f^i}{l^{ij}} \tag{3.3}$$

f^j est la valeur du champ (scalaire) au voisin j et l^{ij} la distance avec ce voisin.

Cette expression a déjà été utilisée par Fujiwara dans le cadre de l'analyse d'image [Fuj95] et fut reprise par Desbrun *et al.* pour le lissage de maillages [DMSB99].

Il est à noter qu'on retrouve exactement l'expression des différences finies dès lors que l'on se place sur une grille régulière. On peut donc voir cette formule, avec ses coefficients dépendant des distances aux voisins

comme une extension de différences finies, les poids associés à la valeur f^j de la fonction dépendant de la configuration dans laquelle on se trouve.

Cette formulation peut également être vue comme un filtrage par analogie au noyau de filtrage W_h utilisé dans les SPH (voir Sec. 5, Chap. 1). L'importance d'un point (de la *valeur* de la fonction en ce point) est en effet inversement proportionnelle à la distance qui le sépare du point considéré. On tiendra donc davantage compte des voisins proches ce qui est naturel puisque c'est leur valeur qui influencera le plus l'aspect local de la fonction, et donc le laplacien.

1.3 Laplacien d'un champ vectoriel

On cherche le laplacien *vectoriel* du champ déplacement $\mathbf{u}$[1]. Puisque chacune de ses composantes est le laplacien de la composante associée de $\mathbf{u}$, on a simplement :

$$\boxed{(\Delta\mathbf{u})^i = \frac{2}{\sum\limits_{\text{voisins j}} l^{ij}} \sum_{\text{voisins } j} \frac{\mathbf{u}^j - \mathbf{u}^i}{l^{ij}}} \tag{3.4}$$

pour un point i donné, en fonction de ses voisins j situés à la distance l^{ij}.

2 Extension au grad (div)

Le $\mathbf{grad}(\text{div}\,\mathbf{u})$ que nous cherchons à calculer s'écrit, pour chacune de ses composantes :

$$[\mathbf{grad}(\text{div}\,\mathbf{u})]_i = u_{x,xi} + u_{y,yi} + u_{z,zi} \tag{3.5}$$

Nous allons, pour calculer cet opérateur, extrapoler l'Équation 3.4 du laplacien, les deux formulations étant donc développées dans un cadre unifié. Le $\mathbf{grad}(\text{div}\,\mathbf{u})$ est en effet lié au laplacien précédemment calculé par l'équation :

$$\Delta\mathbf{u} = \mathbf{grad}(\text{div}\,\mathbf{u}) \;-\; \mathbf{rot}(\mathbf{rot}\,\mathbf{u}) \tag{3.6}$$

Il faut donc, dans l'expression du laplacien, simplement parvenir à différentier les deux termes de cette somme pour isoler la contribution de $\mathbf{grad}(\text{div}\,\mathbf{u})$. Cela se fait en remarquant que le double rotationnel va précisément mesurer la rotation du champ $\mathbf{u}$ et que celle-ci peut être extraite du vecteur déplacement. Celui-ci peut en effet être considéré comme la somme d'une composante purement radiale due aux forces de pression et d'une composante tangentielle indiquant localement la rotation du champ $\mathbf{u}$ (voir Fig. 3.1).

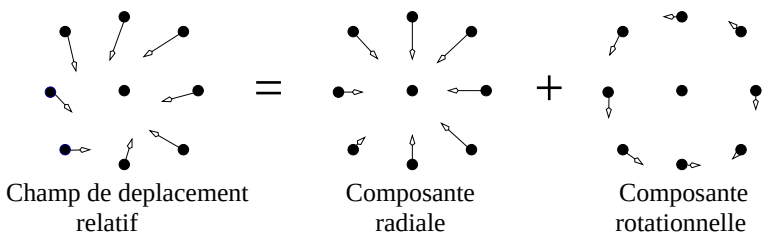


FIG. 3.1: Le champ de déplacement peut être séparé en deux composantes : la radiale, créée par les forces de pression, et la rotationnelle, créée par les forces de cisaillement.

Ce ne sont pas directement les valeurs $\mathbf{u}^j$ du déplacement des voisins qui nous intéressent, mais plutôt leur *déplacement relatif*, $\mathbf{u}^j - \mathbf{u}^i$, par rapport au point i où l'on cherche à calculer la différentielle. On se place en quelque sorte ici dans un repère lié au point où l'on fait les calculs et l'on observe comment se sont déplacés les voisins en supposant notre position fixe.

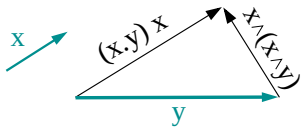


FIG. 3.2: **y** s'écrit comme la somme de deux vecteurs orthogonaux en fonction du vecteur unitaire **x**.

2.1 Projection le long de l'axe

Comme on le voit sur la Figure 3.2, un vecteur **y** peut s'écrire un utilisant en vecteur unitaire **x**, comme la somme d'un vecteur aligné avec **x** et d'un vecteur orthogonal à **x** grâce à la relation : $\mathbf{y} = (\mathbf{x}\cdot\mathbf{y})\mathbf{x} - \mathbf{x}\wedge(\mathbf{x}\wedge\mathbf{y})$, $\wedge$ désignant le produit vectoriel et $\cdot$ le produit scalaire.

On notera $\mathbf{l}^{ij}$ le vecteur reliant le point i à son voisin j et l^{ij} sa norme. Nous pouvons alors réécrire en la décomposant l'équation du laplacien 3.4 en prenant pour **x** le vecteur unitaire $\mathbf{x} = \mathbf{l}^{ij}/l^{ij}$ et pour **y** le vecteur déplacement relatif $\mathbf{u}^j - \mathbf{u}^i$:

$$\begin{aligned}
(\Delta\mathbf{u})^i &= \frac{2}{\sum\limits_{\text{voisins j}} l^{ij}} \sum_{\text{voisins } j} \frac{\mathbf{u}^j - \mathbf{u}^i}{l^{ij}} \\
&= \frac{2}{\sum\limits_{\text{voisins j}} l^{ij}} \sum_{\text{voisins } j} \frac{\left[(\mathbf{u}^j - \mathbf{u}^i)\cdot\frac{\mathbf{l}^{ij}}{l^{ij}}\right]\frac{\mathbf{l}^{ij}}{l^{ij}} - \frac{\mathbf{l}^{ij}}{l^{ij}}\wedge\left[\frac{\mathbf{l}^{ij}}{l^{ij}}\wedge(\mathbf{u}^j - \mathbf{u}^i)\right]}{l^{ij}} \\
&= \underbrace{\frac{2}{\sum\limits_{\text{voisins j}} l^{ij}} \sum_{\text{voisins } j} \frac{\left[(\mathbf{u}^j - \mathbf{u}^i)\cdot\frac{\mathbf{l}^{ij}}{l^{ij}}\right]\frac{\mathbf{l}^{ij}}{l^{ij}}}{l^{ij}}}_{\textit{Composante radiale}} - \underbrace{\frac{2}{\sum\limits_{\text{voisins j}} l^{ij}} \sum_{\text{voisins } j} \frac{\frac{\mathbf{l}^{ij}}{l^{ij}}\wedge\left(\frac{\mathbf{l}^{ij}}{l^{ij}}\wedge(\mathbf{u}^j - \mathbf{u}^i)\right)}{l^{ij}}}_{\textit{Composante rotationnelle}}
\end{aligned} \tag{3.7}$$

En comparant cette équation avec 3.7, nous choisissons d'identifier la composante radiale comme étant le gradient de la divergence :

$$\boxed{[\mathbf{grad}(\text{div}\,\mathbf{u})]^i = \frac{2}{\sum\limits_{\text{voisins j}} l^{ij}} \sum_{\text{voisins } j} \frac{\left[(\mathbf{u}^j - \mathbf{u}^i)\cdot\frac{\mathbf{l}^{ij}}{l^{ij}}\right]\frac{\mathbf{l}^{ij}}{l^{ij}}}{l^{ij}}} \tag{3.8}$$

Cette extrapolation à partir de la formule du laplacien possède une autre justification, plus intuitive. Comme il l'a été dit dans la discussion de l'équation de Navier chapitre 2, le terme $\mathbf{grad}(\text{div}\,\mathbf{u})$ va chercher à préserver le volume du matériau. Localement, la variation de ce volume peut-être représentée comme celle du volume entourant un point, défini par ses plus proches voisins. Il est donc normal de ne considérer que le déplacement *radial* des voisins, car c'est cette composante uniquement qui affecte le volume.

Nous reviendrons à la fin de ce chapitre sur les inconvénients de cette assimilation.

3 Simulation à résolution fixe

3.1 Cas d'école

Nous avons maintenant grâce aux Équations 3.4 et 3.8 le moyen de calculer l'accélération de tout point pour appliquer l'algorithme décrit au début de ce chapitre. Cette accélération est une fonction du champ de déplacement, échantillonné sur ses points voisins. Nous allons mesurer la qualité de ces opérateurs différentiels en simulant un ensemble de points. L'exemple choisi est celui d'un cube de matière uniforme, dont l'une des faces est collée sur un mur. Partant de sa position au repos avec une vitesse nulle, nous allons simuler ses oscillations sous l'effet de la gravité.

Nous avons volontairement choisi de ne faire intervenir aucune force dissipative (frottement de l'air, viscosité) de manière à mettre en évidence le comportement du modèle sur un cas d'école.

[1] Voir Annexe A.

Les résultats visuels obtenus sont assez satisfaisants. Ils sont conformes au modèle de Cauchy présenté au Chapitre 2. Le cube se déforme sans trop plier, oscillant principalement verticalement. Augmenter la raideur du matériau crée des oscillations plus rapides et de plus faible amplitude.

La Figure 3.3 représente l'altitude de l'un des coins du cube en fonction du temps. Les différentes courbes ont été obtenues avec plus ou moins de points simulés correspondant à des maillages réguliers de respectivement $4^3 = 64$, $8^3 = 512$ et $16^3 = 4096$ particules.

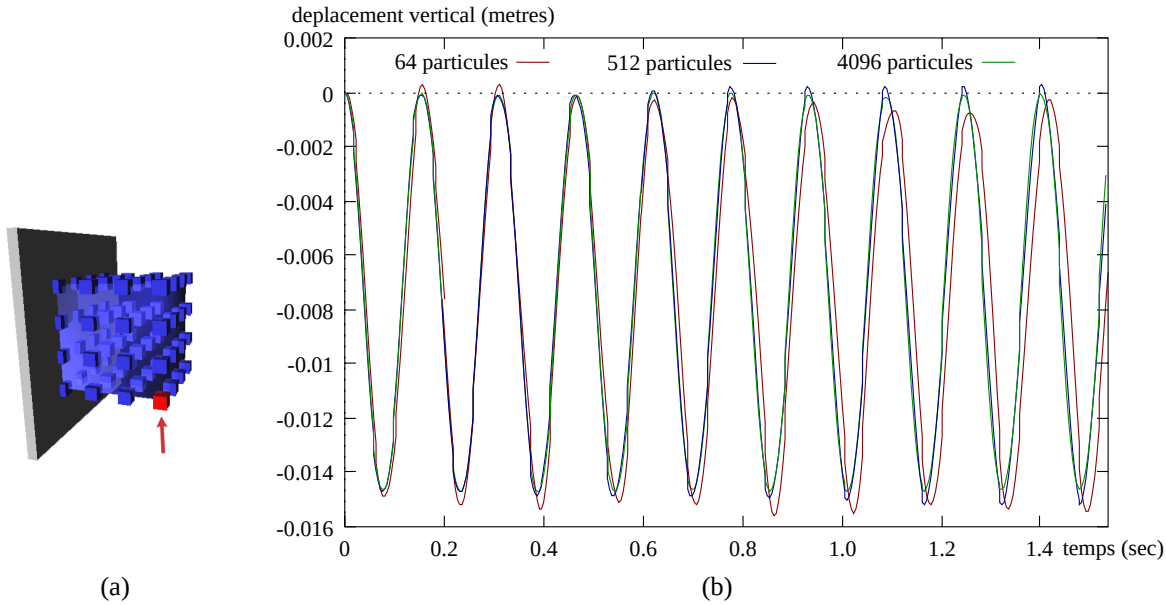


FIG. 3.3: L'altitude de l'un des coins du cube (indiqué par une flèche) lors de la simulation, pour différents maillages réguliers. Aucun frottement n'a été ajouté dans cette simulation.

L'absence de forces de dissipation explique ces oscillations continues et régulières non amorties. Peu de méthodes sont suffisamment "propres" pour permettre ce genre de simulation sans divergence numérique (on intègre avec un schéma d'Euler modifié[2]). L'existence d'une position de référence servant d'attracteur y est probablement pour beaucoup.

La forte ressemblance des courbes obtenues confirme l'indépendance à la discrétisation obtenue grâce à l'utilisation de l'équation de Navier. Les opérateurs différentiels présentés fournissent bien un calcul relativement indépendant de la résolution. Cette propriété est un élément essentiel pour pouvoir passer à une méthode mélangeant à chaque instant les différentes résolutions.

3.2 Ajout de forces dissipatives

Le réalisme de la simulation peut être amélioré par l'ajout de forces dissipatives. Ceci aura pour effet de stabiliser plus ou moins rapidement les oscillations de l'objet, se rapprochant ainsi des matériaux réels où ces forces apparaissent naturellement. Ceci peut se faire à l'aide d'une force de frottement visqueuse classique, inversement proportionnelle à la vitesse :

$$\mathbf{F}_d = -k_d.\mathbf{v}$$

Nous avons également ajouté une force de viscosité dans notre modèle en modélisant les interactions des particules entre elles. Cette force va faire en sorte que chaque particule soit affectée par le mouvement de ses voisines et tente de suivre leur mouvement global, ajoutant ainsi une cohérence interne au matériau. Inspirée par la formulation SPH de [Mon92, DCG96], nous exprimons cette force comme :

$$\mathbf{F}_v^i = \frac{k_v}{\sum_{\text{voisins } j} m_j} \sum_{\text{voisins } j} m_j(\mathbf{v}_j - \mathbf{v}_i) \tag{3.9}$$

[2]Voir Annexe B.

On peut voir cette équation comme une extension de celle utilisée dans les cas d'échantillonnage régulier de l'espace. L'influence d'un voisin a été choisie comme proportionnelle à sa masse.

Cette force est filtrée pour n'en conserver que son aspect dissipateur et on l'annule si elle accélère la particule ou lui fait changer la direction de sa vitesse.

Bien que n'offrant pas de garantie d'un comportement indépendant de la résolution, cette formulation inspirée de [Mon92] ne crée pas d'artefact trop visible. Nous utiliserons des coefficients assez faibles dans nos exemples, principalement pour obtenir un matériau plus rigide.

Nous aurions pu utiliser le formalisme d'élasticité linéaire décrit en Section 4 du chapitre précédent. Cela n'a pas été fait car au moment du développement de cette méthode, nous n'avions pas développé les formules qui y sont présentées, qui sont une extrapolation de celles trouvées dans d'autres articles de la littérature [OH99] au cas du tenseur infinitésimal de Cauchy.

4 Simulation multirésolution

La méthode multirésolution présentée ici va être *adaptative* selon la définition faite en Section 10 du chapitre 1. Nous allons précalculer plusieurs maillages volumiques, définir comment une zone de l'espace peut être simulée par l'un ou l'autre de ces maillages et comment elle interagit alors avec le reste de l'objet pour enfin voir comment et sur quels critères imposer un changement de niveau de détail.

4.1 Création des maillages

Les maillages que nous allons créer seront issus d'une division de l'espace en cubes. Le cube est en effet la seule figure géométrique *régulière* pouvant remplir l'espace. On peut mailler un volume à l'aide de tétraèdres, mais ceux-ci ne pourront avoir tous la même forme. La découpe d'un tétraèdre en de nouveaux tétraèdres plus petits dégrade par ailleurs largement leur qualité, plusieurs applications du processus conduisant à des tétraèdres très irréguliers. On garantit en utilisant des cubes une répartition la plus régulière possible des points d'échantillonnage.

Le maillage dont on va partir est régulier, aligné avec les axes et de même résolution dans les trois directions x, y et z. On ne considère que les nœuds de ce maillage qui se trouvent *à l'intérieur* de l'objet [3]. Ce seront les particules que nous allons simuler. Leur masse est fixée comme étant celle de l'objet divisée par le nombre de particules (si l'objet est supposé de densité uniforme, les points d'échantillonnage étant uniformément répartis).

On construit ensuite récursivement les différents maillages en partant de celui-ci, qui sera le plus fin. Chaque maillage se déduit du précédent en regroupant chaque bloc de 8 particules en une nouvelle particule, celles-ci formant le nouveau maillage (voir Fig. 3.4).

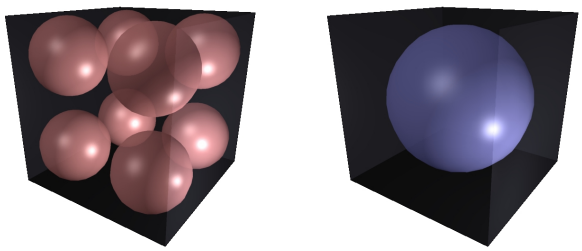


FIG. 3.4: Un même cube sera échantillonné par huit particules filles ou par leur mère.

Une telle particule échantillonnera ainsi un volume de taille huit fois moindre que celui échantillonné par les particules précédentes. On dénotera par les termes de *mère* et *filles* les particules ainsi liées. On va également implicitement classer ces maillages du bas vers le haut, en allant du plus fin vers le plus grossier. Une fille appartiendra ainsi au niveau *inférieur* de celui de sa mère.

Lorsqu'on construit un maillage à partir du précédent, on peut regrouper moins de 8 particules ensemble si on se trouve au bord (voir Fig. 3.5). Dans tous les cas, et pour conserver la masse totale et sa répartition,

[3] On pourrait choisir de déplacer les points situés près du bord pour les amener sur la surface, mais cette optimisation n'est pas nécessaire. Les maillages, même s'ils doivent être proches de la géométrie de l'objet, seront en effet décorrélés de la surface qui va être affichée (voir Chapitre 6), et seul leur mouvement global nous importe.

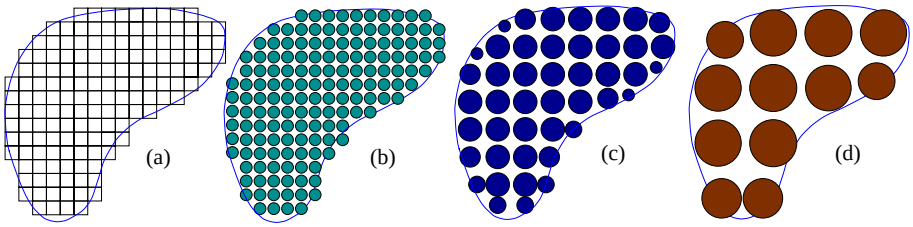


FIG. 3.5: Création récursive des maillages. La grille régulière qui sert à créer le maillage le plus fin (a). On regroupe ensuite les particules pour créer les maillages suivants (b), (c) et (d). Les tailles représentent la masse des particules.

la nouvelle particule aura pour masse la somme de celles de ses filles et une position définie comme leur barycentre pondéré par les masses :

$$m = \sum_{filles\ i} m_i \qquad \mathbf{p} = \frac{\sum_{filles\ i} m_i\, \mathbf{p_i}}{m} \tag{3.10}$$

où m est la masse, et $\mathbf{p}$ la position de la particule mère.

Les maillages ainsi créés ne seront donc plus exactement réguliers, mais la position de chacune des nouvelles particules est optimale au sens de la répartition de masse et l'on crée donc un maillage adapté à la morphologie de chaque objet.

En regroupant ainsi par huit les particules entre elles, on crée un *octree* topologique de particules. Ce n'est pas un octree au sens classique (des cases cubiques régulièrement divisées en huit autres cases) puisque les maillages ne sont plus réguliers et vont pouvoir se déformer au cours de l'animation, mais ça l'est au sens topologique de *voisinage*. Les relations de voisinages entre les particules sont quant à elles bien organisées dans une structure d'octree, des voisines pouvant être définies dans chaque direction (bien que n'existant pas forcément au bord).

4.2 Définition des voisines

Voisines de même niveau

Nous avons choisi qu'une particule se serve, pour calculer les opérateurs différentiels, de ses voisines immédiates. Ce sont les 26 particules (il peut y en avoir moins si on est au bord de l'objet) qui touchent le cube associé à la particule par une face (6), une arête (12) ou un coin (8), et qui sont représentées Figure 3.7 b.

Voisines appartenant aux autres maillages

Une particule pourra également, et c'est la clef des algorithmes multirésolution, prendre en compte des particules se trouvant à côté d'elle mais ne faisant pas partie du maillage de ce niveau. Nous avons choisi de limiter cette interaction entre deux niveaux de détail aux seuls maillages précédant et suivants : parmi les maillages successifs de l'objet, le maillage du niveau n ne pourra utiliser que les niveaux $n+1$ et $n-1$, s'ils existent (voir Fig. 3.6).

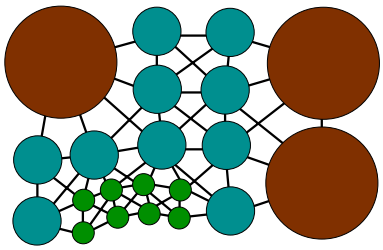


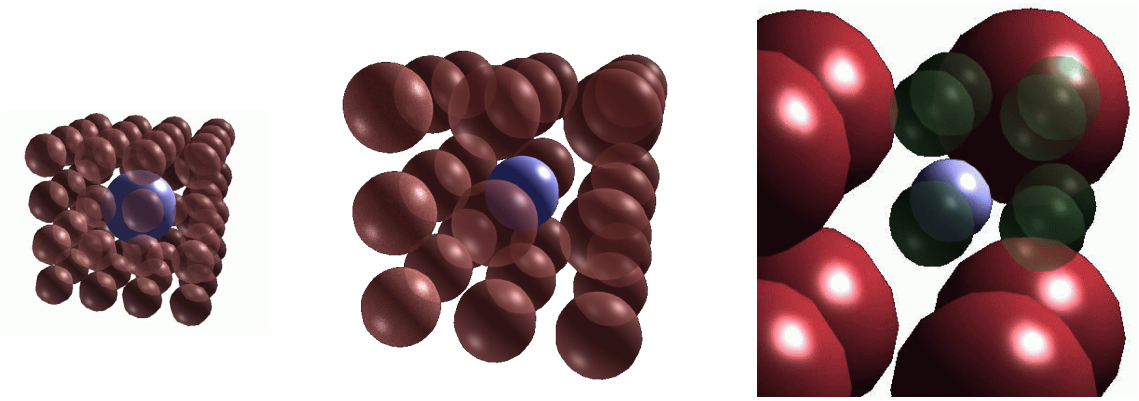
FIG. 3.6: Durant la simulation, les voisins d'une particule ne peuvent appartenir qu'au même niveau de résolution, au niveau supérieur ou au niveau inférieur.

Ce choix limite la souplesse de la méthode, mais créer un tel *octree restreint* (définition de [VB87]) a deux avantages :

– La structure de voisinage est limitée et plus simple. On va faire l'hypothèse que l'objet se déforme suffisamment peu pour que les voisines d'une particules ne changent pas au cours de la simulation. Elles pourront donc être définies et toutes stockées au début de la simulation et on évitera les coûteuses remises à jour de cette structure.
– Imposer des niveaux de résolution proches entre deux particules voisines va garantir que la discrétisation du matériau sera à tout instant continue. Il ne serait pas naturel que deux zones juxtaposées aient deux résolution trop différentes. Si on a besoin de beaucoup de précision à un endroit et de très peu juste à côté, il vaut mieux passer continuement d'une résolution à l'autre.

Il faudra donc garantir au cours de la simulation que chaque particule ne soit entourée que par des particules de niveau directement inférieur ou supérieur

En ayant ainsi limité les maillages parmi lesquels une particule va avoir des voisines, on peut stocker dans des listes toutes les voisines possibles, parmi lesquelles toutes ne seront pas forcément activées à un instant donné. Outre les 26 particules du même niveau déjà décrites, cette liste comporte 7 particules au niveau supérieur (les mères des précédentes) et 56 particules appartenant au niveau inférieur (les filles des voisines du même niveau qui jouxtent la particule considérée). Ce sont en fait toutes les particules qui touchent le cube associé à la particule (Fig. 3.7). Chaque particule aura donc ainsi 89 voisines potentielles.



(a) (b) (c)

FIG. 3.7: Les 56 voisines de niveau inférieur (a), les 26 voisines de même niveau (b) et les 7 voisines de niveau supérieur (et les "sœurs") (c) d'une particule donnée.

4.3 Simulation adaptative

À tout instant, chaque zone du matériau va être représentée par une et une seule particule. Cette particule aura une "taille" plus ou moins grosse (i.e. sera issue d'un maillage plus ou moins grossier) en fonction de la simulation. En pratique, un objet au repos sera représenté par les quelques particules du niveau le plus grossier que l'on aura créé. On appellera les particules servant ainsi effectivement à la simulation les particules *actives*. Toutes les particules des autres résolutions ne serviront pas à cet instant et seront qualifiées d'*inactives*.

S'il se passe quelque chose dans une région, une particule de cette région va pouvoir se *diviser* et être remplacée par ses 8 filles (encore une fois il peut ne pas y avoir 8 filles mais moins, si on considère une particule située au bord de l'objet), qui échantillonneront alors mieux la région. La particule mère qui s'est divisée devient *inactive*, ses filles accédant au statut d'*actives*. La simulation ne prendra plus alors en compte que les filles créées, qui cohabiteront avec les particules non divisées des zones voisines.

On peut avoir à diviser encore une ou plusieurs des filles créées, en les désactivant pour les remplacer également par leur filles, devenues alors actives, et ainsi de suite.

Inversement, lorsque cette zone est soumise à des contraintes moindres, les 8 filles peuvent se *regrouper* et laisser leur place à leur mère. Celle-ci redevient alors active et ses filles inactives. Ces différents processus de division et regroupement de particules devront néanmoins veiller à conserver la structure d'octree restreint de l'arbre. Ainsi, la nécessité de division d'une particule pourra entraîner d'autres divisions dans son voisinage.

La simulation ne va réellement utiliser à chaque instant que les particules actives. Celles-ci vont calculer des forces en fonction des déplacements de celles parmi leurs voisines qui sont également actives, forces qui seront intégrées comme précédemment. La différence avec l'algorithme à résolution fixe donné au début de ce chapitre est que des critères vont pouvoir à tout instant venir modifier la liste des particules actives, et donc les relations de voisinage entre particules, en effectuant des divisions et des regroupements.

4.4 Position relative mère-fille

Repère local

Lorsqu'une particule se divise, elle fait apparaître 8 filles qui vont la remplacer. La position de ces nouvelles particules doit être bien choisie. Si on se contente de placer les 8 filles dans la même position que celle qu'elles occupaient par rapport à leur mère au moment où l'on a construit les maillages, on va assister à des instabilités. En effet, le matériau étant déformé, les filles apparaîtront à des positions non déformées peu intuitives, source d'instabilités que l'on veut minimiser lorsque l'on change de résolution.

On va donc définir la position des filles par rapport à la mère dans un *repère local*, lié au matériau. Ce repère a pour origine la mère et pour axes trois directions définies comme suit. Pour chaque fille et pour chaque direction x, y ou z, il existe 4 particules issues du même maillage que la mère et formant la *face* la plus proche de la fille, orthogonale à la direction donnée. C'est le barycentre pondéré par les masses de ces faces qui définit la direction de l'axe.

La Figure 3.8 explicite cette définition en 2D. On n'a ici que deux axes, définis par deux faces. Les faces sont constituées des deux particules qui vont le mieux échantillonner la déformation dans chacune des directions. On calcule ensuite le barycentre de ces faces pour obtenir l'axe du repère.

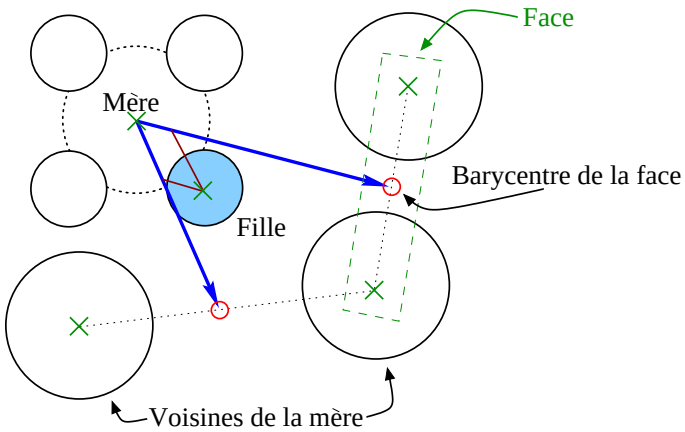


FIG. 3.8: Le système de coordonnées locales (flèches), défini par les voisines de la mère assure un bon placement des filles, même lorsque l'objet est déformé.

Les coordonnées de chacune des filles dans son repère local propre seront calculées dans la position non déformée et ne changeront pas. Ces axes, définis par rapport aux voisines de la mère, vont "suivre" la déformation du matériau et vont permettre de trouver une position intuitive pour les filles lorsqu'elles apparaissent.

Ces difficultés de positionnement n'apparaissent pas lorsque les filles se regroupent, la position de la mère se déduisant simplement grâce à une moyenne pondérée par les masses de celles de ses filles (Eq. 3.10).

Cas particuliers

La définition proposée n'est plus valable lorsqu'on l'applique aux particules du bord. Les faces que nous avons définies n'ont alors pas forcément 4 particules. Si elles en ont moins, c'est tout de même la seule source d'information que nous ayons sur les déformations dans cette direction et nous conservons la même définition, le barycentre des particules de la face comprenant simplement moins de 4 particules.

Il se peut quand on est tout au bord qu'il n'y ait *aucune* particule voisine de la mère pour composer une face dans une direction donnée. La meilleure approximation des déformations ayant eu lieu dans cette direction est alors donnée par la *face symétrique* de la face manquante par rapport à la mère.

Dans le cas ultime ou même cette face symétrique n'est pas définie on se rabat sur l'utilisation d'axes fixes dans les directions x, y ou z.

L'autre subtilité de la définition de ces axes est qu'au moment où la mère se divise, ses particules voisines qui définissent les faces ne sont pas forcément actives. Leur position n'a donc pas été mise à jour depuis le moment où elles se sont divisées[4].

On va dans ces cas là simplement mettre à jour leur position en fonction de celles de leurs filles par l'Équation 3.10. Cette remise à jour n'est pas très coûteuse, mais on pourrait décider ou non de la faire en fonction du temps écoulé depuis la division de la particule et donc de l'imprécision de sa dernière position. Il faut dans tous les cas savoir de quand date la dernière estimation de position pour éviter par exemple de faire ainsi plusieurs fois remonter la position depuis les filles lorsqu'une particule sert dans plusieurs des faces que l'on va utiliser, ce qui est souvent le cas.

4.5 Changement de résolution

Nous allons détailler les critères qui vont décider quelle résolution doit être appliquée en chaque zone de l'objet, ou plus précisément quelles sont les particules qui ont à se regrouper ou à se diviser à une étape donnée de la simulation. Ces choix sont faits en deux passes. La première fait une présélection des particules candidates (à la division ou au regroupement) et la deuxième filtre ces listes pour satisfaire le principe d'octree restreint.

Critère de continuité

On va décider de diviser (resp. regrouper) des particules en fonction de l'importance de la déformation dans la zone considérée. Ce n'est pas en fait de son amplitude, ni même de sa variation que l'on va se soucier, mais plutôt de sa dérivée seconde. Une amplitude constante voire même variant linéairement va être bien gérée par les opérateurs différentiels présentés plus haut car ils ont une précision du premier ordre. Les évolutions d'ordre plus élevé de la déformation, en commençant par la dérivée seconde, seront par contre mal approximées et il faudra pour mieux les gérer diviser les particules dans cette zone.

Il se trouve que le laplacien est justement une très bonne mesure du caractère non linéaire du champ déplacement puisqu'il mesure la somme des dérivées secondes principales. C'est sa norme, calculée en chaque point et multipliée par un terme correctif relatif à la résolution employée, qui va nous servir d'indicateur :

$$\overbrace{\varepsilon_{min} > h^2 \|\Delta \mathbf{u}\|}^{regroupement} \underbrace{> \varepsilon_{max}}_{division} \tag{3.11}$$

où h représente la plus petite distance entre cette particule et ses voisines. Le terme en h^2 vient pondérer la valeur du laplacien tolérable en fonction de la discrétisation locale.

Si elle est supérieure à un seuil ε_{max} donné, on va ajouter cette particule dans la liste de celles devant se diviser. Au contraire, si elle est inférieure à ε_{min} *pour les huit filles* d'une particule, celles-ci vont se placer dans la liste des particules à regrouper. Cette approximation rapide, basée sur la continuité, donne de bons résultats en pratique, le raffinement apparaissant où et quand nous l'espérions intuitivement.

L'écart entre les deux seuils ε_{min} et ε_{max} fait qu'il sera plus facile à une particule de se diviser qu'à ses filles de se regrouper. Cet effet *hystérésis* empêchera la zone d'osciller entre deux résolutions.

Critères liés à l'octree restreint

Les listes précédemment établies vont être filtrées pour garantir que la structure d'octree restreint soit conservée.

Nous passons d'abord en revue les particules désirant se diviser. Ce n'est possible que si toutes leurs voisines ont un niveau égal ou inférieur car on aurait sinon un écart de deux niveaux entre les filles créées et une particule voisine. Si ce n'est pas le cas, la particule qui désirait se diviser est supprimée de la liste, mais les voisines qui empêchaient sa division y sont ajoutées, afin que la division puisse se faire au pas d'après (ou à un pas suivant si la division des voisines a elle aussi demandé la division d'autres particules).

[4] Elles sont en effet forcément divisées si elles sont inactives pour satisfaire le critère d'octree restreint.

De façon similaire, il est possible qu'un regroupement pose problème. Pour des raisons de stabilité, la priorité est donnée aux particules subdivisées, même si elles empêchent un regroupement, et on se contente dans ce cas de supprimer ce regroupement de la liste.

Critères liés au respect du temps-réel

Pour respecter le temps-réel et assurer une fréquence d'affichage fixe, on peut être amené à ajouter d'autres conditions sur la division des particules. On va pouvoir *limiter* le coût de calcul entre deux affichages (qui varie linéairement avec le nombre de particules et leur pas de temps et qui peut donc être mesuré) en interdisant une division trop importante du modèle. On pourra à nouveau diviser pour améliorer la précision locale des calculs lorsque des regroupements auront eu lieu ailleurs dans l'objet.

Inversement, on va *ralentir* volontairement la simulation lorsque celle-ci est stable et pourrait aller plus vite que le temps-réel du fait du faible nombre de particules utilisées. Une simple boucle d'attente basée sur l'heure donnée par la machine permettra d'attendre la synchronisation avec l'affichage suivant.

4.6 Mise à jour de la structure

Les voisines qui vont réellement servir à une particule (leur déplacement servant à calculer la force) vont être celles parmi ses 89 voisines qui sont *actives* à ce moment là. Elles ne peuvent être toutes actives en même temps (elles sont liées par des relation mère-filles). Le nombre de particules actives est au maximum de 56 (si la particule est entourée de particules divisées (Fig. 3.7a) et au minimum de 14 (si tout est regroupé autour d'elle : 7 voisines du niveau supérieur et ses 7 "sœurs" (Fig. 3.7 c).

Les 89 voisines sont établies au départ et ne changeront pas durant la simulation. On mettra par contre à jour une liste dynamique de celles parmi elles qui sont actives (i.e. réellement simulées).

Lorsqu'une particule se divise (resp. regroupe), des tables précalculées permettent de mettre à jour la liste de ses voisines actives, ainsi que les listes des voisines actives de ses voisines. Toutes ces relations peuvent en effet être précalculées en fonction des position relatives des deux anciennes voisines. La mise à jour des listes de voisines actives ne demande ainsi aucun calcul.

5 Multirésolution temporelle

Tout comme la méthode propose une structure de multirésolution à l'aide d'échantillonnages différents de l'espace, nous allons introduire ici la notion de multirésolution temporelle. Le changement du pas de temps global d'intégration avait déjà été utilisé dans la littérature [Jou96, Arn88], et celui du changement du pas de temps individuel l'avait été par Desbrun [Des97], dont nous reprenons le principe ici.

Un pas de temps d'intégration différent pour chaque particule va permettre d'optimiser les calculs. Il permet de garantir en chaque endroit une stabilité numérique de l'intégration, tout en évitant de simuler à la fréquence la plus haute l'ensemble de l'objet.

L'algorithme de simulation est donc un peu plus complexe. Ce qui est décrit ici n'est pas lié au caractère de multirésolution spatiale décrit précédemment et pourrait être appliqué avec la méthode à résolution fixe de la Section 3.

5.1 Critère de Courant

Le critère de Courant (voir annexe B) impose un pas de temps d'intégration compatible avec la vitesse du son à l'intérieur du matériau. Ceci impose pour une particule donnée d'avoir un pas de temps satisfaisant :

$$dt < h \sqrt{\frac{\rho_0}{\lambda + 2\,\mu}}$$

La vitesse du son associée à l'équation de Navier est $\sqrt{\frac{\rho_0}{\lambda+2\,\mu}}$. Le paramètre h représente la plus petite distance entre une particule et l'une de ses voisines.

Puisque les résolutions d'échantillonnage sont divisées par deux entre un niveau et le suivant, la distance d'une particule à ses voisines grandit également d'un facteur deux. Le pas de temps d'un niveau grossier pourra donc être le double de celui du niveau fin qui le succède. En plus d'avoir environ huit fois moins de particules,

celles-ci doivent donc être simulées deux fois moins souvent, ce qui donne un rapport 16 entre les temps de simulation de deux niveaux successifs, si on les suppose intégralement actifs.

5.2 Synchronisation et mise en œuvre

Pour cette raison de facteur deux entre les pas de temps des particules des différents maillages, ainsi que pour des raisons de synchronisation, on n'utilisera en pratique qu'un nombre fini de pas de temps possibles, ceux-ci étant des inverses de puissances de deux du pas de temps d'affichage :

$$dt_i = \frac{dt_{\text{affichage}}}{2^i}$$

On classe les particules dans des listes en fonction de leur pas de temps d'intégration, choisi parmi les dt_i. Entre deux affichages successifs de la simulation, on va simuler de nombreux pas d'intégration. On divise le temps en intervalles de la taille du dt le plus faible, dt_{min}. Au $k^{ème}$ pas de la simulation, on parcourt les listes de particules correspondant à des dt_i sous-multiples du temps $t = k\ dt_{min}$. Le niveau dt_{min-1} sera ainsi effectivement simulé une fois sur deux, dt_{min-2}, une fois sur quatre et ainsi de suite. En pratique, choisir quels sont les dt_i qui doivent être simulés à chaque pas se fait très rapidement grâce à des opérations binaires sur le nombre de pas d'intégration déjà réalisés.

Pour chaque particule de la liste parcourue, on va calculer une force en fonction des derniers déplacements des voisines actives et ensuite l'intégrer pour trouver la nouvelle position. Cette intégration se fait sur un pas de temps correspondant à celui de la particule.

On pourrait aussi imaginer que toutes les intégrations se déroulent à la fréquence de dt_{min}. Les *forces* appliquées à toutes les particules actives seraient toujours calculées à la fréquence de leur dt_i, mais leur *position* serait remise à jour à chaque pas (tous les dt_{min}), en intégrant la *dernière force calculée*.

On aurait ainsi à chaque instant des positions actualisées pour toutes les particules actives, ce qui rendrait le calcul des forces plus précis car basé sur des positions plus récentes. Si on ne calcule pas plus souvent les forces, ce qui est l'opération chère, mettre à jour à chaque pas de temps la position de *toutes* les particules actives entraîne néanmoins un surcoût non négligeable qui ne vaut peut-être pas la peine d'être entrepris.

5.3 Critère de stabilité d'intégration

Le pas de temps déterminé par le critère de Courant est le plus grand tolérable pour une particule donnée. Nous lui avons ajouté un autre critère assurant que le pas de temps est suffisamment petit pour prévenir les problèmes d'intégration numérique. Ceci est obtenu en faisant en sorte que dt satisfasse :

$$||\mathbf{a}\ dt|| = ||\mathbf{v}_t - \mathbf{v}_{t-1}|| < \Delta\mathbf{v}_{max}$$

Ainsi nous réduisons dt lors des soudains changements de vitesse, sources d'instabilités.

Chaque particule adopte à chaque instant le plus grand pas de temps satisfaisant ces deux critères.

6 Résultats

6.1 Premiers essais

Les résultats de cet algorithme multirésolution sont assez intéressants. La simulation est très stable : même si l'on excède les limites de déformations réalistes, la surface commençant à se replier sur elle-même, il suffit de reculer l'outil pour que l'objet revienne très rapidement à sa position d'équilibre. Ce genre de comportement n'existe pas avec les réseaux masses-ressorts qui, une fois qu'ils ont été trop déformés, divergent inexorablement.

L'adaptativité du modèle en fonction des déformations subies est assez intuitive, comme le montre la Figure 3.9. Les différentes résolutions sont toutes employées, les plus fines étant adoptées à proximité de l'outil que l'on manipule, les régions lointaines se contentant de peu de particules.

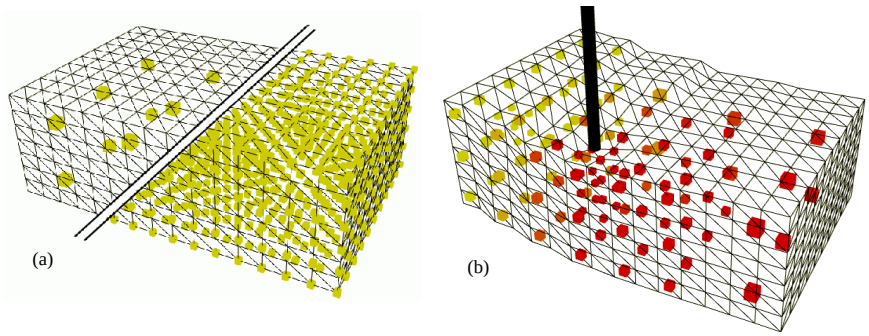


FIG. 3.9: Un parallélépipède déformé par un outil. Les résolutions comportent entre 24 et 1056 particules (a). La discrétisation qui a lieu au contact de l'outil durant la simulation est intuitive.

6.2 Cas d'école

Nous étudions maintenant l'animation d'une tige, dont une des faces est maintenue fixe, qui se plie sous l'effet de la gravité dans un milieu visqueux. La Figure 3.10 montre les états initiaux et finaux, avec des coefficients de Lamé de $\mu = 5000$ et $\lambda = 1000000$.

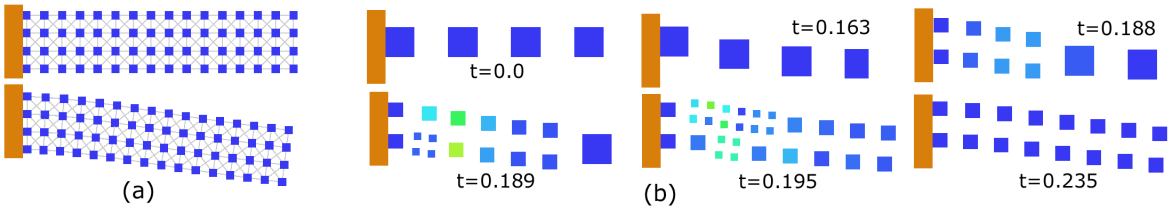


FIG. 3.10: Une tige oscillant sous la gravité. Simulation de référence faite avec 256 particules (a). Images de la simulation adaptative (b).

Le Tableau 6.2 montre le temps moyen de la simulation obtenu en utilisant différentes résolutions spatiales. Comme espéré, la simulation adaptative offre un bon compromis en donnant un temps de calcul proche de celui du niveau 3 (32 particules), tout en profitant des 256 particules potentielles du niveau 4. Le nombre de particules réellement simulées varie entre 4 (début) et 88, se stabilisant à 32 (niveau 3) quand la barre atteint sa position d'équilibre.

Niveau	2	3	4	adaptatif
Nb de particules	4	32	256	4-88
Temps de simulation (unités cpu)	0.87	4.29	38.90	5.27

6.3 Une application temps-réel

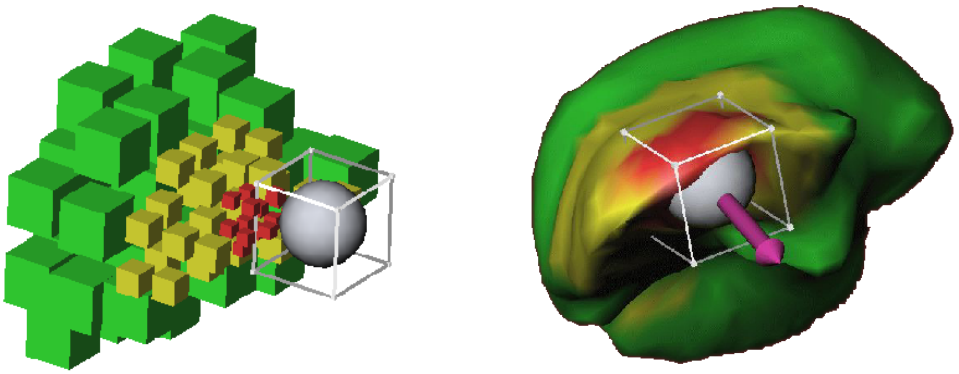
Nous avons réalisé avec cette technique un simulateur de chirurgie laparoscopique. Le foie virtuel, issu de données anatomiques, peut réagir en temps-réel aux actions de l'utilisateur, comme le montre la Figure 3.12.

Les coefficients rhéologiques λ et μ ont été choisis pour leur résultat visuel. Des données bio-médicales auraient dû être disponibles, mais les mesures faites ne sont pas suffisamment fiables car elle varient dans de grandes proportions en fonction de l'expérience et du caractère vivant ou mort de l'organe sur lequel elles sont faites.

Le caractère multirésolution de notre méthode est ici crucial car une simulation à discrétisation fixe utilisant seulement la résolution la plus fine aurait été trop lente d'un facteur dix par rapport au temps réel [5]. En combinant les résolutions, nous pouvons garantir que le temps-réel et la fréquence d'affichage seront respectés, celle-ci étant de 12 Hz dans notre cas sur une Silicon Graphics Onyx2 avec un processeur R10K. Le nombre de particules actives de la simulation varie entre 34 (résolution la plus grossière, utilisée lorsque le foie est au repos) et 130-140 qui est le nombre maximal de particules simulables en conservant le temps-réel avec cette raideur de matériau ($\mu = 4000$ et $\lambda = 5000$). La résolution des particules utilisées est intuitivement correcte :

[5] La simulation se faisant alors avec environ 770 particules.

élevée à proximité de l'outil et simplifiée à plus grande distance (voir Figure 3.11). Le pas de temps le plus faible utilisé était d'environ 0.002 sec (500 Hz) soit 32 pas d'intégration entre deux affichages.



a b

FIG. 3.11: Illustration de la répartition intuitive des particules lors d'une déformation. Le niveau des particules est représenté par la taille des cubes les symbolisant (a) ou par la couleur des nœuds de la surface auxquels elles sont reliées (b). La flèche représente la force renvoyée à l'utilisateur.

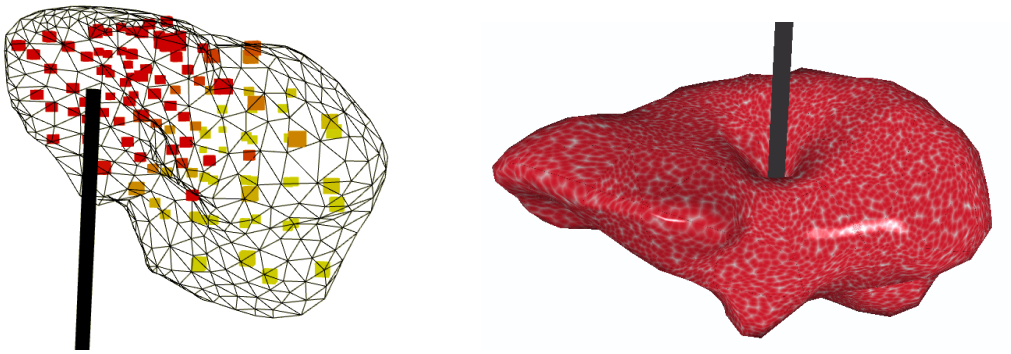


FIG. 3.12: Bien que les contraintes imposées dépassent parfois le formalisme des petites déformations, notre modèle présente néanmoins une réponse d'une bonne qualité visuelle à l'utilisateur. L'ajout de textures augmente sensiblement le réalisme du simulateur.

7 Discussion

7.1 Indépendance de la résolution

Les résultats obtenus sont assez convaincants et il est intéressant de noter que très peu de particules (quelques dizaines) suffisent souvent pour avoir un bon résultat.

Les simulations réalisées donnent un aspect un peu trop mou au foie et augmenter sa raideur se fait au détriment du nombre de particules que l'on peut simuler en conservant le temps-réel.

L'aspect multirésolution de l'animation est assez bien dissimulé à l'utilisateur (voir Chapitre 6), et même si les changements de niveau de détail internes peuvent se remarquer à la surface, le comportement global de l'objet reste cohérent.

Néanmoins des tests d'école réalisés sur l'exemple classique du cube collé à un mur et oscillant montrent que le comportement *dynamique* d'un objet où *se mélangent* les résolutions n'est pas le même que lorsque l'on simule les niveaux de détail séparément. La fréquence des oscillations est différente et le comportement global est beaucoup moins stable.

C'est un cas difficile où l'on ne mesure que le comportement dynamique et l'on peut comprendre que ces imperfections se remarquent moins sur l'exemple du foie où l'on se contente généralement d'appuyer et de maintenir la déformation, puis de relâcher.

Ces instabilités peuvent s'expliquer par le mauvais comportement des opérateurs différentiels lorsqu'on les applique sur des points trop éloignés d'une grille régulière (où ils rejoignaient les résultats des différences finies). Une particule entourée de voisines d'un niveau différent doit en effet utiliser des points d'échantillonnage assez mal répartis autour d'elle.

Il est possible de corriger ce comportement en passant d'une méthode adaptative à une méthode hiérarchique.

7.2 Une méthode hiérarchique

Puisque les particules semblent mal parvenir à prendre en compte leurs voisines issues d'une résolution différente, on va modifier l'algorithme pour que seules les voisines actives *du même niveau* soient prises en compte.

La modification principale réside dans le fait que lorsqu'une particule se divise, elle *reste* active (voir Fig. 3.13).

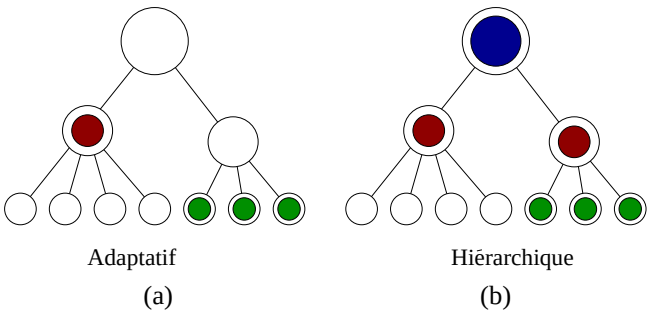


FIG. 3.13: L'adaptatif (a) ne simule que le niveau le plus fin choisi alors qu'en hiérarchique (b), les niveaux inférieurs continuent à être actifs et cohabitent.

Une particule divisée va continuer à calculer une force basée sur le déplacement de ses voisines, mais en ne considérant maintenant plus que celles *actives et non divisées*, du même niveau qu'elle. Elle ne prendra donc plus en compte que les influences des particules actives du même niveau qu'elle. La cohabitation entre les niveaux s'en trouve modifiée.

Descente des forces

La force calculée par la particule *divisée* ne va par contre pas être classiquement intégrée mais *transmise* aux filles. Celles-ci vont ainsi recevoir de la part de leur mère une force qui exprimera les interactions ayant eu lieu entre la zone qu'elles occupent et les zones actives voisines simulées à une résolution plus grossière.

Cette descente va se faire à chaque mise-à-jour de la force et va se propager à toute la descendance *active* de la particule (filles, petites filles, etc...). Toutes les particules actives de cette zone vont ainsi prendre en compte les contraintes générées par la mère et ses voisines de même niveau. Les interactions ayant eu lieu à l'intérieur de ce maillage entre cette zone et les zones voisines sont ainsi propagées aux filles des niveaux inférieurs.

Remontée des positions

Inversement, les mères vont pouvoir prendre en compte les interactions de leurs filles avec les niveaux *inférieurs* grâce à une mise à jour directe de leur position. Les particules divisées déduisent en effet leur position de celle de leur fille par barycentre (Eq. 3.10), reflétant ainsi ce qui a pu se passer dans cette zone aux niveaux inférieurs.

La particule *divisée* ne sert ainsi que en quelque sorte que d'indicateur grossier de ce que ses filles (ou petites-filles si celles-ci sont aussi divisées, etc...) ont simulé finement dans cette zone.

Sa position ainsi actualisée sert à calculer les forces intervenant avec les voisines non divisées pour pouvoir la communiquer à ses filles. On crée ainsi avec cette descente de forces et cette remontée de positions la liaison entre les maillages indispensable à tout processus multirésolution.

Algorithme

L'animation d'un niveau de résolution donné se passe ainsi :
- Pour les particules *divisées* seulement, remettre à jour la position en remontant récursivement celle de sa descendance active par barycentre (en descendant jusqu'aux particules actives non divisées).
- Calculer la force engendrée par les déplacements des voisines de même niveau, actives et non divisées grâce aux opérateurs différentiels.
- Pour les particules divisées, transmettre la force ainsi calculée à toute leur descendance active, qui la stockera.
- Pour les particules *non* divisées, ajouter à cette force celles reçues de tout le reste l'ascendance (mère, grand-mère, etc...) et intégrer la force totale pour déterminer la nouvelle position.

L'algorithme est un peu plus complexe si on autorise la multirésolution temporelle décrite en Section 5. On ne remet plus alors à jour les particules niveau par niveau, mais selon le pas de temps de chacune. Le principe reste le même : remontée de position, calcul et descente de force pour les particules divisées et calcul et intégration de la force cumulée pour les particules actives non divisées.

Avec un tel algorithme, on n'a plus d'interaction (*i.e.* de calcul de force en fonction des déplacements relatifs) qu'entre des particules de même niveau. On va ainsi limiter l'imprécision des opérateurs précédemment décrits qui nuisait aux résultats multirésolution.

Mesure du surcoût du hiérarchique

La contrepartie est une augmentation du nombre de particules actives simulées, et donc du coût de l'animation. Cette augmentation reste néanmoins très limitée. Supposons par exemple un matériau composé de 4 maillages, composés de respectivement 4, 32, 256 et 2048 particules. Prenons, pour simplifier, le cas où à un instant donné, la moitié des particules de chaque niveau est subdivisée (subdivision régulière).

En conservant les mères comme actives dans la méthode hiérarchique qui vient d'être proposée, on doit calculer la force subie par $4+\frac{1}{2}32+\frac{1}{2^2}256+\frac{1}{2^3}2048 = 4+16+64+256 = 340$ particules. L'algorithme adaptatif n'aura pas à prendre en compte les particules divisées, ce qui donnera $\frac{1}{2}4+\frac{1}{2}16+\frac{1}{2}64+256 = 298$ particules actives.

On aura donc dans cet exemple une augmentation de 14% du nombre de particules actives et donc approximativement du temps de calcul (celui-ci étant principalement constitué du calcul de la force et de l'intégration des positions des particules actives) entre la version hiérarchique et celle adaptative de la méthode.

7.3 Problème avec le grad(div)

Nous n'avons pas poussé les tests avec cette nouvelle méthode, que nous avons décrite ici pour l'intérêt de son principe. En effet, nous nous sommes aperçus que l'opérateur servant à calculer le $\mathbf{grad}(\text{div}\,\mathbf{u})$ donnait des résultats erronés.

Ceci s'est révélé lors de tests réalisés en 2D en étudiant les résultats des opérateurs lorsqu'on les appliquait à des champs *analytiques*. Les deux opérateurs s'expriment en 2D comme :

$$\Delta\mathbf{u} = \begin{cases} u_{x,xx}+u_{x,yy} \\ u_{y,xx}+u_{y,yy} \end{cases} \qquad \mathbf{grad}(\text{div}\,\mathbf{u}) = \begin{cases} u_{x,xx}+u_{y,xy} \\ u_{x,xy}+u_{y,yy} \end{cases}$$

Notons tout d'abord que ces opérateurs sont totalement indépendants de la direction dans laquelle se trouvent les voisins, à partir du moment où ceux-ci sont sur une grille régulière, qui n'a donc pas besoin d'être alignée avec les axes. Ce résultat était prévisible dans la mesure où ces opérateurs ne font intervenir que les distances entre les points et non leur position absolue.

Si l'opérateur laplacien donne bien les résultats escomptés, l'opérateur $\mathbf{grad}(\text{div}\,\mathbf{u})$ tel que nous l'utilisons calcule en fait $\mathbf{grad}(\text{div}\,\mathbf{u}) = \begin{cases} u_{x,xx} \\ u_{y,yy} \end{cases}$, les dérivées croisées n'apparaissant pas.

Si au lieu de considérer la projection le long des directions radiales comme on le fait, on prend les directions tangentes, on crée un opérateur calculant $\mathbf{grad}(\text{div}\,\mathbf{u}) = \begin{cases} u_{x,yy} \\ u_{y,xx} \end{cases}$, ce qui est normal puisque la somme de ces deux composantes doit donner le laplacien.

L'assimilation faite dans la Section 2 entre le $\mathbf{grad}(\text{div}\,\mathbf{u})$ et la partie non rotationnelle du champ était donc trop simplificatrice. La version publiée de ce chapitre [DDBC99, DC99a] ne mentionnait pas ce problème, apparu lors de tests postérieurs. On a également pu mesurer sur cet exemple analytique que les résultats se dégradaient assez vite lorsqu'on s'éloignait de la grille régulière.

Le problème plus profond de cet opérateur est qu'il est incapable de calculer une *dérivée croisée*. On a beau modifier l'axe selon lequel on projette, elles ne peuvent apparaître avec cette méthode. Les approximations faites sont trop arbitraires et l'on ne voit pas où l'on pourrait retrouver les termes de dérivée croisée. D'autres pistes de recherche menées en parallèle semblant promettre de meilleurs résultats, nous avons préféré les privilégier pour obtenir une nouvelle expression des opérateurs.

8 Conclusion

Un calcul des opérateurs intervenant dans l'équation de Navier a été proposé. Donnant de bons résultats indépendants de la résolution, ils ont pu être utilisés dans une méthode multirésolution adaptative utilisant un octree.

Les résultats visuels sont assez convaincants et un premier modèle de simulateur chirurgical a été mis au point. Il permet de simuler le mouvement d'une centaine de points en temps-réel, la résolution s'adaptant automatiquement lors de la simulation.

Les tests plus formels que nous avons effectués sur ces opérateurs ont révélé des insuffisances numériques. Cette médiocre qualité nous a conduit à chercher une autre façon de calculer ces différentielles, en partant de considérations géométriques et mathématiques plus poussées, comme nous allons le décrire dans le prochain chapitre.

CHAPITRE 4

Nouveaux opérateurs différentiels

MIEUX CALCULER les opérateurs différentiels impliqués dans les modèles continus de la matière est le but de ce chapitre. Nous allons pour cela utiliser des principes mathématiques et géométriques plus évolués, permettant de mieux cerner les hypothèses que nous ferons. Les calculs seront dans un premier temps menés en deux dimensions, leur expression et leur interprétation étant plus simples. Nous généraliserons ensuite les résultats en trois dimensions.

Les résultats mathématiques obtenus s'avèrent s'apparenter à ceux issus d'une certaine classe d'éléments finis, offrant ainsi un point de vue original sur cette méthode. Nous comparerons ces deux méthodes, ainsi que d'autres à base de masses-ressorts, pour choisir celle qui semble la mieux adaptée au développement d'un modèle multirésolution.

1 Le théorème de Gauss

1.1 Expression mathématique

Les calculs mathématiques détaillés dans les sections suivantes s'appuient sur un théorème classique de physique : le théorème de Gauss. Ce théorème fut énoncé sous diverses formes par Lagrange (1762), Gauss (1813), Green (1828) ou Ostrogradski (1831) et n'a en conséquence pas le même nom dans tous les pays. Quoi qu'il en soit, il permet de réduire le calcul d'une intégrale sur un volume à celui d'une intégrale sur la *surface* du volume :

$$\boxed{\int_V \frac{\partial}{\partial i} x \, dV = \int_{\partial V} x \, n_i \, dS \quad , \forall i} \tag{4.1}$$

x est un champ scalaire, et $\mathbf{n}$ la normale à la surface d'intégration ∂V, frontière de V. En appliquant Gauss composante par composante, on obtient la version vectorielle du théorème :

$$\int_V \frac{\partial}{\partial i} X_i \, dV = \int_{\partial V} X_i \, n_i \, dS \quad , \forall i \tag{4.2}$$

Ces expressions sont les versions les plus générales du théorème, on le connaît mieux sous la forme suivante,

dite d'Ostrogradski :

$$\int_V \operatorname{div} \mathbf{X}\, dV = \int_{\partial V} \mathbf{X} \cdot \mathbf{n}\, dS \tag{4.3}$$

On peut repasser de cette écriture aux précédentes en l'appliquant à un champ valant X_i sur sa $i^{ème}$ composante et 0 sur les autres composantes.

Le passage d'une intégrale sur le volume à une intégrale de surface s'explique intuitivement par le fait qu'il suffit de mesurer ce qui entre et sort de la surface (la composante normale à la surface du champ) pour mesurer l'évolution de ce qui se passe à l'intérieur du volume.

Outre l'intéressant passage du volume à la surface, ce théorème permet de calculer la valeur moyenne de la *différentielle* d'un champ à l'aide des *valeurs* de ce champ sur la surface. On gagne ainsi un ordre dans l'expression des différentielles.

Le principe des calculs qui vont suivre est d'appliquer le théorème de Gauss à de petites régions de l'espace entourant chacune une particule. On va supposer ces volumes de l'espace suffisamment petits pour y considérer la dérivée $\frac{\partial}{\partial i} X_i$ comme constante et pouvant donc être extraite de l'intégrale volumique. L'intégrale de surface pourra quand à elle être évaluée en faisant des hypothèses sur la forme du champ vectoriel considéré.

1.2 Définition du volume

Le volume sur lequel nous allons appliquer le théorème de Gauss est défini comme étant la *région de Voronoï* associée à chaque particule (voir par exemple [dBvKOS97]). Cette région est celle contenant les points de l'espace qui sont plus proches de la particule considérée que d'aucune autre particule (voir Figure 4.1). C'est donc la zone la plus à même d'être *représentée* par la particule. Ces régions s'adaptent ainsi naturellement à la disposition des points d'échantillonnage, leur union couvrant tout l'espace et donnant à chaque particule une importance adéquate.

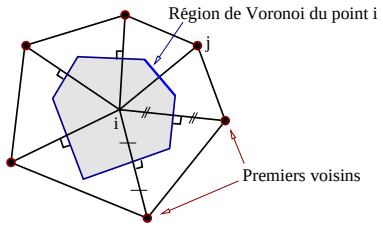


FIG. 4.1: Le point *i* représentera les valeurs du champ à l'intérieur de sa région de Voronoï.

Dans la suite de ce chapitre, la région de Voronoï associée à une particule sera souvent appelée par simplicité "le Voronoï" de cette particule.

Utiliser la définition d'un volume propre pour chaque particule qui s'appuie sur la division de l'objet en éléments peut être rapproché de la méthode des volumes finis [McC89].

Ce choix peut également être rapproché de la méthode des *éléments naturels* [1] [BS95] qui permet, en choisissant des éléments finis suivant les régions de Voronoï d'obtenir de bien meilleurs résultats.

1.3 Champ linéaire par morceaux

L'objet considéré est supposé maillé à l'aide de tétraèdres (de triangles si on est en 2D). Ce choix est motivé par le fait que ce sont les éléments les plus simples capables de mailler l'espace et nous reviendrons sur le processus d'obtention de ces maillages. Les points d'échantillonnage que sont les particules constituent les sommets des tétraèdres de notre maillage.

Le but est ici d'évaluer en chaque particule la valeur de certaines différentielles du champ déplacement **u**, pour pouvoir appliquer l'équation de Navier (voir Chapitre 2). Ce champ n'étant échantillonné qu'en chaque particule, il est naturel de le considérer comme *linéaire par morceaux* sur chacun des tétraèdres (resp. triangles) puisqu'on ne dispose pas de plus d'information sur sa valeur à l'intérieur des tétraèdres. C'est également une manière très simple et rapide de calculer la valeur du champ en tout point de l'élément.

[1] *Natural Element Method*.

Le volume d'intégration et la forme du champ étant définis, reste à choisir le champ **X** auquel nous allons appliquer Gauss et qui amènera à l'expression du laplacien et du **grad** div . C'est en prenant respectivement le **div** et le **grad** du champ déplacement que l'on arrive au résultat, comme nous allons le détailler dans les deux sections suivantes.

2 Calcul du laplacien

2.1 Principe

Le laplacien du champ déplacement vaut $\Delta\mathbf{u} = \mathbf{div}\,(\mathbf{grad}\,\mathbf{u})$, où **div** est la divergence d'une matrice, composée des divergences de chacune des lignes[2]. En appliquant Gauss (Eq. 4.3) à la composante ε de $\Delta\mathbf{u}$ on obtient :

$$\int_V (\Delta\mathbf{u})_\varepsilon \, dV = \int_V \Delta(\mathbf{u}_\varepsilon)\, dV = \int_V \mathrm{div}\,(\mathbf{grad}\, u_\varepsilon)\, dV = \int_{\partial V} (\mathbf{grad}\, u_\varepsilon)\cdot\mathbf{n}\, dS \tag{4.4}$$

On va décomposer l'intégrale de droite en la somme des intégrales sur chacun des triangles (arêtes en 2D) composant la frontière de la région de Voronoï d'une particule i. L'intêret de la méthode est qu'après avoir grâce à Gauss supprimé le terme en div , le terme **grad** u_ε qui reste dans la partie droite de l'Équation 4.4 est *constant* sur chacun des tétraèdres ou triangles (dérivée spatiale première d'un champ linéaire) et que son produit scalaire avec la normale va pouvoir être exprimé simplement.

2.2 Expression en 2D

Pour simplifier les expressions dans les explications qui vont suivre, on va tout d'abord étudier le cas où l'on est en deux dimensions. Nous généraliserons les formules obtenues à la 3D dans la Section 4.

Les arêtes qui composent la frontière de la région de Voronoï du point i passent par les milieux des arêtes reliant i à ses voisins. Elles leurs sont orthogonales et se rejoignent en chacun des *orthocentres* des triangles entourant i.

Soit (i,j,k) un de ces triangles, d'orthocentre noté m, et soit j' le milieu de (i,j) (voir Fig. 4.2). Sur la demi-arête (j',m) de normale dirigée selon $\overrightarrow{ij}$, le produit $(\mathbf{grad}\, u_\varepsilon)\cdot\mathbf{n}$ est constant et vaut :

$$(\mathbf{grad}\, u_\varepsilon)\cdot\mathbf{n} = \frac{u_\varepsilon^j - u_\varepsilon^i}{\|\overrightarrow{ij}\|}$$

C'est en effet sa valeur le long du segment (i,j), donc en particulier en j' et donc sur toute l'arête (j',m) puisque ce terme est constant à l'intérieur du triangle (i,j,k).

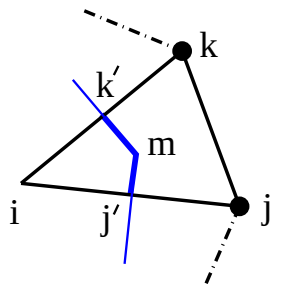


FIG. 4.2: On décompose l'intégrale sur le contour en la somme des intégrales sur les demi-arêtes. m est l'orthocentre du triangle, j' et k' les milieux des arêtes.

On va donc pouvoir réécrire :

$$\int_{j'}^{m} (\mathbf{grad}\, u_\varepsilon)\cdot\mathbf{n}\, dl = \frac{u_\varepsilon^j - u_\varepsilon^i}{\|\overrightarrow{ij}\|}\int_{j'}^{m} dl = \frac{u_\varepsilon^j - u_\varepsilon^i}{\|\overrightarrow{ij}\|}\ \|\overrightarrow{j'm}\| \tag{4.5}$$

La longueur $\|\overrightarrow{j'm}\|$ de la demi-arête (j',m) peut s'exprimer à l'aide des caractéristiques géométriques du triangle (i,j,k). On appelle α, β et γ les demi-angles aux sommets du triangle (voir Fig. 4.3(a)).

[2] Voir l'Annexe A.

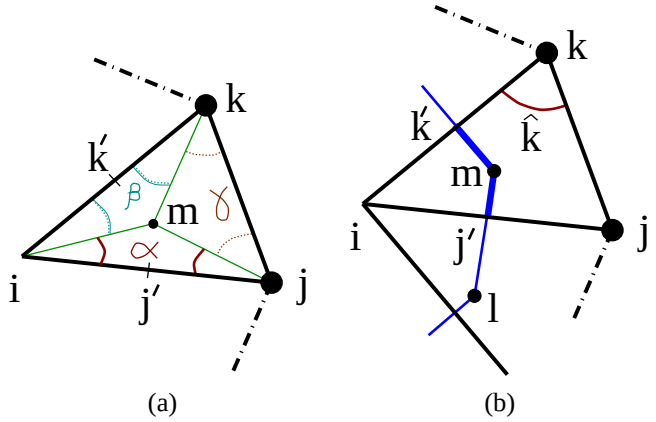


FIG. 4.3: On utilise les angles α, β et γ pour obtenir une expression en fonction de $\hat{k}$.

On a alors $\|\overrightarrow{j'm}\| = \tan\alpha \, \frac{\|\overrightarrow{ij}\|}{2} = \frac{1}{2}\tan\alpha \, \|\overrightarrow{ij}\|$.
La somme des angles d'un triangle valant π, on a $2\alpha + 2\beta + 2\gamma = \pi$, soit $\alpha + \beta + \gamma = \frac{\pi}{2}$.
Ainsi $\tan\alpha = \tan(\frac{\pi}{2} - \beta - \gamma) = \tan(\frac{\pi}{2} - \hat{k}) = \text{cotg}\,\hat{k}$, où $\hat{k}$ est l'angle en k du triangle (voir Fig. 4.3(b)).
Finalement :

$$\int_{j'}^{m} (\mathbf{grad}\, u_\varepsilon) \cdot \mathbf{n}\, dl = \frac{u_\varepsilon^j - u_\varepsilon^i}{\|\overrightarrow{ij}\|} \; \frac{1}{2}\text{cotg}\,\hat{k}\, \|\overrightarrow{ij}\| = \frac{1}{2}\text{cotg}\,\hat{k}\, (u_\varepsilon^j - u_\varepsilon^i)$$

2.3 Intégrale sur tout le contour

On peut réunir les intégrales sur les deux demi-arêtes (l, j') et (j', m) (Fig. 4.3b) situées de part et d'autre de l'arête (i, j) et qui forment le contour de la région de Voronoï de i pour obtenir :

$$\int_{l}^{m} (\mathbf{grad}\, u_\varepsilon) \cdot \mathbf{n}\, dl = \frac{1}{2}(\text{cotg}\,\alpha_j + \text{cotg}\,\beta_j)\; (u_\varepsilon^j - u_\varepsilon^i) \tag{4.6}$$

Les angles α_j et β_j étant définis comme ceux opposés à l'arête (i, j) (voir Fig. 4.4).

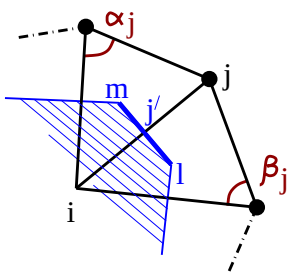


FIG. 4.4: Les cotg des angles opposés à l'arête viennent pondérer la contribution d'un voisin.

Ces deux coefficients cotg α_j et cotg β_j viennent pondérer l'influence du voisin j sur le calcul du laplacien. Remarquons encore une fois leur possible interprétation comme celle d'un filtre, leur valeur augmentant lorsque le voisin j se rapproche de i. Elle s'annule, puis devient négative lorsque les angles dépassent 90 degrés. Dans les maillages que nous utiliserons par la suite, ce cas ne se présentera pas et l'on n'aura affaire qu'à des coefficients positifs.

Le laplacien est supposé suffisamment continu sur la région de Voronoï pour pouvoir être extrait de l'intégrale. Mathématiquement, cela signifie simplement que sa valeur moyenne est atteinte au point i, mais intuitivement on pourra plutôt considérer le laplacien comme évoluant suffisamment *régulièrement* sur la région de Voronoï pour que sa valeur au point central i représente convenablement sa *valeur moyenne* sur la région.

On peut alors écrire :

$$\begin{aligned}
\int_A (\Delta\mathbf{u})_\varepsilon \, dV &= \int_{\partial A} (\mathbf{grad}\, u_\varepsilon) \cdot \mathbf{n} \, dS \\
A \;\; (\Delta\mathbf{u})_\varepsilon &= \sum_{\text{voisins } j} \int_l^m (\mathbf{grad}\, u_\varepsilon) \cdot \mathbf{n} \, dl \\
&= \sum_{\text{voisins } j} \frac{1}{2} (\text{cotg}\, \alpha_j + \text{cotg}\, \beta_j) \; (u_\varepsilon^j - u_\varepsilon^i). \qquad (4.7)
\end{aligned}$$

Soit finalement, pour le point i, en regroupant chacune des composantes ε du laplacien :

$$\boxed{(\Delta\mathbf{u})^i = \frac{1}{2A} \sum_{\text{voisins } j} (\text{cotg}\, \alpha_j + \text{cotg}\, \beta_j) \; (u^j - u^i).} \qquad (4.8)$$

où A désigne l'aire totale de la région de Voronoï associée au point i.

Noter que bien que l'on a considéré le champ $\mathbf{u}$ comme localement linéaire, ce qui devrait normalement conduire à une dérivée seconde nulle, le fait d'en considérer la valeur comme une fonction *par morceaux* sur plusieurs triangles crée un laplacien non nul aux sommets.

Cette formulation du laplacien est en fait assez classique. Nous l'avons en effet par la suite retrouvée, obtenue de manière différente, dans un ouvrage d'analyse numérique de référence ([LT94], page 115).

2.4 Mise en œuvre

Pour une arête (i, j) donnée, le coefficient $(\text{cotg}\, \alpha_j + \text{cotg}\, \beta_j)$ ne dépend que de la géométrie du maillage. En faisant la classique hypothèse des faibles déformations, on peut considérer ce coefficient comme constant durant la simulation et donc le précalculer dans la position de repos. C'est un simple scalaire qui peut être stocké dans l'arête (i, j).

Noter que cette hypothèse des petites déformations signifie seulement que l'on reste dans le domaine de comportement linéaire du matériau. On ne fait pas d'approximation en considérant le terme en *cotg* comme constant lors de la déformation. L'interaction entre deux particules est en effet liée à la géométrie *intrinsèque* de l'objet. Ce sont en quelque sorte les atomes entre les particules, leur nombre, leur disposition, qui vont définir les interactions entre deux points. Ces données ne changent pas au cours de la simulation (sauf si l'on fait intervenir des découpes) et sont donc celles calculées dans la position de repos.

Calculer le laplacien revient donc pour chaque particule à simplement faire une somme *pondérée* des déplacements de ses voisines, ce qui est très rapide. Plus rapide même, dans cette hypothèse, qu'avec un réseau masses-ressorts où chaque arête demande le calcul de sa longueur et donc l'emploi d'une coûteuse racine carrée.

Recalculer ces *cotg* (lorsque l'objet s'est trop déformé, que l'on souhaite un comportement non linéaire, ou lors de découpes éventuelles) reste très peu cher : deux produits scalaires, deux produits vectoriels, une norme et une division.

2.5 Interprétation

La formulation de l'Équation 4.8 satisfait le principe d'action-réaction, la force exercée par i sur j étant l'opposée de celle exercée par j sur i.

Si l'on ne considère plus que le *déplacement relatif* d'un point i par rapport à ses voisins, ce laplacien s'interprète assez simplement. Tout déplacement de l'un des voisins va en effet se traduire par un laplacien, et donc par une force, proportionnelle à ce déplacement. Le point i aura ainsi tendance à suivre chacun de ses voisins dans leurs déplacements. Tous les voisins étant susceptibles de bouger, le point i déduira son déplacement propre d'une moyenne pondérée de ces influences.

Les coefficients pondérateurs $(\text{cotg}\, \alpha_j + \text{cotg}\, \beta_j)$ sont liés à la géométrie. On peut grossièrement considérer leur influence comme proportionnelle au rapprochement du point i et de son voisin j, ce qui est naturel. L'isovaleur exacte donnant les positions où un voisin j a une influence donnée sur le point i décrit une hyperbole. La Figure 4.5 montre les différentes valeurs de $\text{cotg}\, \alpha_j + \text{cotg}\, \beta_j$ lorsque le point j est déplacé. Les deux voisins $j-1$ et $j+1$ sont supposés fixes et formant ici un angle de $90°$ en i.

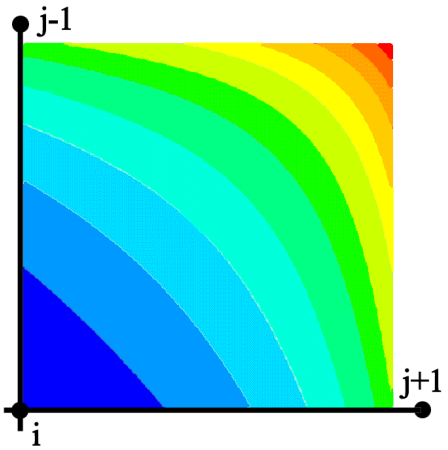


FIG. 4.5: Représentation des iso-valeurs du coefficient pondérateur d'un point j sur le laplacien calculé en i en fonction de la position de j.

2.6 Qualité de l'opérateur

Ce calcul du laplacien est exact si l'on considère un champ linéaire par morceaux. Voyons quelles sont les conditions nécessaires pour qu'il le soit pour un champ *quadratique* par morceaux. Dans la pratique, les champs seront en effet arbitrairement complexes. On peut les classifier en faisant un développement limité de Taylor, pour isoler leurs composantes constante, linéaire, quadratique et d'autres termes de plus haut degré.

Il est donc utile d'étudier le comportement de cet opérateur sur un champ quadratique par morceaux pour mesurer son adéquation à des champs plus complexes que ceux linéaires par morceaux qui nous ont conduits à sa formulation.

Le gradient est maintenant un champ *linéaire* sur chacun des triangles et l'intégrale sur la demi-arête n'est plus son simple produit par la longueur de l'arête. Le gradient calculé au milieu j' de l'arête (i,j), $\frac{\mathbf{u}^j-\mathbf{u}^i}{\|\overrightarrow{ij}\|}$, est néanmoins encore exact à l'ordre 2 en ce point. Un développement limité de Taylor à l'ordre 2 de la valeur de $\mathbf{u}$ en j' donne, en effet, en prenant classiquement $h=\frac{\|\overrightarrow{ij}\|}{2}$:

$$\mathbf{u}^j = \mathbf{u}^{j'} + h\,(\mathbf{grad}\,\mathbf{u})^{j'} + \frac{h^2}{2}\left(\frac{\partial^2\mathbf{u}}{\partial x}\right)^{j'} + O(h^2) \tag{4.9}$$

$$\mathbf{u}^i = \mathbf{u}^{j'} - h\,(\mathbf{grad}\,\mathbf{u})^{j'} + \frac{h^2}{2}\left(\frac{\partial^2\mathbf{u}}{\partial x}\right)^{j'} + O(h^2) \tag{4.10}$$

Soit en soustrayant les deux expressions :

$$\mathbf{u}^j - \mathbf{u}^i = 2h\,(\mathbf{grad}\,\mathbf{u})^{j'} + O(h^2)$$

$$\begin{aligned}(\mathbf{grad}\,\mathbf{u})^{j'} &= \frac{\mathbf{u}^j-\mathbf{u}^i}{2h} + \frac{O(h^2)}{h} \\ &= \frac{\mathbf{u}^j-\mathbf{u}^i}{\|\overrightarrow{ij}\|} + O(h^3)\end{aligned} \tag{4.11}$$

De plus, lorsque les angles α_j et β_j sont égaux, l'arête de Voronoï a la même longueur de part et d'autre de j' ($\|\overrightarrow{lj'}\| = \|\overrightarrow{j'm}\|$) et l'intégrale calculée (valeur en j' multipliée par la longueur de l'arête) est alors exacte à l'ordre deux, quelle que soit la pente du gradient (formule du trapèze).

La Formule 4.8 précédente reste donc exacte pour un champ quadratique lorsque les deux angles opposés à chaque arête sont égaux. Plusieurs maillages satisfont cette propriété, comme celui associé à une grille régulière, avec des angles de 45° et 90°. Ce cas est quelque peu dégénéré (*cotg* nuls) et on lui préférera le maillage composé de triangles équilatéraux de même taille pour optimiser les résultats. Il n'est par contre pas possible de créer un maillage de tétraèdres réguliers en 3D.

On retiendra de cette propriété que le calcul du laplacien est d'autant plus exact (car s'adaptant à des champs plus complexes) que le maillage est régulier. Cette propriété gouvernera la création et l'optimisation des maillages que nous utiliserons par la suite.

3 Calcul du grad(div u)

3.1 Principe

Nous allons calculer le **grad**(div **u**) en utilisant la même méthode. Nous appliquons cette fois le théorème de Gauss (Eq. 4.1) à chacune des composantes ε du vecteur gradient, en prenant comme scalaire x le div **u**. On obtient alors :

$$\int_V (\mathbf{grad}\ \mathrm{div}\ \mathbf{u})_\varepsilon\, dV = \int_V \frac{\partial}{\partial x_\varepsilon} \mathrm{div}\ \mathbf{u}\, dV = \int_{\partial V} \mathrm{div}\ \mathbf{u}\ n_\varepsilon\, dS \tag{4.12}$$

Soit, en regroupant les expressions obtenues sur chacune des composantes ε :

$$\int_V (\mathbf{grad}\ \mathrm{div}\ \mathbf{u})\, dV = \int_{\partial V} (\mathrm{div}\ \mathbf{u})\ \mathbf{n}\, dS \tag{4.13}$$

On va à nouveau décomposer l'intégrale de droite en une somme d'intégrales sur les frontières de la région de Voronoï. La divergence est, tout comme le gradient, constante sur chacun des triangles du maillage (somme de dérivées premières d'un champ supposé linéaire).

Nous allons, pour évaluer l'expression de la divergence, calculer celle du gradient des différentes composantes de **u**. Encore une fois, nous détaillons ici les calculs en 2D, leur extension à la 3D étant reportée à la Section 4.

3.2 Calcul du gradient

Soit un champ scalaire f (une des composantes de **u** en pratique), connu aux trois sommets (i, j, k) d'un triangle par ses valeurs f^i, f^j et f^k, et linéaire à l'intérieur du triangle.

La valeur de f au point **x** intérieur au triangle est donnée par la somme pondérée des trois fonctions de base :

$$f(\mathbf{x}) = f^i\, W^i(\mathbf{x}) + f^j\, W^j(\mathbf{x}) + f^k\, W^k(\mathbf{x})$$

chaque $W(\mathbf{x})$ étant la fonction linéaire valant 1 au sommet considéré et 0 aux autres. Le gradient de ces fonctions de base est un vecteur constant et le gradient de f s'écrit :

$$\mathbf{grad}\ f = f^i\ \mathbf{grad}\ W^i + f^j\ \mathbf{grad}\ W^j + f^k\ \mathbf{grad}\ W^k$$

La somme des fonctions de base est, sur le triangle, une fonction linéaire valant 1 aux trois sommets. C'est donc une fonction constante sur le triangle : $W^i + W^j + W^k \equiv 1$. On a donc $\mathbf{grad}\ W^i + \mathbf{grad}\ W^j + \mathbf{grad}\ W^k = 0$ et le gradient de f peut se réécrire :

$$\begin{aligned} \mathbf{grad}\ f &= f^i \left(-\mathbf{grad}\ W^j - \mathbf{grad}\ W^k\right) + f^j\ \mathbf{grad}\ W^j + f^k\ \mathbf{grad}\ W^k \\ &= (f^j - f^i)\mathbf{grad}\ W^j + (f^k - f^i)\mathbf{grad}\ W^k \end{aligned} \tag{4.14}$$

Désignons par h^i la hauteur du triangle au point i (Fig. 4.6).

Le gradient de W^i, que nous noterons α^i, vaut[3] :

$$\alpha^i = \mathbf{grad}\ W^i = \frac{1}{h^i} \frac{\overrightarrow{jk}^\perp}{||\overrightarrow{jk}||} \tag{4.15}$$

[3]Cette notation est choisie pour sa similarité avec des travaux déjà publiés sur le sujet [DDCB00, Cot97]. Il ne faudrait pas confondre ce vecteur α^i avec l'angle α_i du triangle utilisé plus haut. Dans tout ce qui suit α^i représente le gradient.

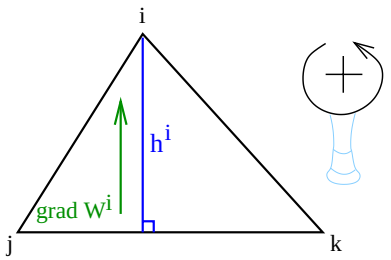


FIG. 4.6: Le gradient de la fonction de base d'un sommet est dirigé selon la normale à l'arête opposée.

où $\overrightarrow{jk}^{\perp}$ désigne le vecteur $\overrightarrow{jk}$ tourné de 90 degrés dans le sens trigonométrique[4]. L'écriture aux sommets j et k se déduit par simple changement d'indices.

Introduisons également l'aire A_{ijk} du triangle (i, j, k), qui peut s'écrire comme :

$$2A_{ijk} = h^i \, \|\overrightarrow{jk}\| = h^j \, \|\overrightarrow{ki}\| = h^k \, \|\overrightarrow{ij}\|$$

Le gradient de f s'écrit alors finalement comme :

$$\begin{aligned} \textbf{grad}\, f &= (f^j - f^i)\, \alpha^j + (f^k - f^i)\, \alpha^k \\ &= (f^j - f^i)\, \frac{1}{h^j \, \|\overrightarrow{ki}\|}\, \overrightarrow{ki}^{\perp} + (f^k - f^i)\, \frac{1}{h^k \, \|\overrightarrow{ij}\|}\, \overrightarrow{ij}^{\perp} \\ &= \frac{1}{2A_{ijk}} \left((f^j - f^i)\, \overrightarrow{ki}^{\perp} + (f^k - f^i)\, \overrightarrow{ij}^{\perp} \right) \end{aligned} \tag{4.16}$$

3.3 Calcul de la divergence

Toujours en 2D, la divergence div $\mathbf{u}$ s'écrit comme div $\mathbf{u} = u_{x,x} + u_{y,y}$. C'est donc la somme de la première composante de **grad** f avec $f = u_x$ et de la seconde composante de **grad** f avec $f = u_y$.

On a :

$$\textbf{grad}\, f = \begin{pmatrix} (\textbf{grad}\, f)_x \\ (\textbf{grad}\, f)_y \end{pmatrix} = \frac{1}{2A_{ijk}} \left[(f^j - f^i) \begin{pmatrix} \overrightarrow{ki}_y \\ -\overrightarrow{ki}_x \end{pmatrix} + (f^k - f^i) \begin{pmatrix} \overrightarrow{ij}_y \\ -\overrightarrow{ij}_x \end{pmatrix} \right] \tag{4.17}$$

La divergence vaut donc, sur le triangle (i, j, k) :

$$\text{div}\, \mathbf{u} = \frac{1}{2A_{ijk}} \left[\left((u_x^j - u_x^i)\, \overrightarrow{ki}_y + (u_x^k - u_x^i)\, \overrightarrow{ij}_y \right) + \left(-(u_y^j - u_y^i)\, \overrightarrow{ki}_x - (u_y^k - u_y^i)\, \overrightarrow{ij}_x \right) \right] \tag{4.18}$$

3.4 Intégrale sur tout le contour

La divergence étant exprimée sur chaque triangle, il ne reste qu'à intégrer le produit $(\text{div}\, \mathbf{u})\, \mathbf{n}$ sur tout le contour de la région de Voronoï pour obtenir $[\textbf{grad}(\text{div}\, \mathbf{u})]$ (Eq. 4.13).

Sur chaque triangle,

$$\int_{\partial A} (\text{div}\, \mathbf{u})\, \mathbf{n}\, dl = (\text{div}\, \mathbf{u}) \int_{\partial A} \mathbf{n}\, dl$$

Intégrer sur un segment un vecteur unitaire qui lui est normal donne comme résultat le vecteur correspondant au segment, tourné de 90 degrés.

[4] Ce choix du sens trigonométrique permet d'assurer que $\overrightarrow{jk}^{\perp}$ sera dirigé vers i et ne pointera donc pas vers l'extérieur du triangle. Il en est ainsi lorsque les points i, j et k sont eux mêmes positionnés dans l'ordre trigonométrique. Si on nomme les points dans le sens inverse, un signe $-$ apparaît ici, mais il sera compensé lors du calcul de la normale à l'arête (Eq. 4.19). Aussi ne reviendrons-nous pas sur cette nuance de numérotation, nommée *inversion de la girafe* pour d'obscures raisons.

En intégrant sur tout le contour, on obtient donc :

$$\begin{aligned}
\int_A \mathbf{grad}\,(\mathrm{div}\,\mathbf{u})\,dS &= \int_{\partial A} (\mathrm{div}\,\mathbf{u})\ \overrightarrow{dl}^{\perp} \\
&= \sum_{triangles\ du\ Voronoi} \left[(\mathrm{div}\,\mathbf{u}) \left(\int_{j'}^{m} \overrightarrow{dl}^{\perp} + \int_{m}^{k'} \overrightarrow{dl}^{\perp} \right) \right] \\
&= \sum_{triangles\ du\ Voronoi} \left[(\mathrm{div}\,\mathbf{u}) \left(\overrightarrow{j'm}^{\perp} + \overrightarrow{mk'}^{\perp} \right) \right] \\
&= \sum_{triangles\ du\ Voronoi} \left((\mathrm{div}\,\mathbf{u})\ \overrightarrow{j'k'}^{\perp} \right) \\
&= \frac{1}{2} \sum_{triangles\ du\ Voronoi} \left((\mathrm{div}\,\mathbf{u})\ \overrightarrow{jk}^{\perp} \right) \qquad (4.19)
\end{aligned}$$

Sur chaque triangle, l'intégrale ne dépend donc pas directement de la forme du Voronoï (dans notre cas, les deux vecteurs $\overrightarrow{j'm}$ et $\overrightarrow{mk'}$), mais seulement des deux points extrêmes j' et k' car on somme ensuite les vecteurs formant le contour.

À la limite, dans ce calcul, peu importe la forme de la région considérée pourvu qu'elle passe par les points j' et k' et reste à l'intérieur du triangle (i, j, k).

Tout comme pour le laplacien, on suppose le **grad**(div **u**) suffisamment continu pour pouvoir être représenté par sa valeur moyenne et ainsi être extrait de l'intégrale[5]. On obtient alors finalement l'expression :

$$\boxed{[\mathbf{grad}(\mathrm{div}\,\mathbf{u})]^i = \frac{1}{2A} \sum_{\text{triangles (i,j,k) du Voronoï}} (\mathrm{div}\,\mathbf{u})\ \overrightarrow{jk}^{\perp}} \qquad (4.20)$$

où div **u** est donné sur chaque triangle par l'Équation 4.18.

3.5 Mise en œuvre

Une fois encore, les termes qui viennent pondérer les déplacements des particules pour créer **grad**(div **u**) sont purement géométriques. Ils peuvent être précalculés dans la configuration de repos et être considérés comme constants dans l'hypothèse de petites déformations[6]. Ce ne sont plus cette fois-ci de simples scalaires, mais ils peuvent être représentés par une matrice 2×2 symétrique.

Calculer le **grad**(div **u**) revient donc à chaque pas de temps à calculer un produit matrice-vecteur de taille 2 (4 multiplications, 2 additions), et ce pour chaque voisin. C'est plus long que pour le laplacien, mais reste très faible.

3.6 Interprétation

Tout comme celle du laplacien, cette formulation satisfait le principe d'action-réaction. La contribution pour le sommet i d'une demi-arête (j', m) ou (m, k') étant l'opposée de celle fournie au sommet j (même divergence, mais normale au Voronoï orientée en sens opposé).

Cette formule possède une intéressante interprétation. Il faut pour cela réécrire l'expression de la divergence de la Formule 4.18 à l'aide de produits vectoriels. Notons $\mathbf{d}^{ij}$ le *déplacement relatif* du point j par rapport au point i : $\mathbf{d}^{ij} = \mathbf{u}^j - \mathbf{u}^i$.

Encore une fois, l'apparition dans toutes les formules calculées en i du déplacement *relatif* des voisins est normale. Elle s'explique par le fait que les opérateurs sont en quelque sorte calculés dans un repère attaché à la particule i. Seules les variations relatives de position importent. On vérifie ainsi également que lors d'une translation globale de l'objet, ces déplacements relatifs vont s'annuler et ainsi conduire à des forces nulles.

[5] Voir ce que cette hypothèse suppose dans le paragraphe dédié au laplacien Sec. 2.3, page 66.
[6] On se reportera à la discussion de cette condition Sec. 2.4, page 67.

Avec $\mathbf{d}^{ij} = \mathbf{u}^j - \mathbf{u}^i$, l'Équation 4.18 de la divergence peut s'écrire :

$$\begin{aligned}
\operatorname{div}\mathbf{u} &= \frac{1}{2A_{ijk}}\left[\left((u_x^j - u_x^i)\,\overrightarrow{ki}_y + (u_x^k - u_x^i)\,\overrightarrow{ij}_y\right) + \left(-(u_y^j - u_y^i)\,\overrightarrow{ki}_x - (u_y^k - u_y^i)\,\overrightarrow{ij}_x\right)\right] \\
&= \frac{1}{2A_{ijk}}\left[\left(\mathbf{d}_x^{ij}\,\overrightarrow{ki}_y + \mathbf{d}_x^{ik}\,\overrightarrow{ij}_y\right) + \left(-\mathbf{d}_y^{ij}\,\overrightarrow{ki}_x - \mathbf{d}_y^{ik}\,\overrightarrow{ij}_x\right)\right] \\
&= \frac{1}{2A_{ijk}}\|\overrightarrow{ij}\wedge\mathbf{d}^{ik} + \overrightarrow{ki}\wedge\mathbf{d}^{ij}\| \qquad (4.21)
\end{aligned}$$

Les produits vectoriels qui apparaissent sont une facilité d'écriture, obtenue en "plongeant" les vecteurs 2D dans un espace 3D. Les vecteurs considérés sont tous situés dans le plan, la norme n'est qu'une façon de ne prendre que leur unique composante non nulle, celle en z.

L'expression du $\mathbf{grad}(\operatorname{div}\mathbf{u})$ sur le triangle (i,j,k) devient alors :

$$[\mathbf{grad}(\operatorname{div}\mathbf{u})]^i = \frac{1}{4\,A\;\;A_{ijk}}\left(\|\overrightarrow{ij}\wedge\mathbf{d}^{ik} + \overrightarrow{ki}\wedge\mathbf{d}^{ij}\|\right)\;\overrightarrow{jk}^{\perp} \qquad (4.22)$$

À la lumière de cette formulation, le $\mathbf{grad}(\operatorname{div}\mathbf{u})$ peut s'interpréter comme suit : la particule i va, pour chacun des triangles (i,j,k) l'entourant, subir une accélération dirigée selon la normale à l'arête (j,k) du triangle. L'intensité et la direction de cette poussée sont exprimés par la divergence.

Supposons pour simplifier que seul le point j s'est déplacé relativement au point i. La divergence vaut alors

$$\frac{1}{2A_{ijk}}\|\overrightarrow{ki}\wedge\mathbf{d}^{ij}\|$$

Le terme d'aire ainsi que $\overrightarrow{ki}$ étant fixes, la divergence va donc être uniquement déterminée par le déplacement relatif du voisin j.

D'après le produit vectoriel de la formule, c'est en fait la position de j par rapport à une droite Δ parallèle à $\overrightarrow{ki}$ qui importe (Fig 4.7). Si j s'est déplacé le long de cette droite, le produit vectoriel est nul et i ne subira aucune influence de la part de j sur ce triangle.

Par contre, plus j s'éloigne de la droite en s'éloignant de k, plus la divergence est positive et plus i sera donc attiré selon $\overrightarrow{jk}^{\perp}$ vers l'arête (j,k). Inversement, si j s'éloigne de la droite Δ en *allant* vers k, i sera repoussé de l'arête (j,k).

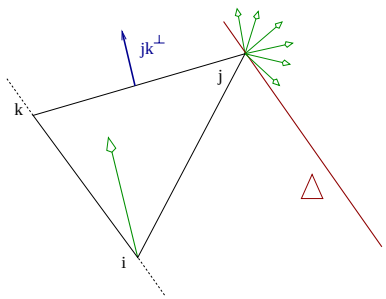


FIG. 4.7: Le point i va se déplacer pour compenser au mieux la variation d'aire due aux déplacements de ses voisins.

On remarque alors que ces déplacements correspondent en fait à la *variation d'aire* du triangle (i,j,k). Lorsque celle-ci a augmenté, i cherche à la réduire en se rapprochant de l'arête (j,k), et inversement.

Noter que la *direction* de déplacement de i n'est pas anodine, un mouvement le long de $\overrightarrow{jk}^{\perp}$ étant en effet le plus efficace pour i pour corriger l'aire du triangle. On retrouve ici l'interprétation comme terme de *préservation de volume* du $\mathbf{grad}(\operatorname{div}\mathbf{u})$ faite dans la section sur l'expression de Navier (Chap. 2).

3.7 Qualité de l'opérateur

Une fois encore, les calculs présentés supposent un champ $\mathbf{u}$ linéaire par morceaux et sont exacts dans ce cas. Voyons, comme nous l'avons déjà fait pour le laplacien, quelles sont les conditions nécessaires pour que le résultat soit également exact pour un champ *quadratique* par morceaux.

Sur chaque triangle, le **grad**(div **u**) est le produit de la divergence sur ce triangle par la normale à l'arête (j,k). La normale reste la même dans le cas quadratique, reste donc à obtenir la même divergence. Celle-ci n'est plus constante, mais linéaire à l'intérieur du triangle et donc le long de l'arête (j',k') sur laquelle on intègre. Pour que le résultat soit le même, il faut et il suffit (toujours grâce à la formule du trapèze) que la divergence ait la même valeur que celle exprimée en (4.18) *au milieu* de l'arête (j',k').

Cette valeur provient de l'expression du gradient, qui doit donc avoir la même valeur ($\frac{1}{h^i}\frac{\overrightarrow{jk}^{\perp}}{\|\overrightarrow{jk}\|}$) au milieu de (j',k') (voir Section 3.2). Le long de la hauteur au triangle en i, la fonction f est maintenant de la forme $f(x) = (\frac{x}{h^i})^2$, si x représente la distance à l'arête (j,k). La valeur obtenue dans le cas linéaire se retrouve en $x = \frac{h^i}{2}$, car $f'(\frac{h^i}{2}) = \frac{1}{h^i}$.

Le gradient aura donc la même valeur au milieu de (j',k') si ce point *coïncide* avec le milieu de la hauteur en i. Ceci n'est vérifié que si le triangle est *isocèle* en i, $\|\overrightarrow{ij}\| = \|\overrightarrow{ik}\|$ (voir Fig. 4.8).

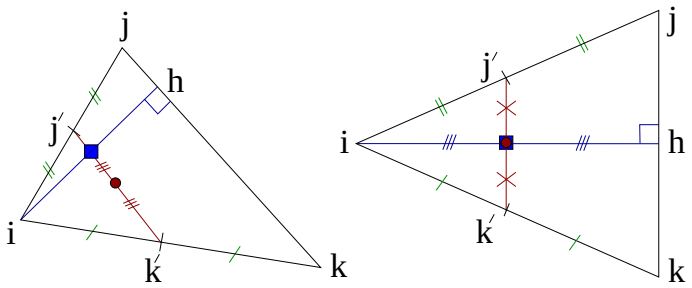


FIG. 4.8: Le milieu de la hauteur *ih* (carré) et celui de l'arête (j',k') (cercle) ne coïncident que dans le cas où le triangle est isocèle en i.

Si l'on souhaite que la propriété soit vérifiée aussi pour le calcul en j et en k, il faut donc que le triangle soit *équilatéral*. C'est la même condition que celle trouvée en Section 2.6 pour le laplacien.

Encore une fois, pour que les opérateurs calculés soient le plus génériques possibles (car s'adaptant à des champs **u** plus complexes), il faut que le maillage soit le plus parfaitement régulier possible (i.e. constitué de triangles équilatéraux). Cette propriété ne pouvant être vérifiée pour un maillage volumique, on cherchera néanmoins à s'en approcher autant que possible.

4 Extension à trois dimensions

4.1 Expression des opérateurs

Dans cette section, nous étendons les résultats précédents au cas 3D. Le principe est exactement le même et l'on va à nouveau appliquer le théorème de Gauss sur la région de Voronoï associée à chaque particule. Le champ déplacement va toujours être considéré comme linéaire par morceaux sur chaque tétraèdre et l'on va pouvoir en exprimer facilement le gradient et la divergence.

La composante normale à la surface du gradient s'obtient par la même formule que précédemment : $(\mathbf{grad}\,\mathbf{u}) \cdot \mathbf{n} = \frac{\mathbf{u}^j - \mathbf{u}^i}{\|\overrightarrow{ij}\|}$. La divergence est un peu plus complexe, mais se déduit comme précédemment des diverses composantes du gradient dans chaque tétraèdre. Nous reviendrons sur son expression dans la Section 5 consacrée à la comparaison avec les éléments finis.

Il faut ensuite intégrer ces valeurs sur le contour de la surface de Voronoï. Celle-ci est un peu plus complexe qu'en 2D, généralement constituée sur chaque tétraèdre de trois surfaces planes, chacune étant orthogonale à l'une des arêtes du tétraèdre. Ces surfaces passent par les milieux des arêtes, les orthocentres des faces et l'orthocentre du tétraèdre (voir Fig. 4.9).

L'expression de la surface de chaque quadrilatère est difficilement exprimable analytiquement. Son calcul ne pose par contre aucun problème, les coordonnées des points qui la compose pouvant être calculées comme nous le verrons Section 4.4.

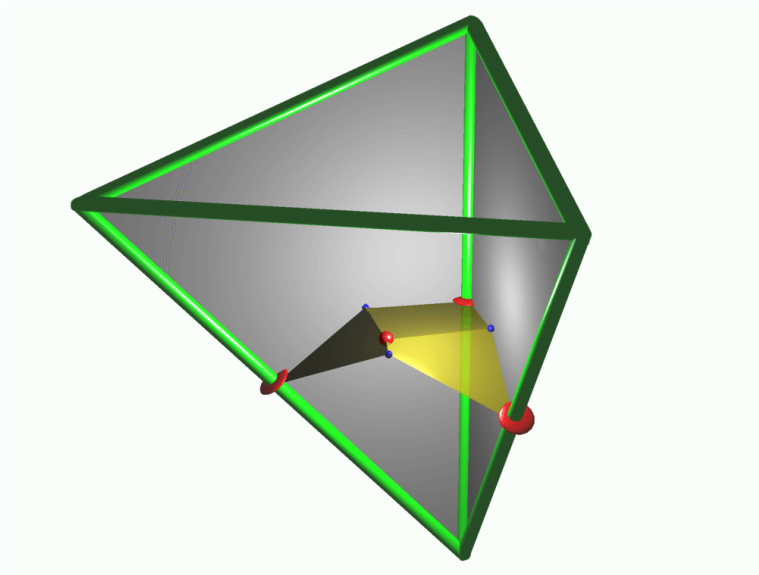


FIG. 4.9: La surface de Voronoï intersecte chaque tétraèdre T en formant 3 polygones orthogonaux aux arêtes. Ils passent par les milieux des arêtes, les orthocentres des faces et l'orthocentre du tétraèdre. On a ici supprimé la face avant du tétraèdre pour montrer les polygones intérieurs.

Une fois calculées les valeurs du gradient et de la divergence ainsi que la surface de Voronoï intersectant chaque tétraèdre, il ne reste plus qu'à sommer ces contributions pour obtenir la valeur de chacun des opérateurs en un point i :

$$(\Delta \mathbf{u})^i = \frac{1}{Vol} \sum_{\text{voisins } j} Surf_{ij} \; \frac{(\mathbf{u}^j - \mathbf{u}^i)}{\|\overrightarrow{ij}\|} \tag{4.23}$$

$$[\mathbf{grad}(\text{div } \mathbf{u})]^i = \frac{1}{Vol} \sum_{\text{tétraèdres voisins } T} Surf_T \;\; (\text{div } \mathbf{u})_T \tag{4.24}$$

où Vol est le volume de la région de Voronoï de i. $Surf_{ij}$ désigne l'aire de la surface de Voronoï propre à l'arête (i, j) (orthogonale à cette arête) et $Surf_T$ celle située *à l'intérieur* du tétraèdre T (représentée Fig. 4.9).

4.2 Expression de la force

L'accélération subie par un point est reliée aux opérateurs que nous venons de décrire par l'équation de Navier :

$$(\rho \; \mathbf{a})^i = \mu \; (\Delta \mathbf{u})^i \; + \; (\mu + \lambda) \; [\mathbf{grad} \; (\text{div } \mathbf{u})]^i \tag{4.25}$$

La *force* subie s'exprime donc par :

$$\mathbf{f} = m \, \frac{1}{\rho} \left[\; \mu \; (\Delta \mathbf{u})^i \; + \; (\mu + \lambda) \; [\mathbf{grad} \; (\text{div } \mathbf{u})]^i \; \right] \tag{4.26}$$

Le terme $\frac{m}{\rho}$ se simplifie en Vol si l'on suppose que la densité est localement approximée par $\rho = \frac{m}{Vol}$. Ce terme Vol disparaît ensuite lors du produit avec les opérateurs exprimés par les Équations 4.23 et 4.24.

La masse m, la masse volumique ρ et le volume de la région de Voronoï se simplifient dans l'expression de la force, prouvant ainsi que l'on ne fait nulle part l'hypothèse d'une densité *uniforme* dans le matériau. On suppose seulement que la masse et la densité d'*une* particule se déduisent l'une de l'autre en utilisant le volume de sa région de Voronoï. La densité peut donc être constante par morceaux sur chacune des régions de Voronoï, voire plus complexe pour peu que volume, masse et densité moyenne locale soient en accord pour chaque particule.

4.3 Mauvaise définition de la région de Voronoï

Sur certains maillages, il peut arriver que la région de Voronoï associée à un point ne soit pas entièrement comprise à l'intérieur du tétraèdre (resp. triangle en 2D), ce qui pose des problèmes.

En 2D cela arrive lorsque l'un des angles du triangle est supérieur à 90°, la longueur de l'arête de Voronoï, donnée par le cotg de l'angle, devenant alors négative : $l = \frac{1}{2}\text{cotg}\,\hat{k}\,\|\overrightarrow{ij}\|$ (voir Eq. 2.2 et Fig. 4.10). Le même problème apparaît en 3D lorsque l'angle diédrique (entre 2 faces d'un tétraèdre) dépasse lui aussi 90°, l'orthocentre du tétraèdre étant alors situé de l'autre côté de la face opposée.

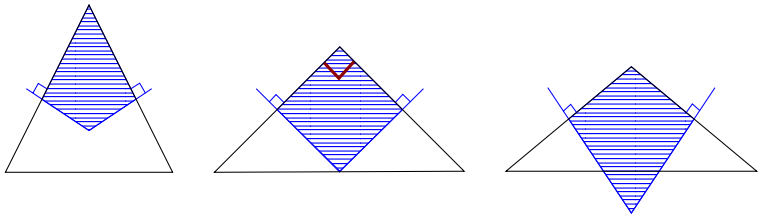
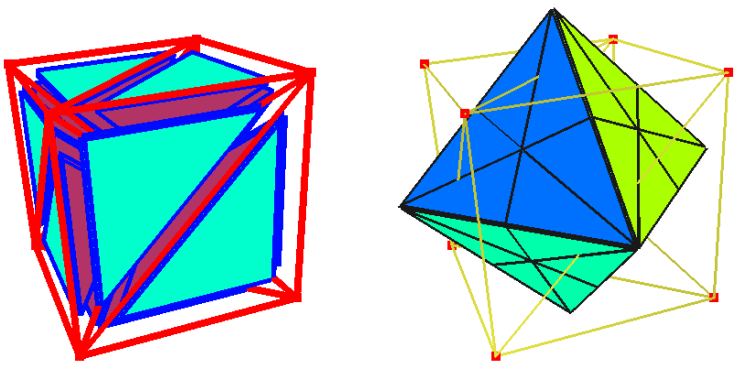


FIG. 4.10: La région de Voronoï peut se trouver à l'extérieur du triangle si l'angle dépasse 90°.

Ce type de maillage est très gênant dans notre cas. Nous avons en effet supposé que la région de Voronoï était bien localisée autour de chaque point pour pouvoir y supposer les opérateurs suffisamment continus et donc approximables par leur valeur moyenne sur la région.

Ces cas d'angles supérieurs à 90° sont rares en 2D où l'on peut rester proche de la triangulation idéale faite de triangles équilatéraux aux angles de 60°. La tétraédrisation d'un simple cube montre qu'il n'en est pas de même en 3D.

Prenons un cube composé de 9 sommets, situés à ses 8 coins et en son centre. Les tétraèdres générés (composés du point central et des triangles composant les faces du cube) possèdent *tous* des angles diédriques supérieurs à 90° (Fig. 4.11a). La région de Voronoï associée au point central est dans ce cas une double pyramide représentée Figure 4.11b.



(a) (b)

FIG. 4.11: Les tétraèdres composant l'objet (a) et la région de Voronoï associée au point central (b).

Dans ce cas particulier, apparaît en plus un second problème qui est que la région de Voronoï d'un point, dépassant des tétraèdres avoisinants, peut se retrouver *à l'extérieur* de la surface de l'objet.

Ces différents problèmes expliquent qu'il faille prendre un soin particulier à la création du maillage et des régions de Voronoï associées.

4.4 Création des régions de Voronoï

Nous avons pour générer les surfaces de Voronoï d'un maillage utilisé `qhull`[7], un logiciel libre développé par la communauté de géométrie algorithmique. `qhull` permet de calculer des diagrammes de Delaunay, et dualement des régions de Voronoï, en dimension arbitraire.

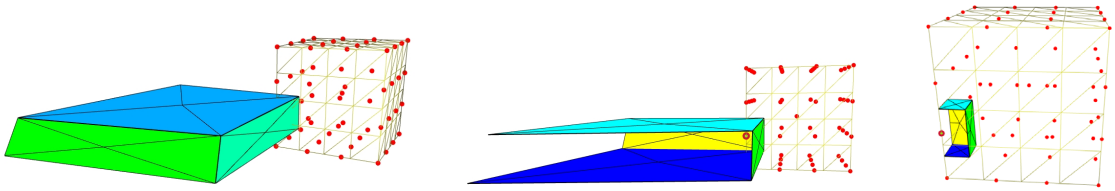
[7] QuickHull a été écrit par *The Geometry Center*, Minneapolis MN. Voir `www.geom.umn.edu/locate/qhull`.

Les régions de Voronoï des points situés *à la surface* de l'objet ont une étendue infinie. Elles comportent dans la description de `qhull` un point particulier, estampillé comme représentant l'infini. Exacte mathématiquement, cette description ne nous suffit néanmoins pas, car nous désirons également connaître la *direction* de l'arête infinie. Aussi avons-nous eu recours à une astuce permettant de ne plus obtenir de points situés à l'infini.

Aux n points composant l'objet dont on cherche les régions de Voronoï, nous en ajoutons 6, disposés en diamant selon les trois axes. Ces points sont suffisamment éloignés de l'objet pour ne pas interférer sur les facettes de Voronoï internes. Ils vont par contre venir *couper* les facettes infinies qui partent des points de la surface, et ainsi permettre de bien délimiter celles-ci. Les facettes infinies sont maintenant celles correspondant à ces 6 points, et nous ne les utiliserons pas.

Une fois les facettes composant les régions de Voronoï obtenues, reste à les traiter. On procède pour cela en plusieurs étapes, illustrées Figure 4.12 pour la région de Voronoï associée à un point situé sur une arête extérieure d'un cube (Fig 4.12a).

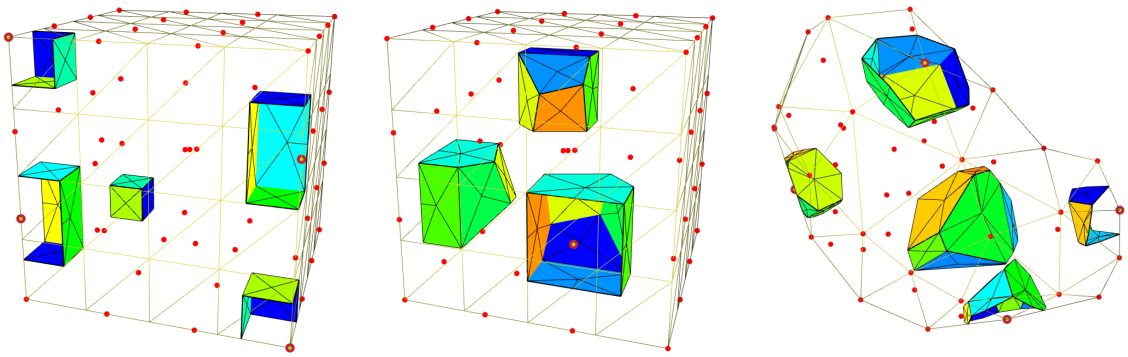
- La première consiste à déterminer quels sont, parmi les points qui composent ces arêtes, ceux situés hors de l'objet. Un test sur la boîte englobante, affiné par un test d'appartenance aux tétraèdres du maillage donne le résultat.
- On peut ensuite supprimer les facettes n'ayant que des points extérieurs (Fig 4.12b). Il faudra néanmoins parfois, dans des configurations pathologiques, vérifier que la facette n'intersecte tout de même pas l'objet bien qu'ayant tous ses points situés au dehors.
- On projette ensuite les points extérieurs *sur* la surface de l'objet, en calculant l'intersection des arêtes avec les faces externes des tétraèdres.
- Reste à fermer les facettes qui avaient une partie extérieure à l'objet en joignant les points que l'on vient de projeter par un chemin sur la surface des tétraèdres (Fig 4.12c).



(a) (b) (c)

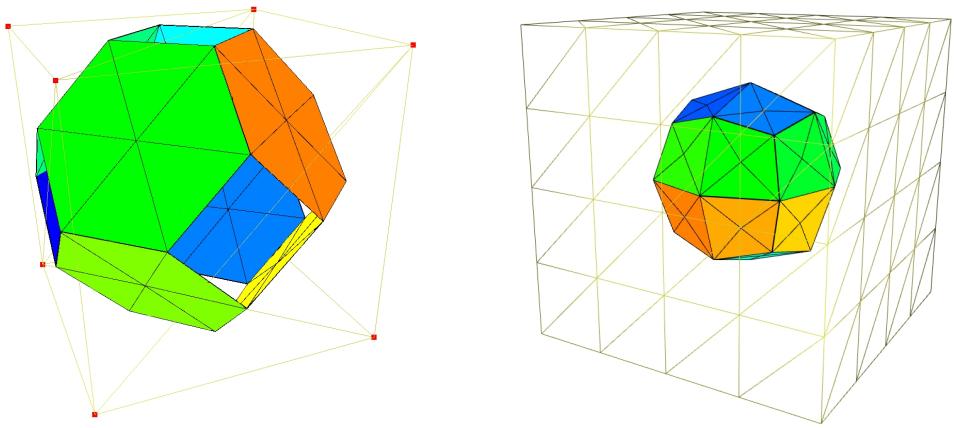
FIG. 4.12: Les arêtes extérieures sont coupées aux limites de l'objet en plusieurs étapes : région de Voronoï complète pour un des sommets du bord (a), suppression des facettes externes (b) et projection sur la surface (c).

Cet algorithme permet l'obtention de régions de Voronoï bien définies et limitées à l'intérieur de l'objet. Les Figures 4.13 et 4.14 montrent quelques régions de Voronoï obtenues sur des maillages de cubes réguliers et sur un maillage de foie beaucoup moins régulier.



(a) (b) (c)

FIG. 4.13: Des exemples de régions de Voronoï obtenus sur différents maillages.



(a) (b)

FIG. 4.14: La région associée au point central d'un cube composé de 9 (a) (comparer avec la Figure 4.11b) et 57 (b) points régulièrement répartis.

Les maillages que nous utiliserons pour représenter les objets seront les duaux des régions de Voronoï obtenues avec `qhull`. Les tétraèdres de Delaunay ont en effet comme arêtes des paires de points partageant une facette de Voronoï. C'est de ces arêtes que nous construirons les relations de voisinage entre points.

Ceci ne suffit pas à éliminer les cas de tétraèdres aux angles supérieurs à 90°, en particulier près du bord de l'objet (comme le montre l'exemple du cube maillé avec 9 points de la Figure 4.14(a)). Néanmoins, dans ces cas, la restriction des facettes à l'intérieur de l'objet que nous opérons limite l'apparition de facettes dans des zones mal définies où les hypothèses faites lors des calculs d'intégrales ne sont plus vérifiées.

5 Comparaison avec les éléments finis

Les calculs développés plus haut sont basés sur des considérations géométriques simples. Il nous est apparu qu'ils présentaient une grande similarité avec une certaine classe de méthodes d'éléments finis. Cette similarité permet de reconsidérer les deux méthodes sous un jour nouveau, comme nous allons le voir dans cette section.

5.1 Éléments finis explicites

La méthodologie des éléments finis a été décrite lors de la présentation des travaux antérieurs (Chap. 1). Pour résumer rapidement, on exprime sur chaque élément du maillage une équation d'équilibre mettant en jeu les déplacements des sommets du maillage. Ces différentes équations sont ensuite rassemblées dans un gros système matriciel prenant en compte les déplacements et les forces subies par tous les nœuds.

Selon les conditions au bord (surface libre, déplacement imposé), qui vont changer au cours de la simulation, la matrice globale change et doit donc être réinversée, ce qui est très long. En profitant astucieusement de ce que peu de modifications sont apportées à la matrice, on peut toutefois rapidement la mettre à jour, comme dans [JP99] (au prix d'une simulation statique et non dynamique).

Cette lenteur, incompatible avec une application temps-réel, nous avait dissuadés d'utiliser des méthodes à base d'éléments finis et conduits à développer les opérateurs différentiels exposés précédemment. La découverte de la méthode des éléments finis *explicites*, qui elle aussi fournit un calcul *local* de l'accélération basé sur les lois de la mécanique des milieux continus, a changé notre vision.

Parmi les deux exemples de la littérature utilisant les éléments finis explicites (Cotin [Cot97] et O'Brien et Hodgins [OH99]), celui de Cotin est le plus proche. Comme nous, il utilise en fait en effet l'équation de Navier (Eq 2.19) tirée de l'utilisation du tenseur infinitésimal de Cauchy et conduisant au calcul des deux opérateurs différentiels vus plus haut. O'Brien utilise quand à lui un tenseur des déformations non linéaire (Green-Lagrange[8]) conduisant à des équations plus complexes.

[8] Voir la comparaison de ces deux formalismes dans le Chapitre 2.

La comparaison de nos opérateurs avec les calculs mis en œuvre dans la méthode des masses-tenseurs de Cotin a montré que les deux approches étaient très similaires. Ce résultat était prévisible de par la similarité des deux méthodes : utilisation de l'équation de Navier, éléments tétraédriques sur lesquels on interpole linéairement la fonction.

Les deux méthodes sont néanmoins différentes, d'une part par la manière dont elles arrivent à l'expression des opérateurs ce qui est déjà intéressant en soi, mais également de par l'expression même des opérateurs, celle-ci différant légèrement d'une méthode à l'autre.

5.2 Réécriture des masses-tenseurs

Utilisant rigoureusement le formalisme des éléments finis, Cotin arrive après plusieurs étapes à l'expression par une matrice 12×12 des équations régissant le comportement d'*un* tétraèdre T . Cette matrice de rigidité K relie les 24 inconnues du système *sur un tétraèdre* que sont les *forces* d'un côté et les *déplacements* en chacun des 4 sommets de l'autre (chacun ayant 3 composantes) :

$$\mathbf{f} = K\ \mathbf{u} \tag{4.27}$$

K s'exprime en fonction des paramètres λ et μ du matériau ainsi qu'en fonction des vecteurs α^i, liés à la géométrie du tétraèdre et introduits plus haut (voir Sec. 3.2). Un vecteur α^i est rappelons-le orthogonal à la face opposée au sommet i et de norme inverse de la hauteur h_i du tétraèdre au point i. C'est en d'autres termes le gradient de la fonction de base linéaire associée au point i.

La matrice K d'un tétraèdre s'écrit alors comme :

$$K = Vol_T\ (\ \Gamma^T\ L\ \ \Gamma\ )$$

où Vol_T est le volume du tétraèdre T et Γ et L des matrices définies comme suit :

$$\Gamma^T = \begin{pmatrix}
\alpha_{1x} & \alpha_{1y} & \alpha_{1z} & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_{2x} & \alpha_{2y} & \alpha_{2z} & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_{3x} & \alpha_{3y} & \alpha_{3z} & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_{4x} & \alpha_{4y} & \alpha_{4z} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{1x} & \alpha_{1y} & \alpha_{1z} & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{2x} & \alpha_{2y} & \alpha_{2z} & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{3x} & \alpha_{3y} & \alpha_{3z} & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha_{4x} & \alpha_{4y} & \alpha_{4z} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{1x} & \alpha_{1y} & \alpha_{1z} \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{2x} & \alpha_{2y} & \alpha_{2z} \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{3x} & \alpha_{3y} & \alpha_{3z} \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_{4x} & \alpha_{4y} & \alpha_{4z}
\end{pmatrix}$$

$$L = \begin{pmatrix}
2\mu+\lambda & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & \lambda \\
0 & \mu & 0 & \mu & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \mu & 0 & 0 & 0 & \mu & 0 & 0 \\
0 & \mu & 0 & \mu & 0 & 0 & 0 & 0 & 0 \\
\lambda & 0 & 0 & 0 & 2\mu+\lambda & 0 & 0 & 0 & \lambda \\
0 & 0 & 0 & 0 & 0 & \mu & 0 & \mu & 0 \\
0 & 0 & \mu & 0 & 0 & 0 & \mu & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mu & 0 & \mu & 0 \\
\lambda & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 2\mu+\lambda
\end{pmatrix}$$

On peut isoler la contribution des deux opérateurs dans cette matrice en jouant sur les paramètres λ et μ : $\mu = 0, \lambda = 1$ donne directement la matrice correspondant à $\mathbf{grad}\ (\text{div}\ \mathbf{u})$ alors que $\mu = 1, \lambda = 0$ donne $\Delta\mathbf{u}$ après soustraction de la matrice précédente.

Influence de deux particules

L'étape suivante consiste à isoler, dans les deux matrices 12×12 précédentes, l'expression de l'influence d'un sommet j sur un sommet i donné. Cette influence se caractérise pour chaque opérateur par une matrice 3×3, dénommée *tenseur* par Cotin. Notons T_{Δ}^{ij} et $T_{\mathbf{grad}(div)}^{ij}$ les tenseurs associés aux deux opérateurs.

En utilisant *Maple*, on parvient à avoir l'expression suivante :

$$T^{ij}_{\mathbf{grad}(div)} = Vol_T \quad (\alpha_i \times {\alpha_j}^T) \tag{4.28}$$

où $\times$ représente le produit matriciel de ces deux vecteurs, dont l'un est transposé.

On peut noter que l'influence de i sur j et celle de j sur i s'expriment par des tenseurs transposés (ce qui peut être utile pour économiser la mémoire et les calculs) :

$$T^{ji}_{\mathbf{grad}(div)} = (T^{ij}_{\mathbf{grad}(div)})^T$$

La matrice T^{ij}_{Δ} a, quand à elle, sur chacun des termes de sa diagonale la valeur $Vol_T \quad (\alpha^i \cdot \alpha^j)$ (produit scalaire cette fois). C'est une matrice antisymétrique, et le terme non diagonal placé à la ligne l et à la colonne c de la matrice triangulaire supérieure ($c > l$) est donné par l'expression :

$$\begin{aligned} T^{ij}_{\Delta}[l,c] &= Vol_T \quad (\alpha^i_c \alpha^j_l - \alpha^i_l \alpha^j_c) \\ T^{ij}_{\Delta}[c,l] &= -\, T^{ij}_{\Delta}[l,c] \\ T^{ij}_{\Delta}[l,l] &= Vol_T \quad (\alpha^i \cdot \alpha^j) \end{aligned} \tag{4.29}$$

Ces tenseurs permettent de calculer directement la *force* subie par un point en fonction des déplacements de ses voisins. Ils s'appliquent aux relations à l'intérieur d'*un* tétraèdre. Pour calculer la valeur d'un des opérateurs, une particule i devra *sommer les influences de tous les tétraèdres dont elle est un sommet*.

Au point i, pour la contribution d'*un* tétraèdre T, les opérateurs s'écrivent :

$$\boxed{(\Delta \mathbf{u})^i = -\sum_{sommet\ j\ \in\ 1..4} \frac{Vol_T}{Vol}\, T^{ij}_{\Delta}\, \mathbf{u}^j} \tag{4.30}$$

$$\boxed{[\mathbf{grad}(\mathrm{div}\ \mathbf{u})]^i = -\sum_{sommet\ j\ \in\ 1..4} \frac{Vol_T}{Vol}\, T^{ij}_{\mathbf{grad}(div)}\, \mathbf{u}^j} \tag{4.31}$$

Vol_T désigne le volume du tétraèdre T considéré alors que Vol est le volume de *toute la région* de Voronoï associée au point i.

Noter que ces sommes se font sur *tous* les sommets du tétraèdre et comprennent donc le point i où l'on calcule l'opérateur. Une expression en fonction des seuls déplacements relatifs $\mathbf{d}^{ij} = \mathbf{u}^j - \mathbf{u}^i$ comme celle présentée en 2D est possible mais moins élégamment concise.

Dans le cas particulier de points disposés sur une grille rectangulaire régulière, on retrouve avec ces formules l'expression des différences finies à un coefficient près [LT94].

Il est intéressant de noter que contrairement aux opérateurs issus de l'utilisation de Gauss, ces expressions ne vérifient pas a priori le principe d'action-réaction. Il n'y a en effet aucune raison avec ces expressions que $T^{ij}\, \mathbf{u}^j + T^{ii}\, \mathbf{u}^i = {T^{ij}}^T\, \mathbf{u}^i + T^{jj}\, \mathbf{u}^j$.

Dans l'hypothèse de faibles déplacements, ces tenseurs peuvent tout comme précédemment être précalculés dans la position de repos. Le calcul de la force subie par un point se ramène donc au calcul de n produits d'une matrice 3×3 par un vecteur, n étant le nombre de voisins du point.

Ces tenseurs étant exprimés, nous allons maintenant les comparer avec les résultats obtenus par la méthode basée sur les régions de Voronoï des Sections 2 et 3.

5.3 Comparaison en 2D

En 2D, les formulations résultant de l'utilisation des Voronoï sont très similaires à celles obtenues par les éléments finis explicites.

La matrice représentant l'opérateur **grad** (div **u**) avec les Voronoï est en effet la *même* que celle trouvée par l'Équation 4.31. Un jeu de réécriture permet de simplifier l'Équation 4.18 de la divergence pour faire apparaître le tenseur $T^{ij}_{\mathbf{grad}(div)} = Surf\ (\alpha_i \cdot \alpha_j{}^T)$ dans l'expression de la dépendance des points i et j ($Surf$ est l'équivalent en 2D du terme Vol de la 3D).

L'action de l'opérateur $\Delta\mathbf{u}$ entre les points i et j est par contre dans notre cas exprimée par un simple scalaire lié aux *cotg* des angles (Eq. 4.8). Là encore, une réécriture permet de faire apparaître les vecteur α^i et α^j. On peut alors montrer que le coefficient de l'Équation 4.8 est en pratique *égal* au terme diagonal du tenseur, $Surf\ (\alpha^i \cdot \alpha^j)$.

Lorsqu'on utilise ce tenseur, on multiplie donc bien le déplacement par le scalaire $Surf\ (\alpha^i \cdot \alpha^j)$ comme on le fait en appliquant la formule issue de Voronoï. On ajoute par contre à ce résultat le produit du déplacement par les termes non diagonaux de la matrice.

On peut en fait montrer que cette multiplication supplémentaire est *sans effet*. Les termes additionnels obtenus s'annulent en effet deux à deux, de part et d'autre d'une arête liant i et un voisin.

Ceci s'explique par le fait que les termes hors-diagonale $\alpha^i_c\alpha^j_l - \alpha^i_l\alpha^j_c$ sont en fait des produits vectoriels de deux vecteurs correspondant aux inverses des hauteurs du triangle (voir la définition des α^i, Eq. 4.15). Le terme hors-diagonale, une fois multiplié par $Surf$ est donc une constante, qui vaut plus ou moins $\frac{1}{2}$ de part et d'autre de l'arête, et qui va donc disparaître lorsqu'on va sommer l'influence de tous les voisins.

En plus de complexifier les calculs, ces termes additionnels inutiles vont donc probablement ajouter un bruit dans la simulation dû aux imprécisions numériques qui vont faire que ces termes ne vont pas forcément s'éliminer deux à deux une fois calculés.

Les deux méthodes (basées sur les éléments finis ou sur les régions de Voronoï et le théorème de Gauss) sont donc *équivalentes* en 2D. Ce résultat est en soi une contribution de cette thèse. Il permet en effet d'affirmer que les éléments finis se basent *implicitement* sur une décomposition en régions de Voronoï du plan. Plus que les régions de Voronoï, on s'aperçoit que c'est en fait la limitation de la zone d'influence d'un point au *milieu* de chaque arête entre lui et ses voisins qui est au cœur des formules établies.

La remarque de la Section 4.2 s'applique donc également aux éléments finis. Pour peu que masse et densité soient choisis d'après la région de Voronoï de chaque point, les éléments finis ne font alors pas l'hypothèse d'une densité uniforme dans *tout* le matériau, mais seulement de masses calculées en fonction de la densité locale et des volumes propres de chaque particule.

5.4 Comparaison en 3D

La comparaison des deux méthodes en trois dimensions est plus difficile car nous n'avons pas d'expression analytique de la surface des facettes de Voronoï.

Détaillons le calcul de l'opérateur **grad** (div **u**) présenté en Section 4.1, en cherchant à utiliser les notations des éléments finis. La valeur de div **u** sur un tétraèdre donné s'obtient alors à partir de la formule du gradient d'une champ scalaire f :

$$\mathbf{grad}\ f = \sum_{sommet\ i \in 1..4} \alpha^i\ f^i \tag{4.32}$$

On applique cette formule à des champs f valant successivement chacune des trois composantes du champ **u**. La divergence est formée de la somme des $j^{ème}$ composantes du gradient de la $j^{ème}$ composante du **u** (div $\mathbf{u} = u_{x,x} + u_{y,y} + u_{z,z}$). On a donc :

$$\begin{aligned} \text{div}\ \mathbf{u} &= (\sum_{i \in 1..4} \alpha^i_x\ u^i_x) + (\sum_{i \in 1..4} \alpha^i_y\ u^i_y) + (\sum_{i \in 1..4} \alpha^i_z\ u^i_z) \\ &= \sum_{i \in 1..4} (\sum_{j \in x,y,z} \alpha^i_j\ u^i_j) \\ &= \sum_{i \in 1..4} (\alpha^i \cdot \mathbf{u}^i) \end{aligned} \tag{4.33}$$

La contribution du déplacement $\mathbf{u}^i$ d'un sommet i à la divergence de l'élément s'écrit donc comme un simple produit scalaire avec le vecteur α^i correspondant.

Si on reporte cette expression dans le calcul de l'intégrale sur la surface des facettes de Voronoï se trouvant à l'intérieur d'un tétraèdre T, on obtient (Eq 4.13) :

$$\begin{aligned}
\int_{V\,\cap\,T} [\mathbf{grad}\,(\mathrm{div}\,\mathbf{u})]\,dV &= \int_{\partial V\,\cap\,T} (\mathrm{div}\,\mathbf{u})\,\mathbf{n}\,dS \\
Vol\,[\mathbf{grad}\,(\mathrm{div}\,\mathbf{u})] &= \sum_{facettes\ de\ \partial V\,\cap\,T} (\mathrm{div}\,\mathbf{u})\,\mathbf{n}\,dS \\
&= (\mathrm{div}\,\mathbf{u}) \sum_{facettes\ f\ de\ \partial V\,\cap\,T} \mathbf{n}^f\,S^f \\
&= (\sum_{i\,\in\,1..4} (\alpha^i \cdot \mathbf{u}^i)) \sum_{facettes\ f\ de\ \partial V\,\cap\,T} \mathbf{n}^f\,S^f
\end{aligned} \tag{4.34}$$

où $\mathbf{n}^f$ et S^f représentent respectivement la normale et la surface des différentes facettes composant la surface de Voronoï de *i et* intersectant le tétraèdre que l'on est en train de considérer.

L'influence de j sur i (i et j étant des sommets du tétraèdre, éventuellement identiques) s'exprime donc par :

$$\frac{1}{Vol}\,(\alpha^j \cdot \mathbf{u}^j) \sum_{facettes\ f\ de\ \partial V\,\cap\,T} \mathbf{n}^f\,S^f \tag{4.35}$$

Cette expression est celle de l'*opérateur* **grad** (div **u**). Passer à celle de la *force* va permettre de comparer les deux approches et se fait, nous l'avons vu en Section 4.2, en multipliant par Vol, qui disparaît donc de l'expression.

Sur chaque tétraèdre, le tenseur T^{ij} indiquant dans l'opérateur l'influence du déplacement du voisin j sur i peut s'exprimer en isolant les termes qui viennent pondérer $\mathbf{u}^j$. Notons donc β^j le vecteur $S^f\,\mathbf{n}^f$ correspondant à l'influence de la facette f. On a alors :

$$T^{ij} = \sum_{facettes\ f\ de\ \partial V\,\cap\,T} \begin{pmatrix} \beta^j_x\,\alpha^j \\ \beta^j_y\,\alpha^j \\ \beta^j_z\,\alpha^j \end{pmatrix} \tag{4.36}$$

Cette expression peut maintenant être comparée avec celle donnée par la méthode des élément finis explicites (voir Eq. 4.28) :

$$T^{ij}_{EF} = \begin{pmatrix} -(Vol_T)\,\alpha^i_x\,\alpha^j \\ -(Vol_T)\,\alpha^i_y\,\alpha^j \\ -(Vol_T)\,\alpha^i_z\,\alpha^j \end{pmatrix} \tag{4.37}$$

Comparaison des expressions

La différence entre les deux méthodes est donc toute entière contenue dans la différence entre les vecteurs β^j et $Vol_T\ \alpha^i$. Plus précisément, le terme $Vol_T\ \alpha^i$ qui intervient dans la formulation par éléments finis peut se réécrire, en introduisant la surface S^i et la normale $\mathbf{n}^i$ de la face opposée à i du tétraèdre T :

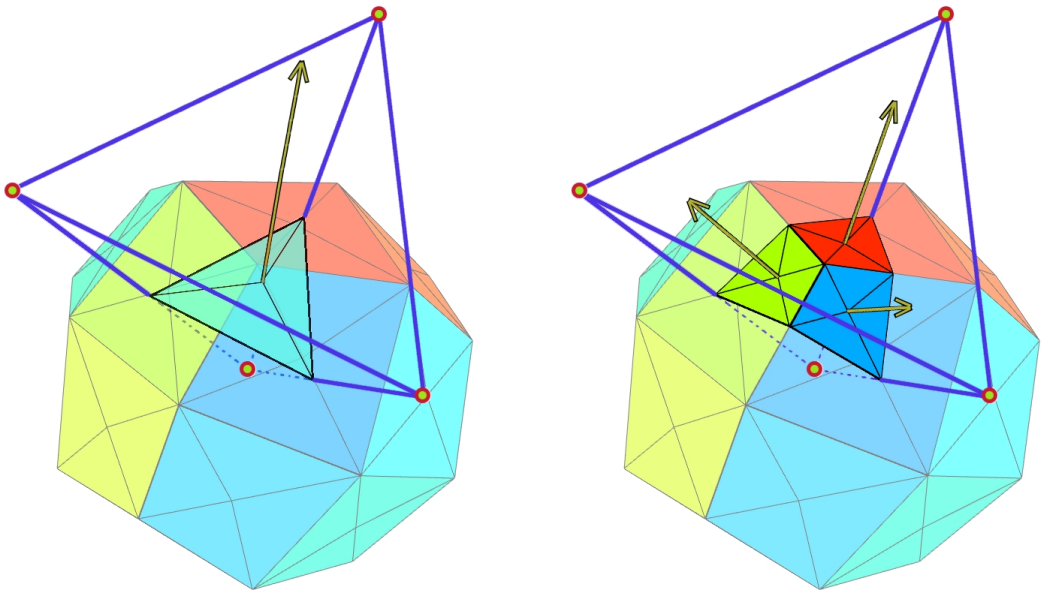
$$Vol_T\ \alpha^i = \frac{1}{3}(S^i\,h^i)\,\alpha^i = -\frac{1}{3}S^i\mathbf{n}^i$$

le terme h^i, hauteur du tétraèdre au point i, intervenant dans l'expression de α^i (4.15) se simplifiant. Le signe "$-$" provient de ce que α^i est orienté vers i alors que $\mathbf{n}^i$ est dirigé vers l'extérieur de T.

On est donc amené à comparer la normale $\mathbf{n}^i$ à la face du tétraèdre, pondérée par $\frac{1}{3}$ de sa surface (Fig. 4.15a) et les normales des intersections des facettes de Voronoï avec T, pondérées par leurs propres surfaces (Fig 4.15b). Noter qu'on a représenté sur la Figure 4.15a non pas la la face du tétraèdre, mais son *homothétie* passant par les milieux des arêtes (de surface 4 fois moindre), afin de mieux pouvoir comparer les formulations.

Ces deux pondérations sont intuitivement très proches. Les différentes facettes composant le Voronoï étant toutes plus ou moins orientée autour de $\mathbf{n}^i$ et la somme de leurs surfaces étant très proche de $\frac{1}{3}S^i$.

En pratique, sur un exemple régulier comme le cube, on constate que les valeurs numériques des coefficients sont très proches, et diffèrent en tout cas du même ordre de grandeur sur l'ensemble des tétraèdres du maillage.



(a) (b)

FIG. 4.15: La différence de calcul de l'opérateur **grad** div entre les deux méthodes tient au calcul pondéré d'une normale à la *face* du tétraèdre pour les éléments finis (a) qui est remplacé dans notre approche par la somme pondérée des trois normales des facettes de Voronoï (b).

Notons que dans le cas particulier où l'intersection des facettes de Voronoï avec T est formée de trois triangles, les deux vecteurs sont *identiques* à un coefficient près. La somme, pondérée par leurs surfaces, des normales de ces trois triangles est en effet égale, d'après une propriété générale des tétraèdres, au produit de la surface de la quatrième face du tétraèdre (généré par les trois triangles) par sa normale. Puisque cette quatrième face passe par les milieux des arêtes, le théorème de Thales stipule que sa surface est $\frac{1}{4}\,S^i$. Les deux vecteurs sont donc égaux à un coefficient $\frac{3}{4}$ près dans ce cas.

Ce cas apparaît toutefois rarement. Les facettes de Voronoï intérieures à T étant généralement des quadrilatères, voire des formes plus complexes sur des maillages non réguliers. Il peut également arriver que plus (ou moins) de trois facettes distinctes soient intersectées dans un tétraèdre donné.

En pratique, nous avons obtenu ce facteur $\frac{3}{4}$ entre les deux méthodes sur l'exemple d'un cube régulier, pour les points situés aux coins du cube.

6 Création d'un modèle hybride

6.1 Inspiration

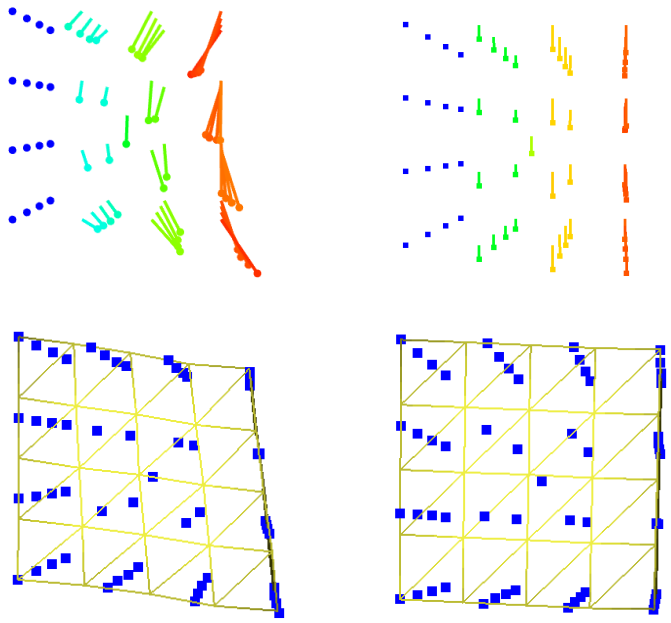
Nous avons vu que, dans la méthode des Voronoï, l'opérateur laplacien s'exprime toujours comme un scalaire. Le tenseur 2×2 T_Δ qui représente le laplacien dans la méthode des éléments finis en 2D est en fait équivalent au scalaire obtenu avec Voronoï, les termes hors-diagonaux s'éliminant deux à deux.

Les termes hors-diagonaux du tenseur T_Δ ne s'éliminent toutefois pas en 3D. Aussi avons-nous voulu essayer, par *analogie* avec la méthode utilisant les Voronoï, de ne prendre en compte que les termes diagonaux de T_Δ.

6.2 Comportement

Simplifier ainsi le tenseur change beaucoup le mouvement obtenu lors de la simulation. Les nombreuses ondes qui parcouraient le matériau avec le tenseur de Cauchy sont réduites à une seule déformation principale si l'on n'utilise plus que les termes diagonaux. Le comportement très "gélatineux" du tenseur de Cauchy, fait de nombreuses oscillations superposées, disparaît alors et laisse place à un matériau plus rigide (à constantes rhéologiques égales) qui retrouve plus rapidement et plus simplement sa position d'équilibre.

Sur l'exemple d'un cube soumis à la gravité, les positions extrêmes atteintes avec les deux modèles sont bien différentes (Fig. 4.16).



(a) (b)

FIG. 4.16: Les positions extrêmes atteintes par un cube soumis à l'action de la gravité en utilisant les éléments finis explicites et le tenseur de Cauchy (a) et la version hybride que nous proposons par analogie avec les résultats de la méthode de Voronoï (b). La première ligne montre le vecteur déplacement de chaque point.

On retiendra que cet opérateur hybride donne des animations très stables qui, bien que moins complexes qu'avec Cauchy, permettent d'obtenir un résultat visuel convaincant.

Noter que l'article tiré des calculs de ce chapitre [DDCB00] faisait l'amalgame entre cette méthode hybride et l'utilisation du tenseur de Cauchy classique, car nous pensions alors que les termes hors-diagonaux s'annulaient également deux à deux en 3D.

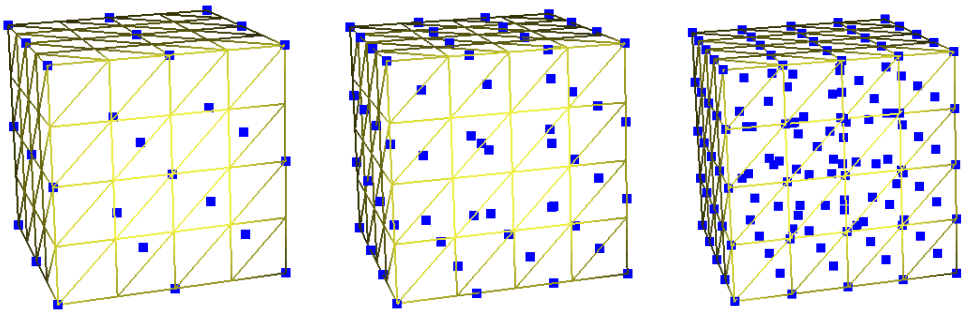
7 Comparaison des différentes formulations

Nous allons ici comparer différentes méthodes de simulation et en particulier mesurer leur indépendance à la discrétisation. Le but est d'obtenir des courbes aussi similaires que possible sur des simulations utilisant différentes résolutions, pour pouvoir développer des applications multirésolution.

7.1 Protocole de test

L'exemple d'école que nous avons choisi est un cube homogène de 10 centimètres de côté et d'une masse d'un kilo. Une de ses faces est maintenue immobile, collée verticalement contre un mur. On mesure la position d'un des coins du cube lorsque celui-ci oscille sous l'effet de la gravité, en l'absence de toute force de frottement (quelques images de l'animation ont été présentées au Chapitre 2, page 38). La position de départ est non déformée, sans vitesse initiale.

Cette animation a été faite avec trois discrétisations différentes du cube comportant respectivement 27, 57 et 135 points (soit 48, 122 et 448 tétraèdres) (voir Fig. 4.17).



(a) (b) (c)

FIG. 4.17: Trois résolutions différentes vont être simulées. Le niveau 0 a 27 points (a), le niveau 1 en a 57 (b) et le niveau 2, 135 (c).

7.2 Les différents modèles comparés

Nous allons comparer les méthodes d'éléments finis explicites utilisant les tenseurs de Cauchy et Green-Lagrange (décrits en détails dans le Chapitre 2 consacré à l'élasticité linéaire). Nous comparerons ces résultats à ceux obtenus avec la méthode basée sur les Voronoï décrite dans ce chapitre. Nous testerons également le modèle hybride détaillé dans la section précédente, issu d'un mélange des opérateurs de différences finies et des concepts issus de la méthode basée sur Voronoï.

Nous essaierons également plusieurs méthodes de masses-ressorts (voir page 14) avec différents calculs de la raideur des ressorts : raideur constante, proportionnelle à la longueur au repos du ressort ou calculée d'après la méthode Van Gelder [Gel98].

7.3 Résultats

Dans toutes les courbes, le temps, en abscisse, est exprimé en secondes et le déplacement vertical du point est exprimé en mètres sur l'ordonnée. Les trois courbes représentent à chaque fois, de haut en bas, les niveaux 0, 1 et 2.

Les différentes raideurs des simulations ont été choisies pour obtenir un mouvement de même amplitude dans toutes les animations ($\mu = 7000$ et $\lambda = 20000$ pour Cauchy et Green-Lagrange, $\mu = 2800$ et $\lambda = 8000$ pour la version hybride, $k = 0.05 * 7000$ pour les masses-ressorts à raideur constante, $k = l_0 * 7000$ pour la version proportionnelle et $k = \frac{20000\ V}{{l_0}^2}$ pour Van Gelder). La fréquence d'échantillonage temporelle est de 800 Hz pour les niveaux 0 et 1 et de 1600 Hz pour le niveau 2, l'intégration utilisant la méthode d'Euler modifiée [9].

Le tenseur de Cauchy

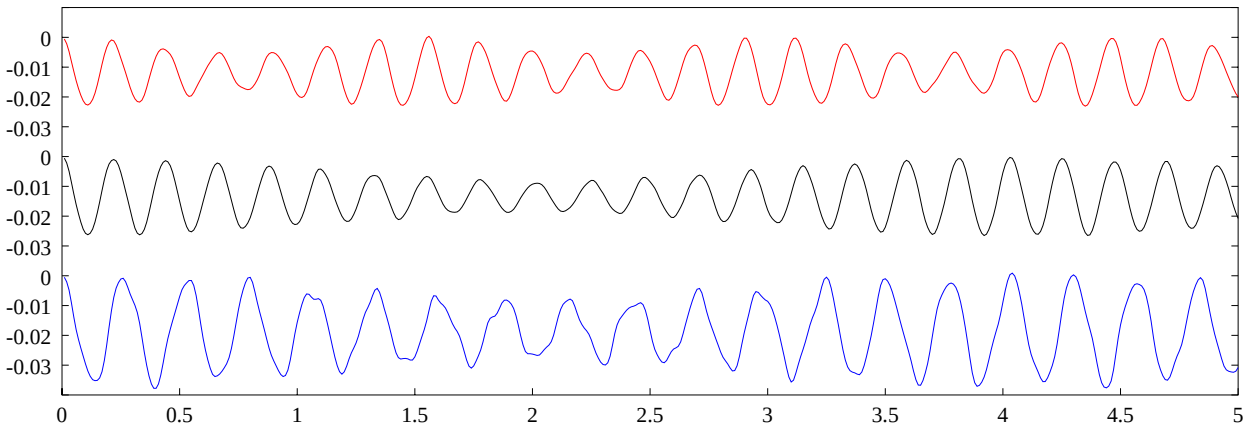


FIG. 4.18: Simulation avec le tenseur de Cauchy.

[9] Voir Annexe B.

Le tenseur de Cauchy donne des résultats sinusoïdaux d'amplitude variable. Une analyse sur une plus longue période montre qu'ils sont en fait très réguliers, la variation d'amplitude ayant elle aussi une fréquence propre.

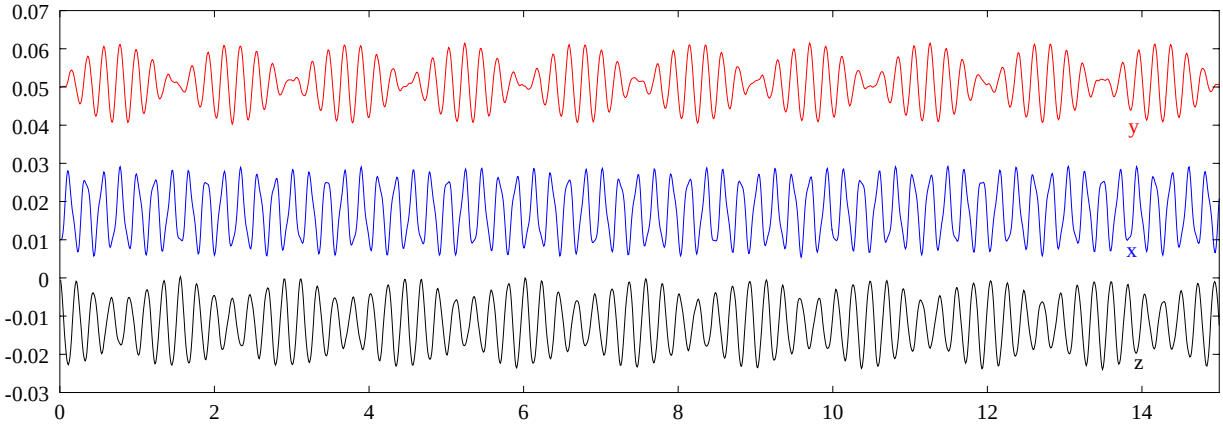


FIG. 4.19: Tenseur de Cauchy, sur 15 secondes de simulation. La position en x,y et z du coin du cube fait apparaître un mouvement régulier.

L'animation montre bien l'expression de la combinaison de deux phénomènes : une oscillation régulière, perturbée par un mode de résonance de plus basse fréquence qui module les oscillations. Bien que de *forme* similaire, il est à noter que la fréquence et l'amplitude des oscillations ne sont néanmoins pas rigoureusement les mêmes pour les trois résolutions utilisées.

Augmenter la raideur du matériau pour avoir de plus petites déformations et donc rester plus proche du domaine de validité de ce tenseur ne change rien. Même avec des oscillations de quelques millimètres d'amplitude, les différentes résolutions ont encore des amplitudes et des fréquences de mouvement variant dans un rapport de 1 à 2.

La méthode basée sur Gauss et Voronoï

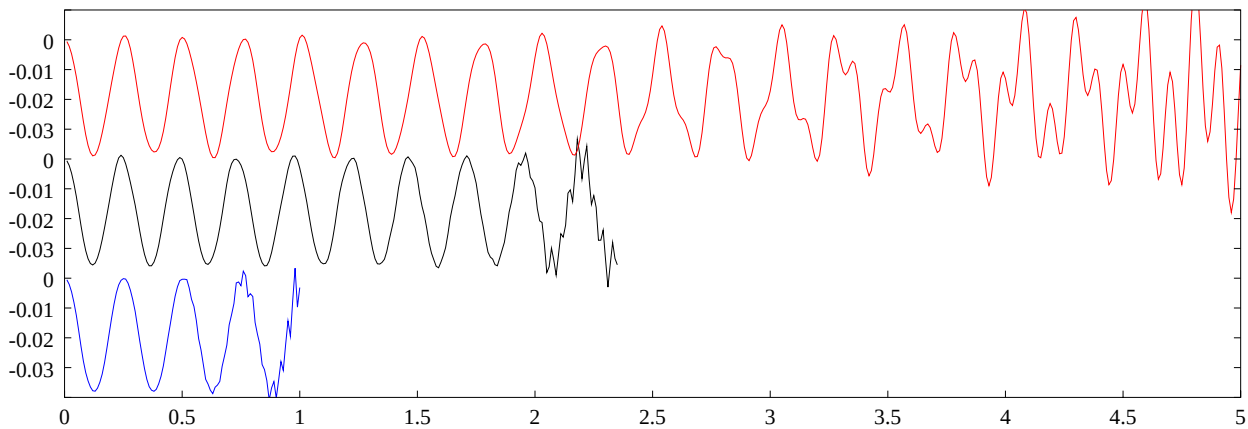


FIG. 4.20: Simulation avec la méthode basée sur les régions de Voronoï et le théorème de Gauss.

Les résultats obtenus avec cette méthode sont bons, mais instables. Les oscillations sont propres, d'amplitudes identiques et de fréquences très proches. Malheureusement, des instabilités viennent perturber la simulation, d'autant plus rapidement que l'on utilise beaucoup de particules. Diminuer le pas de temps d'intégration ne retarde pas l'apparition de ce problème.

Nous avons pu déterminer que ces perturbations étaient liées à l'opérateur **grad** (div **u**), le laplacien ayant quant à lui un comportement extrêmement stable. Cette différence de comportement est assez étonnante au regard de la comparaison faite en Section 5.4 entre cette formulation et celle des éléments finis avec le tenseur de Cauchy (qui elle n'est pas ainsi perturbée).

Ce terme en **grad** (div **u**) exprime rappelons-le la recherche d'une préservation de volume du matériau. Nous avons donc réalisé une simulation n'utilisant *que* ce terme et dans laquelle on vient appuyer avec un outil virtuel sur un cube. Les forces qui sont calculées paraissent alors très intuitives (voir Fig. 4.21), repoussant les points vers l'extérieur de l'objet, dans les directions laissées libres.

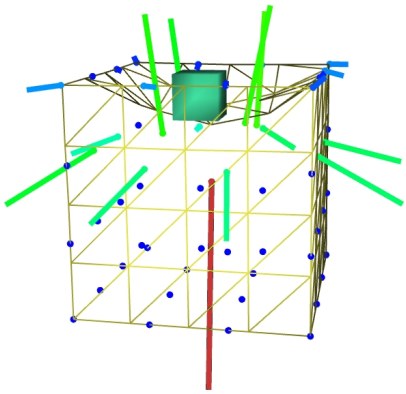
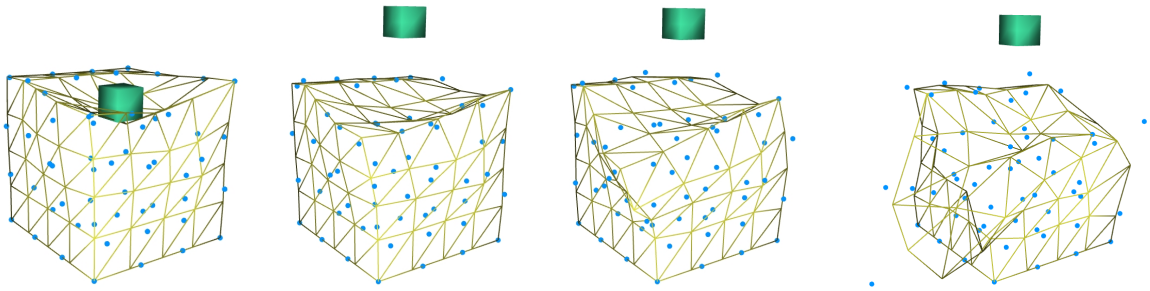


FIG. 4.21: Les forces créées par l'opérateur **grad** (div **u**) sont bien celles que l'on attend d'un opérateur assurant une préservation de volume.

Le problème est que lorsque l'on relâche la pression exercée sur l'objet, celui-ci ne parvient pas à conserver une position d'équilibre stable. Il continue de se déformer, très lentement : de faible amplitude au départ, les forces générées continuent à déformer l'objet, l'entraînant dans un cercle vicieux vers des positions plus déformées et des forces plus importantes (voir Fig. 4.22).



(a) (b) (c) (d)

FIG. 4.22: On constate une lente divergence de la simulation utilisant l'opérateur **grad** (div **u**) issu de la méthode de Voronoï.

Une animation faite dans des conditions rigoureusement identiques en utilisant la méthode du tenseur de Cauchy ne fait pas apparaître de telle divergence. L'objet garde sa forme déformée, compatible avec une préservation de volume, arrivant rapidement à une position stable où les forces s'annulent.

Bien qu'extrêmement proches mathématiquement et géométriquement (voir la comparaison Sec. 5.4), les deux méthodes simulent donc un comportement différent, la divergence du **grad** (div **u**) dans la méthode basée sur les Voronoï n'ayant pas trouvé à ce jour d'explication mathématique.

Les modèles masses-ressorts

Nous avons essayé différents modèles masses-ressorts en choisissant la raideur des ressorts selon divers critères. Celle-ci était soit constante pour tous les ressorts, quelle que soit la résolution employée, soit proportionnelle à leur longueur à vide pour prendre en compte les variations de longueur entre les résolutions. Enfin, nous avons également implémenté le calcul préconisé par Van Gelder [Gel98] (voir Chapitre 1, Sec. 11.4), et qui est supposé être ce qui peut se rapprocher le plus des éléments finis dans un formalisme de masses-ressorts.

Les résultats sont globalement moins réguliers qu'avec les modèles continus, l'amplitude des oscillations variant en particulier davantage. C'est particulièrement le cas lorsque la raideur est choisie comme inversement

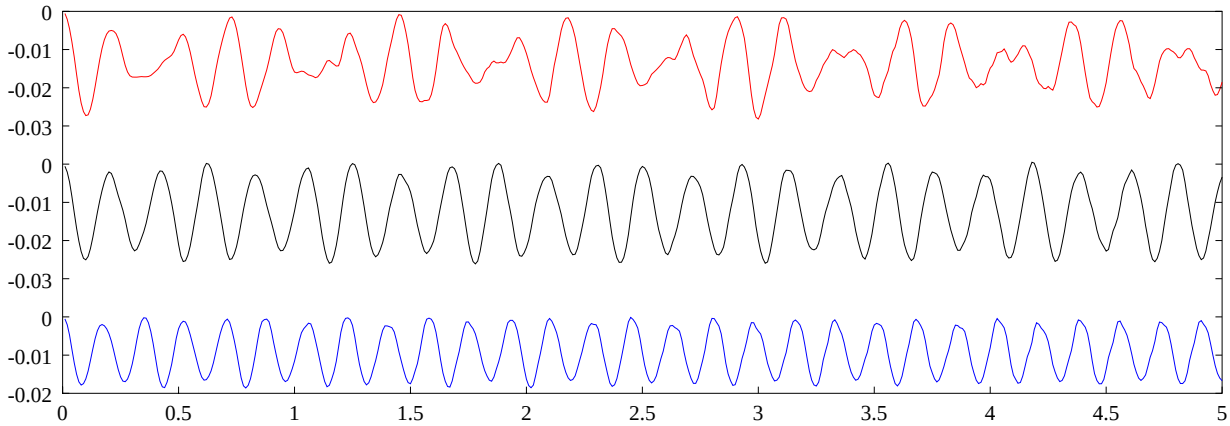


FIG. 4.23: Simulation avec des masses-ressorts. La raideur est une même constante pour tous les ressorts.

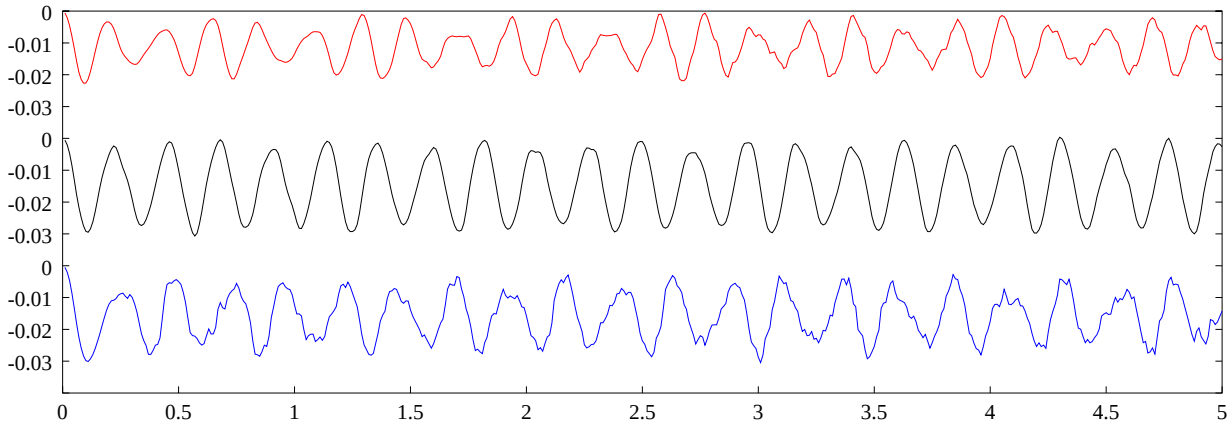


FIG. 4.24: Simulation avec des masses-ressorts. La raideur est proportionnelle à la longueur à vide du ressort.

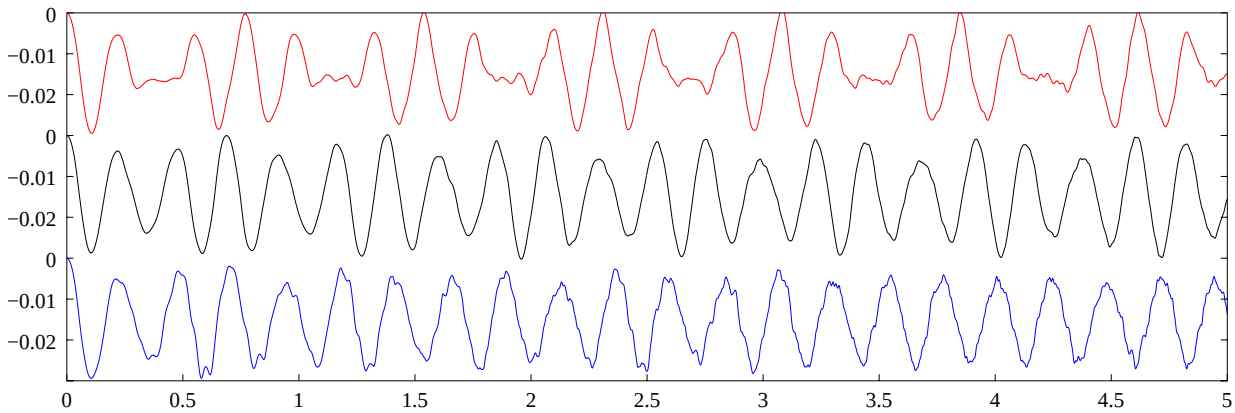


FIG. 4.25: Simulation avec des masses-ressorts. La raideur est *inversement* proportionnelle à la longueur à vide du ressort. Noter que pour obtenir des courbes d'amplitudes similaires, on a dû utiliser des raideurs de respectivement 20, 13 et 6 pour les résolutions 0, 1 et 2.

proportionnelle à la longueur à vide des ressorts (voir Figure 4.25). Il faut alors régler le coefficient de raideur pour obtenir des courbes de même amplitude car sinon celle-ci est moins forte pour les résolutions élevées (ce qui est normal puisque la raideur augmente lorsqu'on raffine davantage). On pourrait vouloir compenser cette modification de raideur par un coefficient proportionnel à la discrétisation, mais cela revient ici à vouloir éliminer le facteur $\frac{1}{l_0}$, ce qui ramènerait aux résultats de la Figure 4.23.

Il faut ici exclure de l'analyse les résultats peu probants obtenus sur le niveau 0 (niveau le plus grossier, courbes du haut), probablement dus au manque de cohésion dans le matériau lié au faible nombre de ressorts.

Pour les niveaux 1 et 2, on obtient d'assez jolies sinusoïdes, bien que moins régulières que précédemment. Les raideurs constantes donnent un résultat lisse, mais très dépendant de la résolution (comparer les fréquences

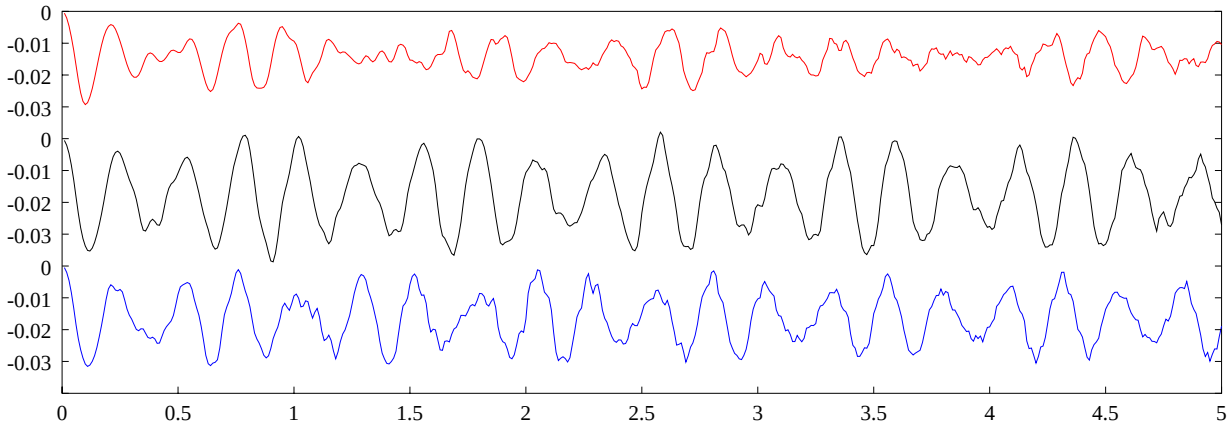


FIG. 4.26: Simulation avec des masses-ressorts. La raideur est calculée selon la formule de Van Gelder [Gel98].

Figure 4.23).

Les simulations avec d'autres raideurs font apparaître au niveau 2 (courbes du bas) des oscillations parasites qui viennent dégrader la forme sinusoïdale du signal. À noter que bien que très perturbée, l'animation obtenue avec Van Gelder (Fig. 4.26) au niveau 2 a une fréquence d'oscillation très proche de celle du niveau 1. C'est beaucoup plus loin d'être la cas avec les autres calculs de raideurs.

Ces résultats sont relativement meilleurs que ce que nous imaginions. Bien qu'assez perturbées, les sinusoïdes obtenues sont relativement proches, ce qui rehausse l'intérêt que l'on peut avoir pour les masses-ressorts. Il faut néanmoins relativiser ces propos en rappelant que l'exemple du cube, de forme très régulière se prête bien aux masses-ressorts, ceux-ci pouvant être bien alignés à l'intérieur du matériau. Si l'allure des mouvements (leur amplitude en particulier) est assez proche, le comportement dynamique est toutefois loin d'être identique entre les résolutions, ce qui est gênant si l'on souhaite utiliser plusieurs résolutions ensemble. Reste également à régler le problème des dissipations internes d'énergie, l'introduction de ressorts amortis risquant de grandement perturber le comportement dynamique.

La version hybride du tenseur de Cauchy

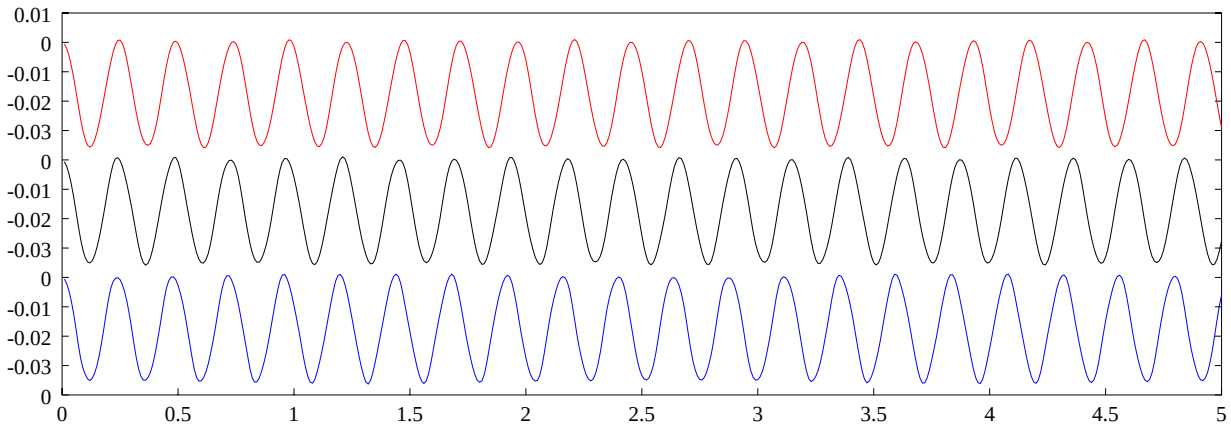


FIG. 4.27: La version hybride du tenseur de Cauchy (le laplacien est calculé à l'aide d'un simple scalaire).

La version hybride du tenseur de Cauchy (voir Sec. 6) donne de bien meilleurs résultats. Les courbes obtenues sont des sinusoïdes parfaites, qui ont toutes exactement la même amplitude. Il subsiste une légère différence en fréquence, celle-ci augmentant légèrement avec la résolution (on mesure 4.08, 4.12 et 4.16 Hz respectivement aux niveaux 0, 1 et 2).

La simulation est très stable, et nous avons simulé 5 minutes de cette animation sans constater de divergence de la méthode. L'existence d'une position de référence contribue probablement beaucoup à cette stabilité numérique.

L'utilisation de termes de frottements internes permet de diminuer l'amplitude des oscillations. Nous vérifions sur les courbes de la Figure 4.28 que ces forces dissipatives ne viennent pas perturber le bon comportement multirésolution de l'opérateur. 3.

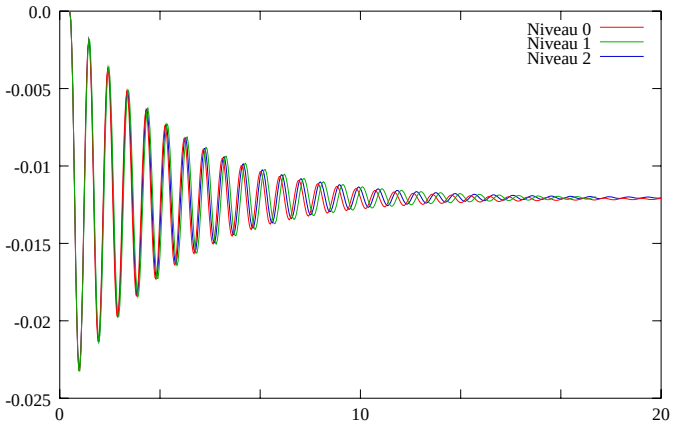


FIG. 4.28: L'introduction de frottements internes n'altère pas le comportement multirésolution de l'opérateur hybride.

Ces courbes sont à comparer avec celles obtenues avec celles obtenues avec notre premier modèle (Chapitre 3, Figure 3.3b, page 49). Plusieurs dizaines de secondes de simulation sont ici possibles sans appartition de nette divergence entre les résolutions, contre quelques secondes seulement avec la méthode du Chapitre 3.

Ce modèle se prête donc très bien à une utilisation multirésolution. C'est au prix d'une simplification du mouvement généré (le cube oscille en pratique selon un mouvement de cisaillement, sans vraiment se plier) que l'on va pouvoir espérer mélanger les résolutions.

Le tenseur de Green-Lagrange

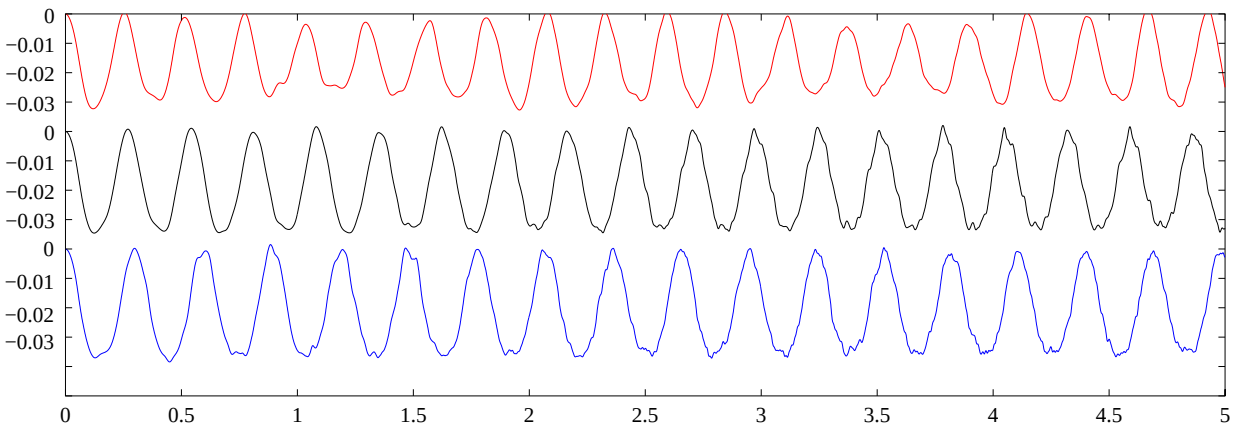


FIG. 4.29: Simulation avec le tenseur de Green-Lagrange.

Le modèle de Green-Lagrange donne également de très bons résultats. Les trois résolutions produisent des oscillations similaires, en amplitude et en fréquence. Les courbes ne sont pas aussi lisses qu'avec la méthode hybride, perturbées par de petites oscillations parasites. Les fréquences varient également davantage d'une résolution à l'autre (3.86, 3.67 et 3.4 Hz respectivement pour les niveaux 0, 1 et 2). Rappelons toutefois que ces résultats sont obtenus en simulant un comportement de matériau beaucoup plus complexe qu'avec la méthode hybride. L'ajout de forces dissipatives ne dégrade pas les qualités multirésolution de la méthode, comme le montre la Figure 4.30.

Les premiers essais réalisés avec cette méthode ont révélé un comportement numériquement instable, dû en réalité à une erreur dans le code. Cette divergence après quelques secondes d'animation nous avait poussé à chercher d'autres méthodes et à développer la méthode hybride présentée plus haut (voir Section 6).

Bien que plus chère numériquement (d'un facteur 2 environ par rapport à la méthode hybride ou à celle basée sur Cauchy), ce formalisme présente de nombreux avantages qui en font un modèle de choix pour une

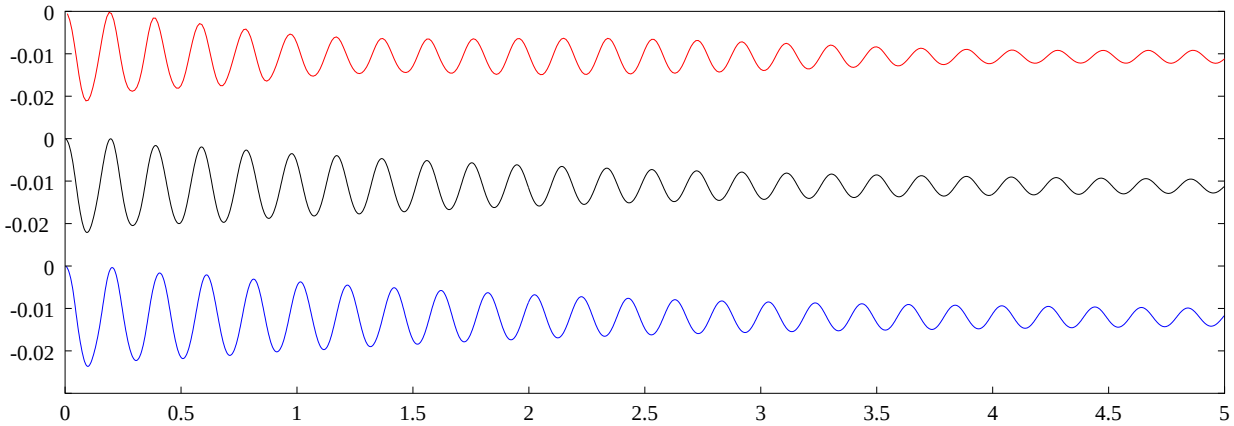


FIG. 4.30: Tenseur de Green-Lagrange avec ajout de forces dissipatives internes.

simulation multirésolution. Cette méthode, comparée aux autres méthodes de ce comparatif, produit en effet le mouvement du cube le plus réaliste lors de l'animation, celui-ci se pliant naturellement (voir page 38). Elle permet de plus d'effectuer une rotation globale de l'objet sans qu'il y ait création de forces (à l'inverse de celles utilisant le tenseur de Cauchy, voir Chapitre 2), ce qui génère une animation bien plus réaliste.

Malgré la légère dégradation des courbes constatée sur notre exemple, cette méthode permet une cohabitation de plusieurs niveaux de résolutions différentes. Le réalisme des animations produites en fait donc un très bon choix de modèle pour une animation d'objets déformables, utilisant ou non la multirésolution [10].

8 Conclusion

Nous avons dans ce chapitre présenté un mode de calcul original des opérateurs différentiels utilisés dans l'équation de Navier. Basées sur le théorème de Gauss et la découpe de l'espace en régions de Voronoï, ces formulations font intervenir les propriétés géométriques du maillage, et donc la discrétisation de l'objet.

Si l'on reste dans le cadre de faibles déformations, on ne recalcule pas les différents coefficients au cours du temps, les supposant suffisamment constants. Le calcul des forces appliquées en chaque point est alors une simple multiplication de coefficients géométriques par les déplacements des points voisins, ce qui est rapide.

Nous nous sommes attachés à interpréter à chaque fois la signification pratique des mathématiques intervenant dans les formules. Ces explications géométriques et intuitives permettent de mieux saisir le comportement de l'objet que l'on va simuler.

Nous avons pu rapprocher nos formulations de celles obtenues par la méthode des éléments finis. Identiques en 2D, les deux méthodes se révèlent très proches en 3D. On notera toutefois que nous avons pu mettre en évidence l'inutilité en 2D des coefficients hors-diagonaux du tenseur représentant le laplacien en éléments finis, sources de bruit et d'instabilités. Ce résultat et le rapprochement des deux méthodes contribue à offrir un point de vue nouveau sur la méthode des éléments finis.

Les subtiles différences entre les deux modèles suffisent à introduire des instabilités dans la version 3D des opérateurs, ce qui fait qu'il ne semble pas y avoir d'état d'équilibre dans la méthode basée sur Voronoï et que l'on assiste à une lente divergence de la simulation.

En rapprochant les deux méthodes, nous avons défini un tenseur hybride, version scalaire de celui de Cauchy, qui simplifie les réponses aux déformations du matériau.

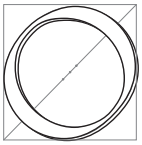
La comparaison générale de différentes méthodes de simulation faite à la fin de ce chapitre (Cauchy, Green-Lagrange, hybride, Voronoï et masses-ressorts) permet de mettre en évidence la grande indépendance à la résolution de ce nouveau modèle hybride basé sur le tenseur de Cauchy *scalaire*. Bien qu'un peu moins parfait, le modèle de Green-Lagrange se prête lui aussi bien à la simulation multirésolution.

Le chapitre suivant va décrire comment plusieurs maillages vont pouvoir être utilisés dans un algorithme multirésolution qui s'appuiera sur la description des opérateurs différentiels que nous venons de proposer.

[10] L'annexe C décrit en détails l'implémentation de ce formalisme dans le cadre des éléments finis *explicites*.

CHAPITRE 5

Modèle multirésolution hiérarchique



N A ÉTUDIÉ dans le précédent chapitre la formulation d'opérateurs différentiels efficaces et indépendants de la résolution. Leur utilisation dans un algorithme multirésolution va consister à utiliser simultanément plusieurs maillages de résolutions différentes. L'introduction de points *fantômes* permettra à ces maillages de communiquer et il ne restera qu'à choisir quelle résolution est la mieux adaptée à chaque zone de l'objet. Les méthodes décrites dans ce chapitre ont fait l'objet d'une publication à *Computer Animation* [DDCB00] (avec le modèle hybride, voir Chapitre 4) et à Siggraph 2001[DDCB01] (avec le tenseur de Green-Lagrange et l'intégration semi implicite).

1 Idée générale

L'algorithme expliqué dans ce chapitre s'appuie sur une découpe de l'objet en un maillage de tétraèdres. Ce choix profite de la simplicité et de la souplesse de ces maillages, tétraèdres. Ce choix profite de la simplicité et de la souplesse de ces maillages, seuls capables de mailler tout objet délimité par une surface triangulée. La notion de voisinage entre points se déduit du maillage, les points voisins étant ceux reliés par une arête de tétraèdre.

Nous supposerons ici que chaque point est capable de calculer la force qui lui est appliquée en fonction des seuls déplacements de ses voisins. Outre les méthodes d'éléments finis explicites ou celles basées sur les régions de Voronoï vues dans le chapitre précédent, on pourrait donc utiliser l'algorithme que nous allons décrire avec les opérateurs de la première méthode de cette thèse (Chapitre 3), voire avec une méthode de type masses-ressorts si l'on est capable de la rendre suffisamment indépendante du maillage.

De la qualité du maillage dépend grandement celle des résultats numériques obtenus, en particulier avec la méthode des éléments finis. Le tétraèdre idéal est régulier, toutes ses arêtes ayant la même longueur. Il est en pratique impossible de mailler un volume en utilisant uniquement de tels tétraèdres réguliers, mais on peut avec une méthode de relaxation essayer de s'en approcher.

L'idée générale de la méthode présentée ici est qu'il serait à la fois coûteux en temps de calcul et insatisfaisant au niveau de la simulation numérique d'essayer d'*adapter* le maillage en fonction de la simulation.

Il faudrait en effet subdiviser les tétraèdres existants lorsque l'on cherche à ajouter des points [HPH96, OH99]. Il n'existe malheureusement pas de découpe *régulière* d'un tétraèdre, et l'on est forcément amené à dégrader la qualité du maillage (en terme d'angles diédriques et de rapport des longueurs d'arêtes). Des études ont été faites sur le type de subdivision possible et l'optimisation de la position des points ajoutés [PLL98]. Il

n'existe pas de solution optimale et l'on ne peut assurer la qualité du maillage généré.

On pourrait envisager de remailler complètement l'objet lorsque les tétraèdres sont de trop mauvaise qualité, mais cette opération est coûteuse et incompatible avec le temps-réel. Il faut également s'assurer que le maillage reste *conforme* à chaque instant. Enfin, on a peu de chances de retrouver un maillage de grande qualité lorsque l'on revient dans la position non déformée, suite aux regroupements successifs.

D'ailleurs, la simplification du maillage peut elle-même poser problème. Regrouper localement plusieurs tétraèdres en un seul peut amener à des maillages de mauvaise qualité dans lesquels des tétraèdres de tailles très différentes cohabitent et ne peuvent se simplifier. On peut éviter ce problème en imposant que seuls les tétraèdres issus d'une *même* subdivision puissent se regrouper, ce qui restreint la souplesse du procédé. Conserver un historique des subdivisions peut alors permettre de revenir au maillage initial, supposé de bonne qualité.

En pratique, on constate une divergence de la simulation, liée à la dégradation du maillage, après deux ou trois subdivisions successives des tétraèdres. C'est cette constatation qui nous a décidés à développer une méthode à base de maillages fixes, de grande qualité, pouvant ensembles approximer à plusieurs échelles un même mouvement.

2 Cohabitation de différents maillages

2.1 Création des maillages

Les différents maillages utilisés sont totalement indépendants les uns des autres et n'ont en particulier pas à *partager* de sommets. Ils représentent le même objet à des résolutions différentes, leur création pouvant comporter une phase d'optimisation visant à rendre les tétraèdres aussi réguliers que possible.

Nous utiliserons dans nos simulations entre 2 et 7 maillages différents pour un même objet. Il est à noter que quelques dizaines de points suffisent à générer le mouvement général d'un objet. On n'utilisera les maillages fins que pour simuler le mouvement de détail de petites zones, souvent proches de l'outil.

En pratique, nous avons utilisé les logiciels `qslim` [GH97] et `GHS3D` [SIM] pour créer nos maillages. L'objet initial est donné par sa représentation surfacique détaillée, sous forme d'une surface triangulée. `qslim` permet de simplifier ce maillage pour obtenir des représentations avec plus ou moins de triangles du même objet. Une option du programme permet d'imposer la création de triangles aussi réguliers que possible.

Ces différents maillages surfaciques sont ensuite maillés volumiquement par `GHS3D` qui infère la taille à donner aux tétraèdres internes de celle des triangles surfaciques. Le maillage volumique comprend une optimisation de la position des points *internes* visant à égaliser la longueur des arêtes (voir Fig. 5.1). On dispose après maillage de statistiques sur la qualité des tétraèdres créés.

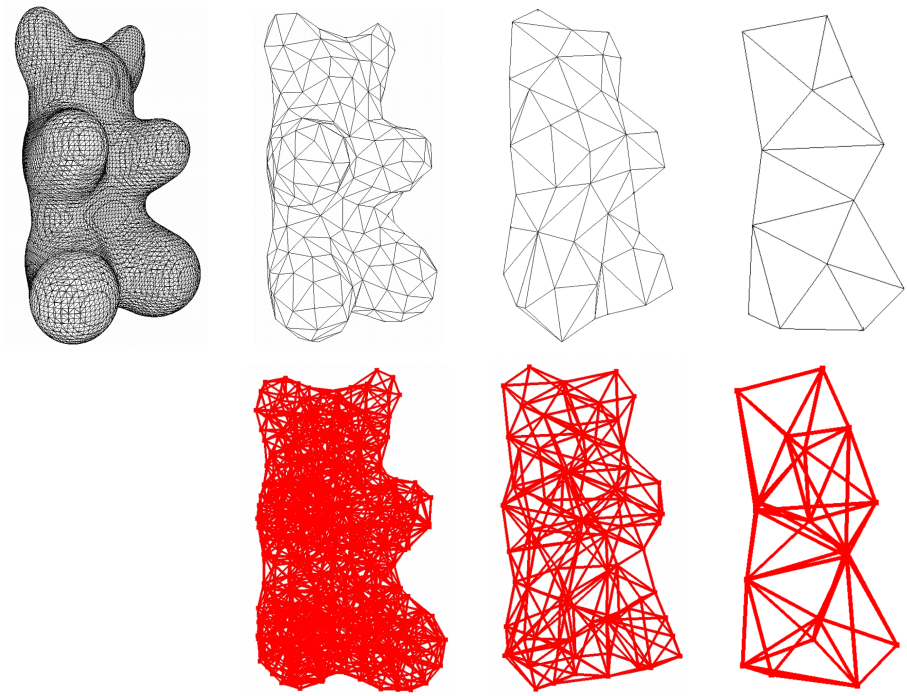
Il peut arriver qu'une différence de quelques triangles sur la surface conduise à des maillages volumiques de qualités assez différentes. Aussi avons-nous utilisé un script qui, une fois le nombre approximatif de tétraèdres désirés donné, simplifie le maillage surfacique de l'objet pour obtenir plus ou moins de triangles et ne conserve parmi les maillages volumiques résultants, que celui correspondant aux meilleures statistiques.

2.2 Interface entre deux maillages

Supposons dans un premier temps que notre objet soit maillé avec deux maillages différents, l'un fin et l'autre plus grossier. Imaginons de plus qu'une moitié de l'objet doive être simulée avec le maillage fin, l'autre pouvant se contenter des calculs plus approximatifs du maillage grossier. La simulation de chacune des deux parties se fait très bien en utilisant les algorithmes et les opérateurs vus précédemment, et il ne reste qu'à gérer l'interface entre les deux maillages pour que l'objet forme un tout et ne soit pas la simple juxtaposition de deux simulations.

Pour ce faire, nous utilisons une technique qui peut être rapprochée de la méthode de décomposition de domaine[1]. Les deux maillages vont légèrement se superposer et certains de leurs sommets, que nous désignerons par le terme de *fantômes*, serviront alors à faire l'interface entre les deux maillages.

[1] Voir `www.ddm.org` pour de plus amples informations.



a b c

FIG. 5.1: Le maillage triangulaire original (34700 triangles) est dégradé à différents degrés (ici 550, 130 et 30 triangles), puis maillé volumiquement (1874, 172 et 28 tétraèdres, ligne du bas).

Pour cohabiter, les deux maillages ont seulement besoin de connaître la valeur du champ *déplacement* dans la zone située à l'interface. C'est en effet la seule donnée nécessaire aux points pour calculer leur mouvement. Les points fantômes d'un maillage porteront cette information de déplacement. Ils la déduiront de l'autre maillage et la communiqueront à leurs voisins du même maillage, servant ainsi d'interface.

2.3 Les points fantômes

Chaque point d'un maillage pourra avoir l'un des statuts suivants :

- Un point *actif* calculera la force qu'il subit à chaque pas de temps en fonction des déplacements de ses voisins. Il intégrera cette force pour trouver sa nouvelle position et donc son nouveau déplacement.
- Un point *inactif* ne sera simplement pas simulé.
- Entre les deux, nous trouverons les points *fantômes*. Ceux-ci ne calculent pas de force, mais auront un déplacement actualisé en fonction de celui du maillage actif de la même zone.

Un point actif aura donc pour voisins d'autres points actifs (aux déplacements réactualisés) ou des points fantômes (aux déplacements interpolés), qui tous lui indiqueront quelle est la déformation locale dans la zone qui l'entoure. Ce point réagira en sommant les influences de ses différents voisins, prenant ainsi, en particulier, en compte le déplacement de la zone d'interface (voir Fig. 5.2).

Un point fantôme déduira son déplacement de celui du tétraèdre du maillage actif dans lequel il est situé. On ne peut en effet avoir de meilleure estimation de ce qui ce passe en ce point qu'en utilisant le tétraèdre qui le contient car c'est lui qui représente cette partie de l'espace pour ce maillage. Le point fantôme utilise ses coordonnées barycentriques dans ce tétraèdre pour calculer la valeur de son propre déplacement en fonction des valeurs du déplacement aux quatre sommets du tétraèdre englobant (les points Q_i de la Figure 5.2 pour l'exemple du point F).

Une moyenne pondérée par les coefficients barycentriques des déplacements est équivalente à une interpolation linéaire de la valeur du champ à l'intérieur du tétraèdre. Cette interpolation est cohérente avec la méthode de calcul choisie pour les opérateurs différentiels, qui elle aussi suppose que le champ est linéaire par morceaux sur chaque élément. On n'ajoute donc pas ici d'hypothèses de continuité, se contentant d'être en accord avec les hypothèses faites précédemment.

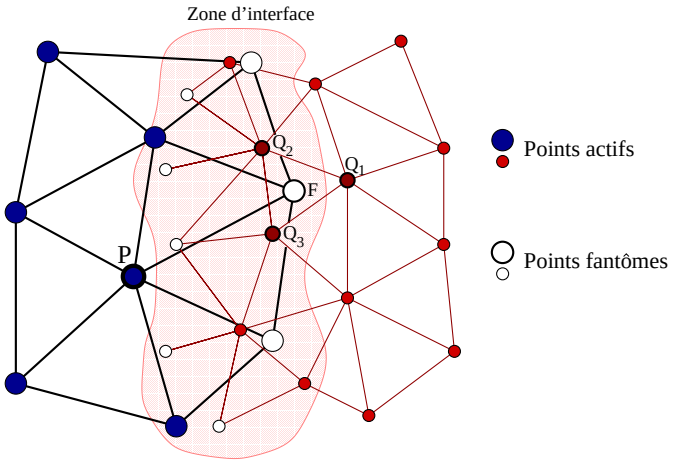


FIG. 5.2: Le voisin F de P est fantôme et sert à P à savoir ce qui se passe à sa droite. Il interpole son déplacement d'après celui des points Q_i du maillage fin.

En pratique, le point fantôme aura évalué ses coordonnées barycentriques dans la position de repos et les utilisera comme poids durant la simulation. L'évaluation du déplacement d'un point fantôme ne nécessitera donc que quatre multiplications scalaire-vecteur et trois additions de vecteurs, ce qui est beaucoup plus rapide que l'évaluation et l'intégration de la force subie que réalise un point actif.

Il est normal de considérer ces coefficients pondérateurs comme constants au cours de la simulation. On peut en effet voir le point fantôme comme "pris" dans un tétraèdre gélatineux, dont il va suivre les déformations. Ses coordonnées intrinsèques à l'intérieur de ce tétraèdre ne varient donc pas au cours de la simulation.

Il peut arriver, au bord de l'objet, qu'un point fantôme d'un maillage fin ne soit pas à l'intérieur d'un tétraèdre du maillage grossier. Nous le lierons alors avec le tétraèdre le plus proche, avec une de ses coordonnées barycentrique négative, de sorte qu'il extrapolera néanmoins le déplacement du tétraèdre.

Ce que nous venons de décrire pour deux maillages peut se généraliser, à condition de bien prendre soin à l'apparition des zones d'interface. Il faudra en particulier veiller à ce que les points actifs aient bien des voisins actifs ou fantômes et que les fantômes soient capables d'interpoler leur déplacement.

Les conditions et la mise à jour de la structure lors du changement de statut d'un point sont l'objet de la section suivante, qui va permettre de choisir localement les maillages les mieux adaptés au cours de la simulation.

3 Adaptation de la simulation

3.1 Structure de données hiérarchique

Les sommets des différents maillages vont être reliés, dans la structure de données, avec les tétraèdres des deux maillages de résolutions directement inférieure et supérieure. Chaque point stockera son tétraèdre parent (maillage plus grossier) et fils (maillage fin), ainsi que ses coordonnées barycentriques dans ces deux tétraèdres. Il pourra ainsi, s'il devient fantôme, déduire son déplacement de celui des autres maillages (Fig. 5.3).

Chaque point stockera également une liste de points *fils*, définis comme étant ceux du maillage immédiatement inférieur (plus fin) situés à l'intérieur de sa région de Voronoï, définie par les voisins du point de même niveau que lui (voir Fig. 5.4).

Puisque situés à l'intérieur des régions de Voronoï, les points d'un maillage fin ne peuvent être fils que d'*un* point du maillage grossier. Cette relation permet donc de définir une structure d'arbre couvrant tous les sommets des maillages, chaque père étant lié à ses fils (voir Fig. 5.4).

On stocke aussi dans chaque point l'ensemble des premiers voisins de ses fils (hormis les fils eux mêmes). Ces points seront en effet nécessaires aux fils lorsqu'ils deviendront actifs pour pouvoir calculer une force (Fig. 5.4).

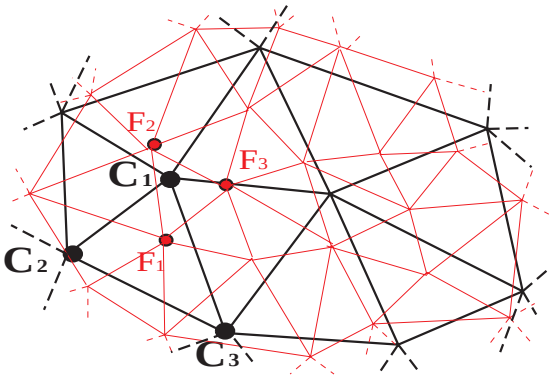


FIG. 5.3: Liaison des maillages (en 2D) : en fonction des maillages effectivement simulés, le sommet C_1 pourra calculer sa position à partir de celles des points (F_1, F_2, F_3) du maillage fin. Les déformations pourront aussi être propagées aux noeuds du maillage fin : F_1 pourra déduire sa position de celle des points (C_1, C_2, C_3) de son triangle parent.

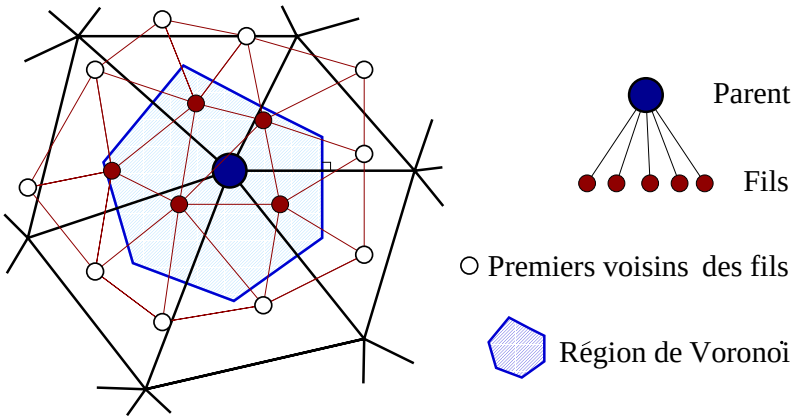


FIG. 5.4: Les fils d'un point sont les sommets du maillage fin situés dans sa zone de Voronoï. Ce sont eux qui remplaceront le point lorsqu'il se subdivisera. On stockera également la liste des premiers voisins de ces fils.

3.2 Subdivision et regroupement d'un point

Lorsqu'un point actif de l'un des maillages devient trop grossier pour représenter les déformations de cette zone, il se subdivise et est remplacé par ses fils. Il acquiert alors le statut de fantôme : il déduira son déplacement de celui de son tétraèdre fils. Sa position pourra être utilisée par ses voisins, qui peuvent eux être encore actifs. Ses fils ont quant à eux le statut d'actifs, et ils simuleront plus précisément les déformations de la région de Voronoï du point subdivisé.

Pour pouvoir se simuler, les points fils créés ont besoin d'avoir des déplacements actualisés en chacun de leurs voisins. C'est à cela que vont servir les premiers voisins des fils, qui deviendront fantômes (ou resteront actifs s'ils l'étaient déjà) lors de la subdivision.

Ces premiers voisins fantômes vont donner aux fils les déplacements des zones voisines, soit directement s'ils sont actifs (leur particule parente s'étant divisée), soit par moyenne, en utilisant une interpolation sur leur tétraèdre parent.

Inversement, on pourra simplifier une zone de l'espace en remplaçant un ensemble de fils par leur parent, celui-ci redevenant actif. Les points fils ne disparaissent pas forcément de la simulation car ils peuvent être nécessaires comme fantômes, voisins d'un point encore actif. Pour le savoir, un compteur indique combien de fois chaque point a été sollicité pour être premier voisin actif. Ce compteur est mis-à-jour lors de la subdivision ou du regroupement d'un point, et l'on sait quand il est à zéro que le point fils peut être supprimé lors du regroupement.

3.3 Critères de subdivision-regroupement

Les critères de subdivision et de regroupement des particules sont également les mêmes que ceux décrits au Chapitre 3, Sec. 4.5, page 54. Ils sont à nouveaux basés sur la variation d'amplitude de la norme du laplacien, qui représente la dérivée seconde locale du champ, et donc son inadéquation avec le modèle linéaire utilisé pour calculer les opérateurs.

Rappelons que contrairement à ce qui est décrit au Chapitre 3, on n'a pas ici à chercher à conserver un octree restreint.

4 Raffinements du modèle

4.1 Utilisation d'un octree non restreint

Contrairement à ce qui avait été fait dans le premier modèle de cette thèse (Chapitre 3), nous avons ici choisi de ne pas nous limiter à la création d'un octree restreint. Une zone simulée avec des particules actives d'un certain niveau pourra donc côtoyer des régions simulées à n'importe quel niveau de la hiérarchie (et non plus seulement les niveaux directement inférieurs et supérieurs).

Ce choix simplifie grandement les procédures de subdivision et de regroupement de particules qui n'ont plus à chercher à conserver la structure d'octree restreint. Outre l'économie faite en temps de calcul par les vérifications supprimées, on gagne en simplicité et en lisibilité algorithmique, la conservation d'un octree restreint étant en pratique très subtile à programmer efficacement.

Justification

La conservation d'un octree restreint n'est plus vraiment nécessaire avec notre méthode d'animation puisque les résolutions ne communiquent plus vraiment entre elles comme c'était le cas dans la première méthode du Chapitre 3. L'utilisation de points fantômes fait que les déplacements d'une zone d'interface peuvent être déduits récursivement de n'importe laquelle des résolutions actives de cette zone. S'il y a plus d'un niveau d'écart entre une zone active et les points actifs d'une zone d'interface voisine, les différents points fantômes vont permettre en transférant l'information d'un maillage au suivant de la propager jusqu'à la résolution adéquate.

Les résultats de la première méthode ont également montré qu'un critère de subdivision bien choisi, en l'occurence fonction de la courbure du champ déplacement (voir Chapitre 3, Sec. 4.5), permet d'obtenir en pratique une discrétisation intuitive du matériau, proche de celle d'un octree restreint (voir Sec. 3.3). Nous espérons donc que la structure conservera d'elle-même son caractère d'octree (quasi) restreint.

Subdivision modifiée

Du fait de la non restriction à un octree restreint, la procédure de subdivision d'un point se trouve un peu modifiée. Les premiers voisins des points fils sont en effet toujours ajoutés comme fantômes à la simulation (s'ils n'étaient déjà actifs), mais doivent maintenant en plus s'assurer qu'ils seront capables de mettre à jour leur déplacement.

Il se peut en effet que les points composant leur tétraèdre parent ne soient pas encore actifs, cette région n'ayant pas été subdivisée. Si tel est le cas, ces points seront eux aussi ajoutés comme fantômes et iront chercher l'information au niveau supérieur. Cette procédure est récursive et des fantômes sont ajoutés tant que l'on n'est pas remonté au niveau réellement actif de cette zone.

Suppression des fantômes

Tous ces points fantômes qui peuvent être ajoutés doivent donc aussi être supprimés lors du regroupement de la particule. Il est assez lourd de conserver dans chacun de ces points fantômes un compteur indiquant le nombre de particules de niveaux inférieurs ayant encore besoin du déplacement de ce point pour mettre à jour leur propre déplacement par interpolation. Il faudrait alors à chaque regroupement parcourir récursivement les points fantômes des tétraèdres parents des premiers voisins pour voir s'ils ne peuvent être supprimés.

Il est de plus possible qu'un de ces points, juste après avoir été supprimé, soit de nouveau nécessaire comme fantôme, une nouvelle subdivision le réclamant. Aussi avons-nous mis en place un processus de "ramasse-miettes" qui servira à supprimer les points fantômes devenus inutiles.

Régulièrement, et en pratique à chaque affichage de la scène (soit généralement tous les 32, 64 ou 128 pas de simulation selon le pas de temps utilisé), l'ensemble des points fantômes *non subdivisés* est parcouru pour y supprimer ceux devenus inutiles.

Les points fantômes subdivisés ne sont pas testés car on les sait probablement utiles à la simulation. Il est possible, si le matériau est très largement subdivisé, qu'un point fantôme d'un niveau très grossier ne serve plus, tous ses voisins étant également fantômes. Ce cas apparaissant rarement et ne concernant que quelques points au plus, nous ne le traitons pas.

4.2 Mise à jour des points fantômes

Nous avons mis en évidence un problème avec cette méthode lorsque les maillages successifs de l'objet ne sont pas suffisamment imbriqués. Il peut en effet arriver que les quatre points du *tétraèdre fils* d'un point donné (ceux qui serviront à calculer son déplacement une fois subdivisé) ne soient pas tous directement des fils de ce point (le point F dans la Figure 5.5).

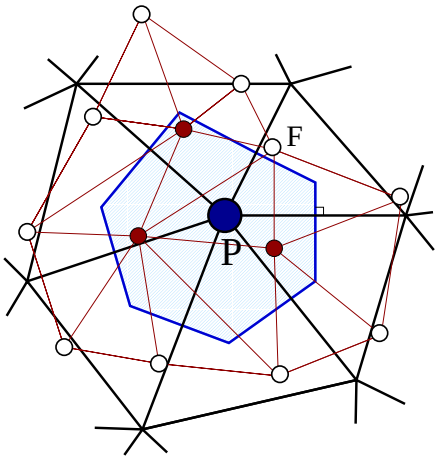


FIG. 5.5: Un des points (F) du tétraèdre fils de P ne fait pas directement partie des fils de P.

Si l'un des quatre points F^i du tétraèdre fils ne fait pas partie des fils d'un point P donné, on obtient une configuration assez confuse lors de la subdivision de P. Sa position dépend alors en effet de celle d'un point F qui peut être également fantôme (si les zones voisines de P ne se sont pas subdivisées).

Si tel est le cas, on a alors une dérive systématique de la position du point P. Les positions des points fantômes F et P dépendent alors en effet mutuellement l'une de l'autre. Selon l'ordre dans lequel on calcule ces positions, on introduira alors forcément une dérive dans leurs positions, l'une étant mise à jour avec un pas de simulation de retard sur l'autre. Cela se traduit dans les simulations par un phénomène d'amplification des oscillations, une sorte d'inertie artificielle étant alors introduite.

Pour éviter ce problème nous avons choisi de déclarer comme actifs tous les sommets du tétraèdre fils d'un point qui se subdivise. On évite ainsi de créer cette boucle de dépendance. L'autre solution pourrait être d'utiliser des maillages qui évitent ce genre de problèmes, ce qui est faisable si l'on augmente la différence entre les résolutions de deux maillages successifs.

Plus généralement, et pour vraiment pouvoir considérer les points fantômes comme des points au déplacement correctement réactualisé, il faudra veiller à ce que leur mise à jour se fasse dans un ordre donné. Il faudra mettre leur position à jour en *remontant* dans l'arbre, des feuilles vers la racine, les points fantômes dépendant d'autres points fantômes étant mis-à-jour après ceux-ci. L'information la plus à jour contenue dans les feuilles (particules actives) sera ainsi bien propagée à tous les niveaux sans introduire de retards.

4.3 Intégration temporelle

Nous avons essayé d'utiliser différentes méthodes d'intégration temporelle sur ce modèle. Nous nous sommes limités à des méthodes ne demandant qu'une seule évaluation par pas de simulation [2]. Euler modi-

[2] Toutes ces méthodes sont décrites en Annexe B.

fié et Stoermer donnent rigoureusement les mêmes résultats. On préférera utiliser Euler modifié car il fournit une expression de la vitesse des points, nécessaire au calcul des forces de frottement internes. Newton-Cotes est, quand à lui, bien moins stable.

Tout comme dans le Chapitre 3 (voir page 55) consacré à notre premier modèle, les particules bénéficient ici aussi d'une multirésolution temporelle, basée sur les mêmes critères (critère de Courant et limitation de la modification de la vitesse).

Utilisation d'un W-cycle

Différentes listes contiennent à chaque instant la liste des particules utilisant un dt_i donné. Rappelons que ces dt_i sont des sous-multiples par une puissance de deux du dt d'affichage.

$$dt_i = \frac{dt_{\text{affichage}}}{2^i}$$

Un schéma de type *leap-frog*, aussi nommé *W-cycle*, permet d'éviter que certains pas de simulations aient à simuler *tous* les dt_i alors que d'autres pas ne simuleront que le niveau dt_{max}. Si tel est le cas, la durée de calcul de ces différents pas peut varier énormément, ce qui nuit à une simulation temps-réel.

Au lieu de cela, la simulation des différents dt_i sera régulièrement répartie, chaque pas de simulation nécessitant un temps de calcul plus constant, puisqu'il ne simule que deux (voire un) dt_i à chaque étape (Fig. 5.6).

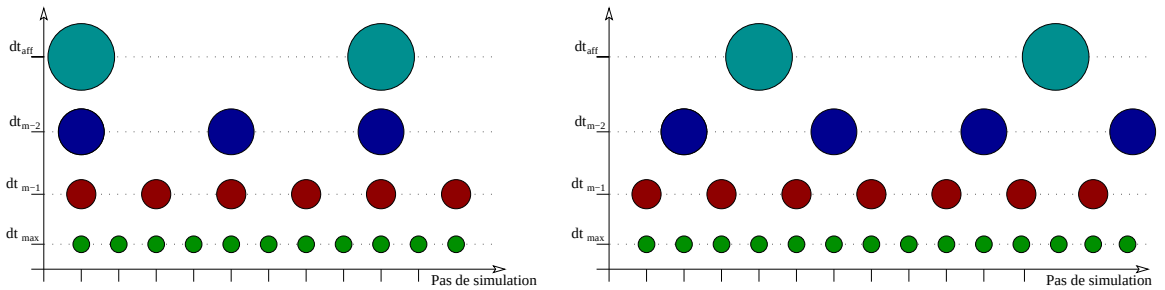


FIG. 5.6: La synchronisation décalée des simulations des différents pas de temps permet d'éviter les goulots d'étranglement en répartissant la charge sur toute la simulation.

Intégration semi-implicite

Nous avons également implémenté une intégration *semi-implicite*, inspirée des méthodes de [DSB99] [3]. Cette terminaison n'est pas officielle et s'inspire de la différence entre éléments finis explites et implicites. Plutôt que d'inverser à chaque pas de temps une matrice globale représentant tout le système comme le fait l'intégration implicite, nous allons une nouvelle fois approximer *localement* les effets de l'intégration implicite.

Nous allons précalculer pour chaque particule la matrice jacobienne de sa force $J = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}$. Cette matrice est une approximation au premier ordre de la force subie par cette particule lorsqu'elle se déplace. Pour l'estimer, nous calculons la force subie par chaque particule lorsqu'elle est déplacée dans les trois directions x,y et z, en supposant fixes et à leur position de repos toutes les autres particules.

L'amplitude de ce déplacement n'a pas d'importance dans le cas linéaire (tenseur de Cauchy, méthode hybride), un déplacement double créant une force double et donc un jacobien identique. Il n'en va pas de même losque l'on utilise le tenseur de Green-Lagrange, et nous choisissons alors de déplacer la particule avec une amplitude voisine de celle qu'elle subira lors de la simulation. En pratique nous avons fixé cette amplitude à 1/10ème de la distance inter-particulaire minimale du maillage auquel appartient la particule.

Comme dans [DSB99], nous allons supposer cette matrice J constante au cours du temps, ce qui permet d'approximer une intégration implicite pour un coût modique. L'intégration semi-implicite consiste alors simplement en la multiplication à chaque pas de temps de la vitesse par une matrice précalculée.

[3] Voir les détails en Annexe B.

$$\mathbf{v} \mathrel{+}= \frac{\mathbf{f}}{m} dt \qquad \mathbf{v} \mathrel{*}= (I_3 - \frac{dt^2}{m} J)^{-1} \qquad \mathbf{x} \mathrel{+}= \mathbf{v}\, dt \tag{5.1}$$

L'application de ce procédé semi-implicite stabilise l'intégration temporelle et nous permet en pratique de doubler le pas de temps d'intégration sans qu'apparaissent d'instabilités.

Les forces de frottements additionelles créées par cette méthode peuvent être compensées en réduisant les coefficients ϕ et ψ qui quantifient la dissipation interne (Équation 2.22). Noter toutefois que ces frottements dissipent également les rotations rigides de l'objet, à l'inverse des frottements issus de l'Équation 2.22, ce qui peut être gênant dans certaines applications.

5 Simulation temps-réel

Nous allons ici expliquer comment obtenir un simulateur chirurgical temps-réel utilisant le modèle multi-résolution de ce chapitre.

5.1 Parallélisation

L'une des machines utilisée pour les démonstrations du prototype de simulateur comporte six processeurs. Nous avons donc cherché à utiliser cette puissance pour accélérer l'exécution du programme.

Plusieurs processus différents vont pouvoir être lancés simultanément. Ils partageront la même zone de mémoire (à l'exception du processus gérant la boucle de simulation du *Phantom*) et seront synchronisés à l'aide de sémaphores.

Le premier de ces processus gère donc le retour haptique utilisé par le bras à retour d'effort *Phantom*. Ce processus bénéficie d'une priorité *real-time* qui assure qu'il sera mis-à-jour à la fréquence de 1000Hz. Cette fréquence de rafraîchissement de la force est en effet imposée par le *Phantom* pour maintenir une sensation de force cohérente.

Notre simulation ne met pas forcément à jour la force à cette fréquence (en pratique ce sera fait à une fréquence variant entre 400 et 3700Hz) et ce processus se contente d'envoyer à nouveau la dernière force imposée en attendant la suivante. Il met également à jour les variables indiquant la position du bras. Récupérer les positions et envoyer des forces se fait donc de manière asynchrone par le programme principal de la simulation.

Les différentes étapes de la simulation proprement dites sont les suivantes (le pourcentage indique grossièrement la part du temps de la simulation consacrée à chaque partie) :

- simulation du modèle interne (60%) ;
- mise-à-jour des positions des points de la surface (5%) ;
- détection des collisions entre la surface et l'outil (20%) ;
- affichage de la scène (15%) ;
- attente éventuelle pour se synchroniser avec l'affichage.

La version mono-processus exécute ces tâches, dans cet ordre, entre deux affichages successifs.

La version multi-processus va séparer la simulation du modèle interne, partie la plus coûteuse de l'algorithme, dans un processus à part. Apparaissent alors des dépendances entre les processus : la simulation ne peut débuter qu'*après* la détection de collision et doit se terminer *avant* la mise-à-jour des points de la surface.

Nous avons décidé de décaler d'un affichage la détection et la réponse aux collisions afin de désynchroniser ces deux étapes. L'affichage prendra en compte la collision qui a été faite au pas de simulation *précédent* en non plus à celui qui vient d'avoir lieu. Ce décalage de 1/25ème de seconde n'est pas visible en pratique et la surface semble toujours bien repoussée hors de l'outil.

Le graphe représentant la simulation devient alors :

Noter qu'il est alors possible que la simulation du pas suivant commence *avant* l'affichage courant. On optimise ainsi le temps de calcul utilisable par la simulation qui peut aller jusqu'à celui séparant deux affichages moins le temps (court) nécessaire à la mise-à-jour de la surface.

Enfin, un autre processus a été utilisé pour gérer l'affichage de textures dynamiques gourmandes en calcul sur la surface du foie (gouttes de sang, blanchiment, brûlures). Nous ne détaillerons pas davantage ce procédé lié au rendu de l'objet et développé dans le cadre de l'action incitative *AISIM*.

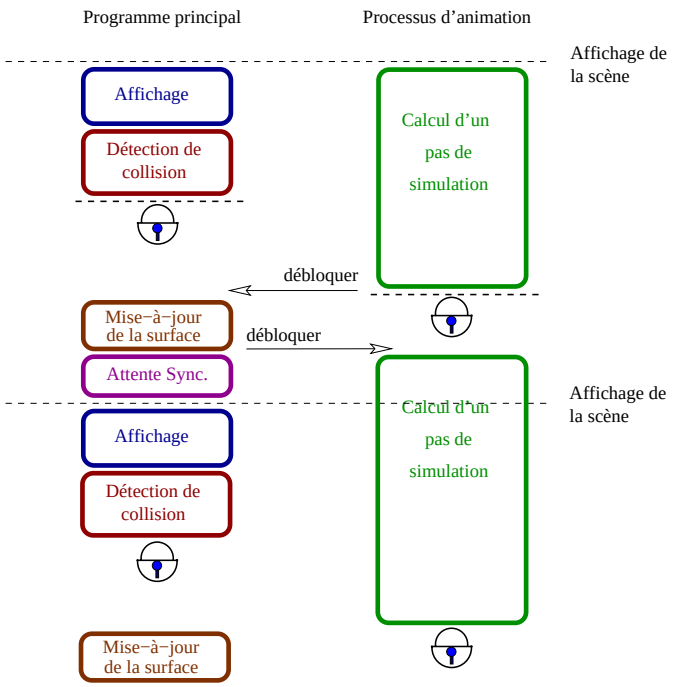


FIG. 5.7: Le programme principal et le processus chargé de l'animation sont liés par un système de sémaphores assurant leur synchronisation.

5.2 Temps-réel vrai

Un petit processus annexe sert uniquement à compter le nombre de rafraîchissements d'écran effectués depuis le dernier affichage. Il sert à assurer une simulation en temps-réel vrai en faisant en sorte que le programme attende éventuellement pour afficher une nouvelle image (étape "Attente Sync" de la Figure 5.7). On garantit ainsi que, même si l'on peut calculer plus vite que nécessaire, on attendra pour respecter le temps-réel.

Cette attente, même infime est nécessaire, et il faudra veiller à ne pas *dépasser* le temps imparti, ce qui décalerait l'affichage d'une *frame*. Pour ce faire, on peut limiter le nombre de particules de la simulation en limitant la subdivision.

Une variable indique en permanence le pourcentage de temps passé à la simulation pour le pas précédent, par rapport au temps total disponible entre deux affichages. Un seuil à ne pas dépasser (95% dans nos exemples) détermine s'il est encore possible de subdiviser. On garantit ainsi un affichage régulier de la scène.

6 Résultats

Nous présentons ici quelques images issues de la simulation. La Figure 5.8 montre l'influence de l'augmentation du terme de préservation de volume sur la forme de l'objet déformé.

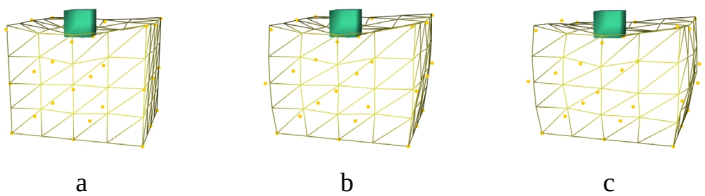


FIG. 5.8: Influence de l'augmentation de λ sur la préservation de volume. Les images représentent $\lambda = 0$ (a), $\lambda = 50000$ (b) et $\lambda = 500000$ (c).

Lorsque l'outil vient appuyer trop fortement sur la surface, les maillages plus fins deviennent actifs dans cette zone, montrant ainsi l'adéquation du critère basé sur le laplacien pour définir les points à subdiviser ou à regrouper.

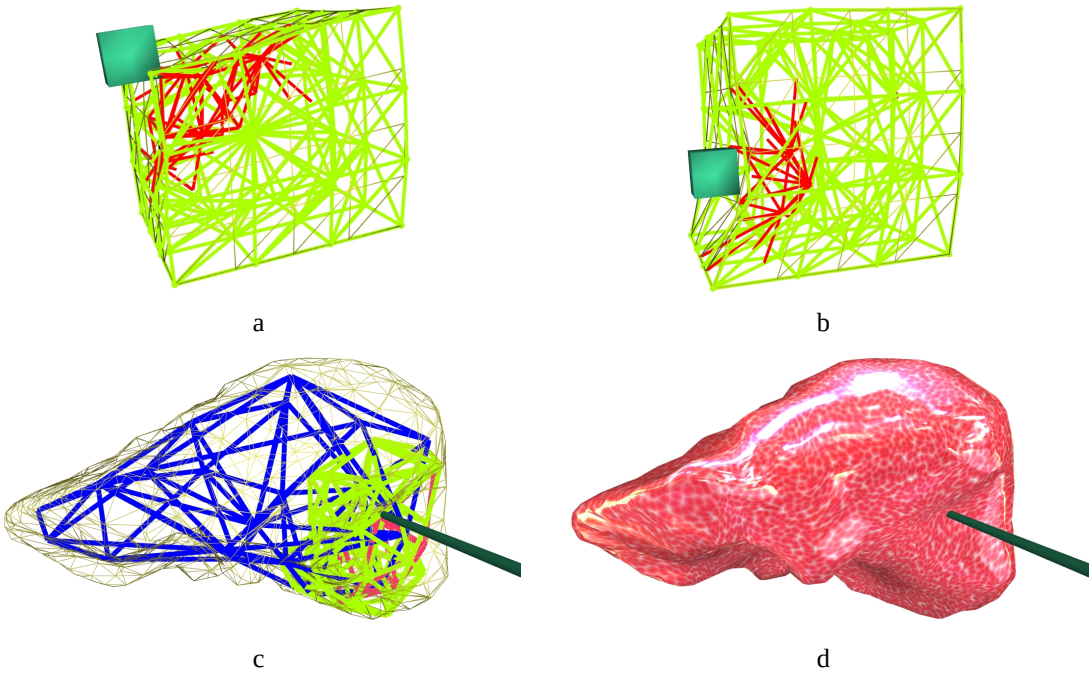


FIG. 5.9: L'utilisation des maillages fins, faite d'après des critères liés à la continuité, résulte en une discrétisation intuitive des zones proches de l'outil.

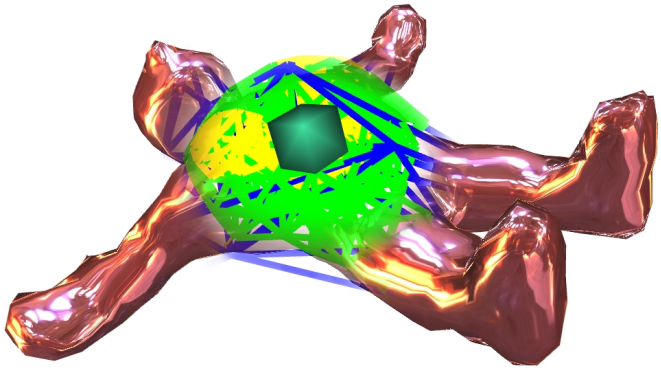


FIG. 5.10: Autre exemple d'objet déformable. Les secousses appliquées sur le ventre de ce bonhomme se répercutent sur ses bras qui oscillent légèrement.

7 Conclusion

Nous avons présenté dans ce chapitre un deuxième modèle multirésolution, utilisant les opérateurs décrits dans le chapitre précédent. L'originalité de cette approche réside dans l'utilisation de maillages *indépendants* pour représenter l'objet. Chaque zone de l'objet est réellement simulée en utilisant un et un seul des maillages, choisi en fonction des besoins de la simulation.

Ces différentes zones interagissent avec les zones voisines, simulées par d'autres maillages, grâce à l'utilisation de points fantômes. En propageant la déformation calculée d'un maillage à un autre, ces points fantômes permettent aux maillages de connaître la déformation de leur voisinage et peuvent ainsi simuler leur propre déformation. Bien que composé de différents maillages, l'objet forme donc un tout cohérent.

Ce modèle multirésolution étant mis au point, il reste à l'"habiller" en lui associant une surface. Ce lien entre la surface et le modèle interne est le propos du prochain chapitre.

CHAPITRE 6

Interface avec l'utilisateur

BEAUCOUP du réalisme d'un simulateur provient de la qualité des images affichées. Le mouvement de l'objet décrit lors des précédents chapitres ne concerne que celui de particules représentant l'intérieur de l'objet. Nous allons dans ce chapitre décrire comment la *surface* de l'objet va rendre compte de ces déplacements. Nous détaillerons également la façon dont ont été implémentées la détection et la réponse aux collisions avec un objet virtuel, et en particulier le calcul de la force renvoyée quand on utilise un dispositif à retour d'effort.

C'est donc, plus généralement, de la gestion de l'interface avec l'utilisateur que nous allons maintenant discuter. Cette partie va permettre d'utiliser les résultats des chapitres précédents en décrivant la dernière couche du prototype de simulateur laparoscopique que nous avons créé.

1 Affichage de la surface

La surface sert non seulement à reproduire les déformations calculées de l'objet, mais aussi, dans notre cas, à cacher l'utilisation d'un processus multirésolution à l'utilisateur. Celui-ci n'a en effet pas à être troublé par la variation du nombre de particules utilisées.

1.1 Liaison avec les particules

La surface des objets sera constituée de triangles, primitive classique et optimisée pour l'affichage d'une surface. Le nombre de triangles composant la surface devra être un compromis entre qualité et rapidité, dépendant des capacités graphiques de la machine.

Ce choix n'affecte pas la résolution des maillages volumiques utilisés pour la simulation, la surface et le modèle physique interne étant *totalement* décorrélés. En pratique, notre maillage triangulaire sera pour la plupart des objets celui correspondant au maillage tétraédrique le plus fin, composé de quelques milliers de triangles.

La surface sera rattachée aux particules internes à l'aide de *liens*. Ceux-ci sont des vecteurs d'*offset* $\mathbf{o}$ fixes, qui définissent la position $\mathbf{n}$ d'un nœud de la surface directement à partir de celle $\mathbf{p}$ d'une particule interne (Fig. 6.1) :

$$\mathbf{n} = \mathbf{p} + \mathbf{o}$$

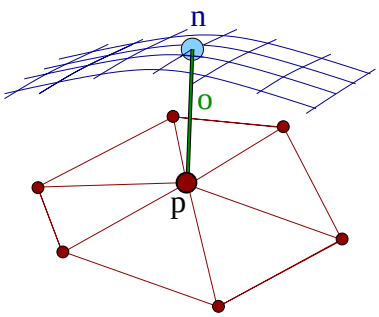


FIG. 6.1: Un nœud de la surface est lié à une particule du modèle interne grâce à un décalage **o**.

Gestion de la multirésolution

Un précalcul permet de déterminer quelle est la particule la plus proche de tout nœud de la surface ainsi que le décalage **o** associé. Cette recherche est en fait réalisée dans *tous* les maillages, de sorte qu'un nœud pourra déduire sa position de celle de n'importe quel maillage, et en pratique de celui qui sera actif dans cette zone.

La définition des fils d'une particule comme étant les particules du maillage inférieur situées à l'intérieur de sa région de Voronoï assure qu'un nœud de la surface sera lié à des particules faisant partie de la même sous-branche de l'arbre : si un nœud est lié à p dans un maillage, ce nœud sera lié à l'une des filles de p dans le maillage inférieur, et ainsi de suite.

Parmi les différents points (un par maillage) auxquels un nœud est lié, un seul sera donc actif à tout moment de la simulation. Lorsque la discrétisation interne évolue et entre deux affichages successifs, il est donc très facile à chaque nœud de déterminer quelle est le point actif parmi ceux auxquels il est lié. Si ce n'est celui qu'il a utilisé pour déterminer sa position à l'affichage précédent, c'est donc son fils (ou son père).

On recherche donc, en partant du niveau utilisé à l'affichage précédent, quel est le niveau réellement actif pour ce nouvel affichage. Ce sera très souvent le même, parfois celui directement supérieur ou inférieur et très rarement un niveau plus éloigné. Cette recherche est donc très rapide.

Lissage de la position

La méthode précédemment décrite a l'inconvénient de déplacer trop grossièrement les nœuds de la surface lorsque la simulation utilise des maillages volumiques internes de faible résolution. Dans ces cas là, on déduit en effet rigidement la position de quelques milliers de nœuds de celle des quelques dizaines de particules simulées. La conséquence est que le mouvement de la surface se fait par *plaques*, chacune étant liée à l'une des particules internes actives.

Pour éviter ce problème et lisser les positions des nœuds de la surface, nous utilisons un filtrage de la position de *plusieurs* particules internes pour déterminer la position d'un nœud. Nous avons choisi encore une fois la simplicité et la rapidité en effectuant une interpolation linéaire de la position de plusieurs particules.

Un nœud va être lié à trois particules, celles-ci formant le triangle au dessus duquel il se trouve (voir Fig. 6.2). Sa position sera alors une somme pondérée des positions décalées des particules :

$$\mathbf{n} = \frac{\sum_{i \in 1..3} w^i (\mathbf{p}^i + \mathbf{o}^i)}{\sum_{i \in 1..3} w^i} \qquad w^i = \frac{1}{||\mathbf{o}^i||} \tag{6.1}$$

Les coefficients pondérateurs w^i sont pris comme étant inversement proportionnels aux distances entre le nœud et les trois points $\mathbf{p}^i$ (analogie avec coordonnées barycentriques de la projection du nœud sur le triangle). Ils sont renormalisés de façon à ce que leur somme fasse 1 et que l'on évite ainsi d'effectuer une division inutile. Toutes ces informations (w^i, $\mathbf{p}^i$ et $\mathbf{o}^i$ pour chaque niveau de la hiérarchie) sont précalculées et stockées dans les nœuds de la surface.

Un nœud est ainsi capable de déterminer, avec plus ou moins de précision, sa position à partir de celles des triangles du modèle physique. Il utilisera à un instant donné de la simulation le triangle mis-à-jour (i.e. composé de points actifs ou fantômes) offrant la meilleure précision, c'est-à-dire appartenant au niveau le plus fin (voir Fig. 6.2). On obtiendra ainsi un déplacement optimal du nœud en fonction de la précision des déformations disponible dans sa zone.

Il peut arriver avec cette méthode que la position d'un nœud change soudainement lors des subdivisions ou des regroupements de points, le nœud changeant alors de triangle de référence. Pour éviter cet effet de *popping*

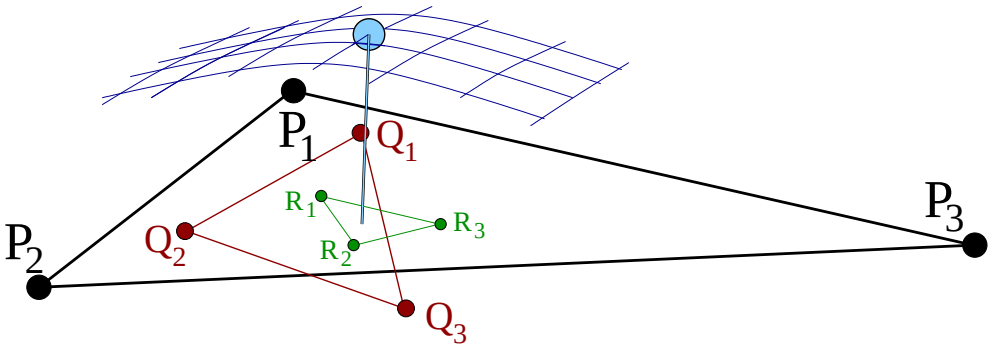


FIG. 6.2: Un nœud de la surface est lié aux différents triangles surfaciques des maillages tétraédriques.

de la surface, on pourra ajouter un léger filtrage temporel permettant de passer continuement, sur quelques images, d'un triangle à un autre.

On peut aussi limiter ce phénomène en jouant sur les seuils de subdivision et de regroupement pour contrôler à partir de quelle distance de la position d'équilibre (qui par construction ne produit aucun *popping*) on décide de subdiviser, limitant ainsi les déplacements brusques.

Ce modèle à base de décalage rigide entre la surface et le modèle interne est en théorie limité à de faibles déplacements, et devrait en particulier mal supporter les rotations de l'objet. En pratique, les vecteurs de décalage $\mathbf{o}^i$ sont suffisamment petits pour que des rotations raisonnables ne produisent pas d'effet visuel trop notable.

Si c'était le cas, il suffirait de définir les décalages $\mathbf{o}^i$ dans un repère *local* lié à chaque triangle. Cela demanderait simplement de mettre à jour la normale de ces triangles, ce que nous n'avons pas jugé nécessaire de faire.

2 Collisions avec l'outil

Le simulateur chirurgical va consister principalement en un objet déformable (un modèle de foie dans nos exemples) et un outil virtuel manipulé par le chirurgien. Nous allons voir comment ces deux objets interagissent.

2.1 Détection de la collision

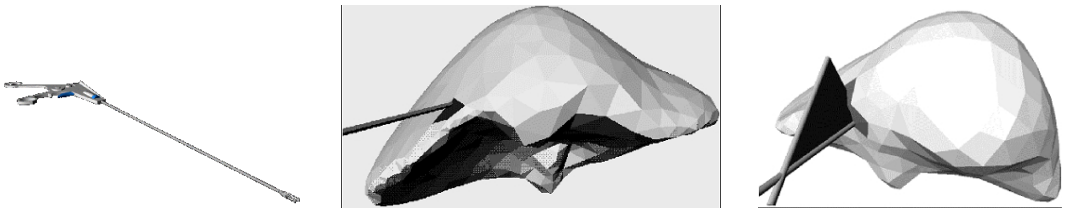
Les méthodes classiques de détection de collision sont bien adaptées aux objets rigides (partitionnement de l'espace et boîtes englobantes pour accélérer la recherche). Pour traiter la collision avec un objet déformable, il convient d'utiliser d'autres méthodes.

Dans le cadre de l'action incitative AISIM [INR] a été mis au point une méthode très rapide basée sur l'utilisation du matériel graphique [LCN99]. Profitant du fait qu'un seul des objets est déformable et que l'outil est de forme assez simple (voir Fig. 6.3a), l'idée est de placer une caméra orthographique à l'intérieur de l'outil pour obtenir rapidement les parties de la surface qui sont éventuellement intersectées. Un rendu hors-écran (*offscreen*) de la surface de l'objet, vue au travers de cette caméra, et tirant ainsi parti des performances graphiques de la machine, peut fournir la liste des triangles intersectés.

Si l'outil est déplacé rapidement, et en fonction de la fréquence des détections, il peut avoir parcouru une grande distance entre deux détections de collision successives. Ce cas peut être traité en utilisant une caméra projective classique et des plans de *clipping* qui définiront l'espace dans lequel s'est déplacé l'outil (voir Fig. 6.3c).

Même rapide (150 fois plus qu'avec des algorithmes de partitionnement de l'espace comme *Rapid*), cette détection reste assez chère. Aussi avons-nous choisi de ne la faire qu'avant chaque affichage de la scène et non pas à chaque pas de simulation. En faisant ainsi coïncider affichage et détection de collision, on s'assure que l'affichage prendra bien en compte la collision qui vient éventuellement d'avoir lieu.

Les mouvements d'un chirurgien étant assez lents et la détection se faisant à environ 25Hz, le déplacement de l'outil est au plus de quelques millimètres entre deux détections successives et il n'est donc pas nécessaire d'utiliser en pratique la version dynamique de la méthode(Fig. 6.3c) ou une détection plus fréquente.



(a) (b) (c)

FIG. 6.3: Une caméra placée à l'intérieur de l'outil (a) permet de détecter rapidement les collisions avec l'objet, que l'on considère l'outil comme statique (b) ou dynamique (c).

2.2 Réponse à la collision

Une fois la collision détectée, reste à la propager au modèle physique interne. Cette partie est loin d'être évidente et la solution proposée, bien que rapide, est parfois mise en défaut en particulier quand la taille de l'objet est importante (ce qui n'est pas le cas avec les outils laparoscopiques).

La détection de collisions a renvoyé une liste de polygones, qui sont les intersections des triangles formant la surface avec le volume de vue de la caméra (*frustum*). Le traitement décrit ici va être appliqué à chacun de ces triangles.

On calcule tout d'abord le barycentre B du polygone d'intersection entre le triangle et la caméra (voir Fig. 6.4). Les trois sommets du triangle seront ensuite déplacés au *prorata* des coefficients barycentriques de B dans le triangle.

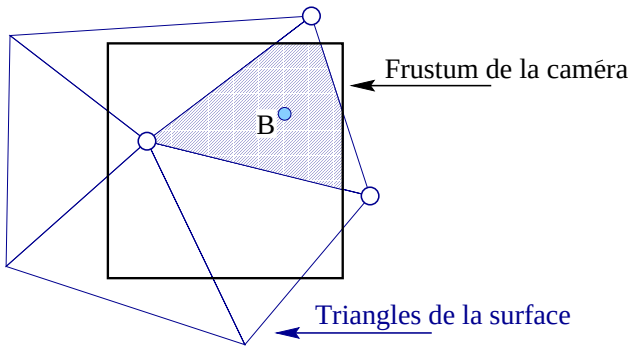


FIG. 6.4: Pour chacun des triangles intersectés, on calcule le centre de gravité B de la zone d'intersection. Les trois sommets du triangle seront déplacés en fonction des coefficients barycentriques de ce point.

Déterminer la meilleure direction de déplacement des nœuds du triangle est assez complexe. Nous avons choisi de la diriger selon l'opposée de la normale du triangle, de sorte que la surface sera déplacée le plus efficacement possible. Sa norme est telle que le centre de gravité B sorte de l'outil. On ne garantit donc pas que le triangle sera totalement repoussé hors de l'outil, ce qui serait difficile à calculer, mais on fait en sorte qu'il sorte suffisamment. En quelques itérations de l'algorithme, le triangle sera presque totalement hors de l'outil.

Un coefficient permet de régler à quel point on désire pousser B hors de l'outil. Il peut être utile de le régler en fonction de la rigidité de l'objet, mais un coefficient de 1 (le point B est repoussé exactement à la limite de l'outil) donne de bons résultats, la surface ne paraissant pas rester en collision.

Une fois les points de la surface déplacés, il reste à communiquer ce mouvement aux particules internes. Ceci est fait en utilisant le même processus que celui qui avait permis, dans l'autre sens, de déterminer la position de la surface à partir de celle des particules. Les particules internes sont en effet déplacées en utilisant les poids w^i et les décalages $\mathbf{o}^i$ (voir Eq. 6.1) qui les lient aux nœuds de la surface.

Chaque particule peut recevoir plusieurs déplacements, pondérés par les w^i, de la part des différents nœuds de la surface qui lui sont liés. Elle moyennera donc à la fin du processus de réponse aux collisions les déplacements qu'elle a reçus pour savoir quelle est la réelle influence de l'outil sur sa position.

3 Retour d'effort

En plus d'un affichage et d'un mouvement de qualité, le retour haptique est une composante importante d'un simulateur réaliste. Différents systèmes permettent de transmettre à l'utilisateur une force dépendant de ses gestes et de la simulation. On pourra se référer à [Cot97, MPT99] pour un inventaire des techniques existantes.

Nous avons utilisé un dispositif de retour haptique de type *PHANToM*, produit par *Sensable Technologies* [Tec]. Donnant la position de ses 6 degrés de liberté, il permet en retour de transmettre à l'utilisateur une force de direction et d'intensité données.

FIG. 6.5: Le Phantom desktop.

3.1 Principe

Toutes les particules déplacées lors de la détection de collision seront stockées dans une liste. Lors des pas de simulation qui auront lieu entre deux détections de collision, ces particules calculeront bien la force qui leur est appliquée, mais ne l'*intégreront* pas. À la place, elle transmettront cette force à l'utilisateur, qui sentira donc l'effet de son geste et devra contrecarrer cette force pour maintenir l'objet dans sa position déformée.

On simule entre 16 et 128 pas de simulation entre chaque détection de collision ce qui avec un affichage à 25-30Hz correspond à un retour haptique d'une fréquence allant entre 400 et 3700 Hz, compatible avec les sensations fines du toucher.

3.2 Lissage de la force

Si l'on se contente de sommer les forces issues de toutes les particules déplacées pour les renvoyer à l'outil, on risque d'obtenir des discontinuités dans la force. Un très léger déplacement de l'outil peut en effet le faire entrer en collision avec un nouveau triangle de la surface, de nouvelles particules internes venant ajouter leur force. On aura alors une soudaine et artificielle augmentation de la force reçue (et inversement une diminution si on quitte un triangle).

Pour éviter ce problème, il suffit de quitter cette version discrétisée de la surface pour repasser à une version continue. La force ne sera pas alors directement liée au nombre de particules déplacées, mais plutôt à la *surface* de la zone de collision. On moyenne donc les forces issues des particules déplacées et on multiplie cette force moyenne par la surface de contact.

Il apparaît un autre problème avec la méthode précédemment décrite. Lorsque l'on déplace les particules internes après la détection de collision, celles-ci bougent soudainement puis restent immobiles pendant tous les pas de temps suivants, se contentant de transmettre la force calculée. Ce déplacement brusque crée une soudaine variation de la force, qui évolue ensuite lentement durant les pas de simulation suivants.

Pour éviter ce problème, il suffit d'appliquer *progressivement* le déplacement lié à la collision. On divise donc le déplacement imposé par les 16, 32, 64 ou 128 pas qui vont avoir lieu et on applique cette petite partie de déplacement à chaque pas, la collision s'appliquant ainsi continuement entre deux détections.

Le dernier problème rencontré avec le retour d'effort vient de la réponse aux collisions qui repousse chaque triangle de plus en plus hors de l'outil. Arrive un moment où le triangle est totalement sorti de l'outil et n'entre donc plus en collision. Les particules précédemment immobiles qui y sont attachées se retrouvent libres et se déplacent donc d'un coup vers leur position de repos. Elles entrent alors forcément en contact avec l'outil qui n'a pas bougé énormément. Celui-ci les repousse fortement, créant un brusque à-coup dans la force transmise.

Cet inconvénient peut être supprimé en empêchant que le triangle puisse être totalement sorti hors de l'outil. Au delà d'un certain seuil (0.1mm par exemple), on considère la collision comme suffisamment gérée et l'on arrête de repousser le triangle. On évite ainsi ce brusque retour dans l'outil du triangle libéré.

Il faut par contre faire en sorte d'éviter que le triangle ne "colle" à l'outil, le suivant dans ses déplacements puisqu'on l'empêche de totalement sortir. On pourra alors comparer la direction de mouvement de l'outil, la normale à la surface et la force subie par la particule pour repérer les cas où l'on cherche à sortir de l'objet et où il faut laisser sortir le triangle.

4 Conclusion

Nous avons décrit dans ce chapitre la façon dont sont reliées les particules internes et la surface affichée de l'objet. Assez simples, ces méthodes permettent d'obtenir rapidement un affichage correct dissimulant à l'utilisateur les changements de discrétisation interne. Le retour haptique apporte un grand réalisme à la simulation et il convient de bien le régler pour que des discontinuités ne viennent pas perturber les sensations obtenues.

Le travail décrit dans ce chapitre a été réalisé en collaboration avec différents stagiaires intervenant dans le cadre des actions incitatives INRIA *AISIM* et *CAESARE*. Ils ont principalement contribué à l'intégration dans le simulateur des travaux réalisés dans le cadre de l'action incitative sur l'aspect rendu de l'organe simulé.

Pierre-Olivier Agliati a créé l'ossature du simulateur et y a intégré l'affichage des reflets d'une lampe virtuelle sur l'objet. Antoine Leroy et Sylvain Trimoreau ont durant leur stage ajouté la gestion des effets dynamiques (gouttes de sang, blanchiment, brûlures) qui peuvent apparaître sur la surface. Ils ont également modifié la structure de données de la surface (utilisation de *triangles fans* et simplification de maillage) et ont parallélisé le code.

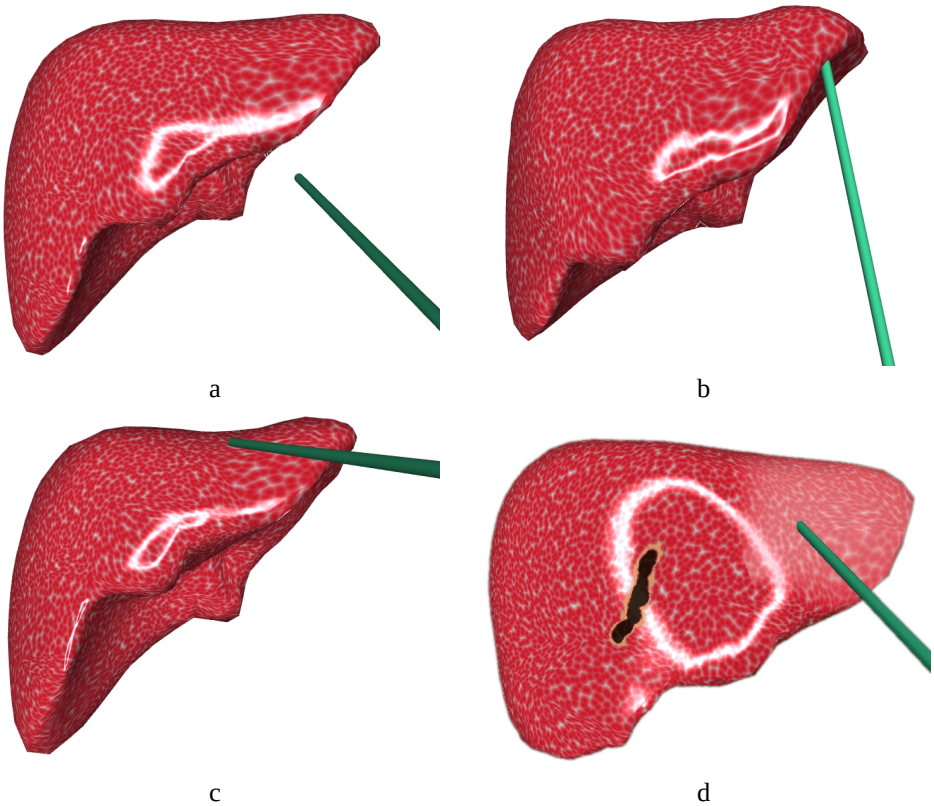
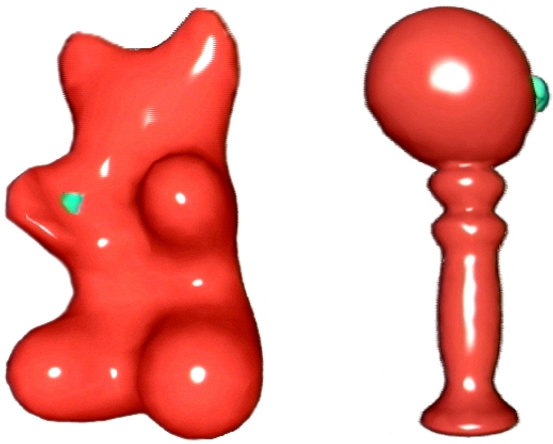
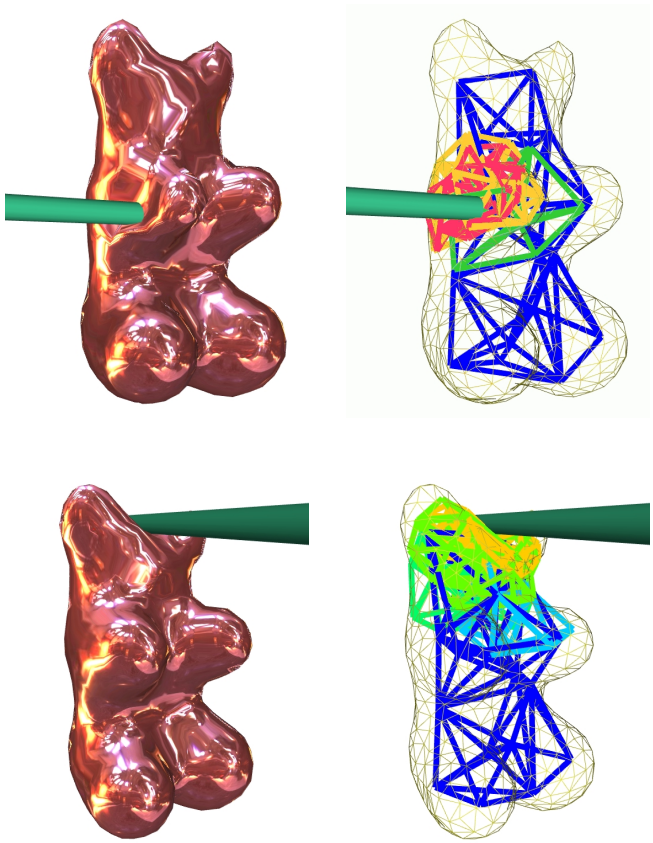


FIG. 6.6: Images du simulateur montrant quelques déformations du foie. La Figure (d) montre quelques effets de surface (brûlures, blanchiment, reflets).



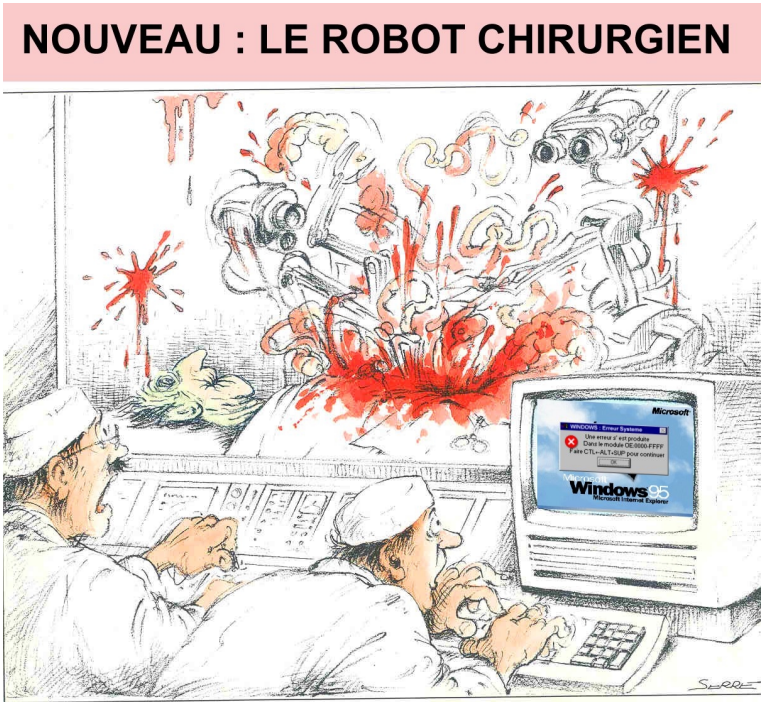
a b

FIG. 6.7: D'autres exemples d'animation, aux comportements plus ou moins rigides en fonction des paramètres choisis.



a b

FIG. 6.8: Le rendu final utilise des textures d'environnement qui explicitent la déformation subie (a). Le maillage interne correspondant (b).



"Intel products are not intended for use in medical, life saving, or life sustaining applications."

Intel Copyright Notice

Conclusion

APRÈS avoir décrit dans ce mémoire les différentes pistes de recherche que nous avons suivies, nous souhaitons dans ce chapitre dresser un bilan du travail réalisé durant cette thèse. Nous en résumerons les différentes contributions, pour ensuite dégager des pistes de recherche futures. Nous pourrons alors conclure et replacer ce travail dans un cadre plus général.

Résumé des contributions

Nous avons, lors de cette thèse, cherché à tirer parti de la multirésolution pour animer en images de synthèse les déformations d'objets mous. L'application principale de ces travaux, liée au caractère temps-réel de notre simulation, est principalement la mise au point d'un simulateur pédagogique de chirurgie laparoscopique.

Après avoir inventorié les techniques existantes et détaillé leur adéquation à nos buts, nous avons décrit le formalisme utilisé par les physiciens pour décrire un matériau continu. Ce formalisme permet en effet, grâce à sa relative indépendance à la discrétisation de l'objet, d'envisager une simulation multirésolution. Il faut pour cela pouvoir calculer, sur une grille arbitraire, la valeur d'opérateurs différentiels du second ordre qui donnent l'expression de la force subie par chaque particule en fonction des déformations de l'objet.

Premier modèle adaptatif

Nous avons pour cela mis au point un premier modèle multirésolution adaptatif. Le calcul des opérateurs était basé sur une extension en 3D des formulations issues des développements de Taylor 1D exprimés sur des points non régulièrement répartis.

Relativement insensibles à la discrétisation, ces opérateurs ont ensuite été utilisés dans un algorithme adaptatif basé sur une découpe en octree de l'espace. Les résultats visuels obtenus avec cette méthode sont assez convaincants, mais la méthode pêche mathématiquement, l'un des opérateurs étant incomplet.

Nouveaux opérateurs différentiels

Aussi avons-nous ensuite cherché à mettre au point de nouveaux opérateurs différentiels, cette fois ci basés sur des considérations géométriques (régions de Voronoï autour des particules) et mathématiques (utilisation du théorème de Gauss) plus pointues. Les démonstrations formelles de l'expression des opérateurs ont été suivies d'interprétations intuitives et géométriques de leur signification et de leur comportement.

Nous avons, en particulier, pu montrer l'*équivalence* de cette méthode avec celle des éléments finis linéaires dans le cas 2D, ce qui apporte un éclairage nouveau sur cette méthode. Ce rapprochement a permis d'établir que

le tenseur des éléments finis comportait des termes inutiles se compensant dans les calculs, probables sources de bruit et d'instabilités.

La comparaison en 3D a quand à elle montré la très grande ressemblance des deux modèles, leur légère différence n'expliquant pas le comportement divergent du terme de préservation de volume de la méthode. Nous avons par contre, inspirés par la formulation obtenue à partir de l'utilisation de Gauss et Voronoï, développé un formalisme d'éléments finis hybride novateur.

Nous terminons cette étude par une comparaison générale de différents modèles utilisés en animation. Apparaît alors l'intérêt de cette version hybride scalaire, ce modèle étant celui offrant la plus grande indépendance à la résolution.

Modèle multirésolution hiérarchique

Nous proposons alors un second modèle multirésolution, hiérarchique cette fois ci. Celui-ci s'appuie sur la cohabitation à tout instant de plusieurs maillages *indépendants*, animés grâce aux opérateurs précédemment décrits.

Chaque zone de l'objet est réellement simulée par un et un seul maillage, celui-ci étant choisi pour offrir le meilleur compromis entre qualité visuelle et temps de calcul. Les différents maillages qui composent l'objet peuvent néanmoins *communiquer* grâce à l'introduction de points fantômes servant d'interface, afin qu'ils forment un tout unifié.

Ces travaux ont permis la réalisation d'un prototype de simulateur chirurgical hépatique garantissant le temps-réel vrai. La liaison entre le modèle physique multirésolution et la surface de l'objet se fait par interpolation des positions des particules internes. Elle permet un affichage de qualité de la surface de l'organe, celle-ci répondant aux sollicitations de l'outil virtuel. Le retour d'effort, basé sur les valeurs physiques du modèle interne, permet d'augmenter sensiblement le réalisme du simulateur.

Travaux futurs

Aux quelques réponses et avancées apportées par cette thèse reviennent en écho de nouvelles interrogations, inhérentes à tout travail de recherche. Quelques améliorations pratiques peuvent être envisagées, comme la simulation des découpes de l'objet. La multirésolution pour un système dynamique pose quand à elle un problème de fond, qui s'est révélé ardu à mettre en œuvre.

Améliorations du modèle

Les travaux proposés visaient à la mise au point d'un modèle déformable temps-réel utilisable au sein d'un *prototype* de simulateur chirurgical. L'objectif est atteint, mais de nombreuses améliorations peuvent être apportées. Nous ne parlerons même pas des optimisations pouvant accélérer la simulation ou des modifications esthétiques du simulateur qui ne font pas directement partie de ce travail de recherche.

Le modèle multirésolution hiérarchique présenté au Chapitre 5 peut probablement être amélioré. Les choix arbitraires qui ont parfois été faits en raison de leur adéquation intuitive à nos buts (interpolations linéaires généralisées, octree non restreint) peuvent éventuellement être remis en cause.

Il peut arriver que le nombre de triangles composant la surface soit un facteur limitant, leur affichage prenant trop de temps. On pourra alors utiliser des niveaux de détails géométriques de la surface et il peut être intéressant d'essayer de les lier aux niveaux de détails du modèle physique. Les mouvements les plus subtils bénéficieront ainsi d'un rendu adéquat.

L'interface avec l'utilisateur peut également bénéficier d'améliorations sensibles. Si la détection des collisions semble bien traitée par l'algorithme utilisant le rendu *offscreen*, la *réponse* aux collisions peut encore poser problème. Lorsque l'outil est de grande taille il est très difficile de savoir comment déplacer les points

de la surface pour les sortir de façon naturelle de l'outil. Ce problème ne se pose pas trop dans le cadre d'une application chirurgicale, mais reste à régler si l'on souhaite utiliser cette technique à d'autres fins.

Le retour d'effort est également perfectible. Les possibles vibrations ou discontinuités que l'utilisateur peut ressentir détruisent en effet grandement le sentiment d'immersion que celui-ci aurait pu avoir. Il convient donc de consacrer beaucoup d'attention à ce problème, ce qui engendrera probablement des calculs supplémentaires.

L'intégration temporelle des forces calculées est encore trop sensible. Il est impressionnant de constater combien un pas de temps d'intégration donné conditionne les raideurs maximales permises. Le passage d'une animation inconditionnellement stable à une divergence immédiate se faisant lors d'une augmentation de quelques pour cent des coefficients.

Ce phénomène est particulièrement sensible lorsque l'on cherche à augmenter les forces dissipatives à l'intérieur de l'objet pour le rendre moins oscillant. Nous avons imposé que ces forces ne puissent pas accélérer une particule ou inverser sa vitesse. Le seuil de divergence s'en trouve repoussé, mais il reste des problèmes. On est ainsi encore trop limité par les raideurs, le pas de temps ou le nombre de particules simulées, les trois paramètres étant directement liés, si l'on cherche à assurer le temps-réel.

Une autre piste que nous n'avons pas exploré consiste à utiliser une intégration temporelle *semi-implicite*. Les méthodes de calcul de forces explicites permettent en effet d'avoir une bonne approximation du jacobien de la force. Ce jacobien pourrait être utilisé pour mieux prévoir la position suivante de chaque particule et pouvoir ainsi augmenter le pas de temps d'intégration sans risque de divergence [1].

Découpes de l'objet

Une importante partie du développement d'un simulateur chirurgical, non traitée lors de cette thèse, consiste à autoriser l'objet à subir des découpes. Le choix d'un modèle physique explicite permet d'envisager raisonnablement cette extension. Il suffit en effet de supprimer le coefficient gérant l'interaction entre deux particules pour simuler leur séparation. Cotin a dans sa thèse proposé une approche similaire, basée sur la suppression de certains des tétraèdres du maillage [Cot97].

La présence de découpes couplées à une simulation multirésolution est par contre un problème plus complexe. S'il est facile de supprimer des interactions entre les particules du maillage le plus fin, il reste à définir comment doivent interagir les particules des maillages plus grossiers, qui se voient ainsi partiellement séparées. Nous avions pensé simuler ce genre de phénomène en réduisant progressivement les coefficients d'interaction entre particules mères en fonction du nombre de particules filles séparées.

Se pose un nouveau problème lorsque l'on cherche à modifier la *topologie* de l'objet, en le séparant en plusieurs morceaux après découpe. Les différents morceaux peuvent en effet alors subir des rotations importantes et nous avons vu que le modèle de Cauchy s'y prête mal. L'introduction de *repères locaux*, par rapport auxquels est définie la position de l'objet peut être envisagée pour remédier à ce problème.

Ces repères sont capables d'absorber la partie rigide des déformations, ce qui limiterait l'amplitude des déformations gérées par le modèle physique et le rendrait donc plus stable. Ils pourraient être rattachés aux particules du niveau le plus grossier, qui donnent une indication de la position globale de l'objet.

Au delà de la mise au point d'un modèle physique, l'évolution de la *surface* de l'objet lors des découpes est un problème complexe. Faire apparaître des triangles au bon endroit en modifiant localement la géométrie est envisageable. Prendre en compte une petite découpe supplémentaire proche, sans ajouter à chaque fois de nouveaux triangles trop petits est plus difficile.

Ce problème est rendu encore plus complexe par la gestion des *textures* de la surface. Faire en sorte que les nouveaux triangles apparaissent avec une texture uniforme, sans remettre en cause celle des autres triangles est un problème ouvert.

Étude de la multirésolution dynamique

Cette thèse a également soulevé des problèmes directement liés à la multirésolution. Nous avons en effet pu détailler une dizaine de modèles physiques et comparer leur indépendance à la discrétisation de l'objet. Il en

[1] Voir Annexe B.

ressort que de très subtiles nuances (comme celle existant entre les éléments finis et la méthode à base de Voronoï) peuvent changer radicalement le comportement. S'il est facile d'obtenir des comportements grossièrement similaires à différentes résolutions, faire réellement correspondre les animations est très difficile.

La multirésolution *dynamique* est donc un phénomène subtil, qui ne se laisse pour l'instant approcher que par des méthodes au comportement simple. Utiliser davantage les connaissances des physiciens dans ce domaine devrait permettre d'étendre ce type de simulation à des comportements plus complexes.

On se heurte toutefois ici à un problème difficile, car dynamique. La très grande majorité des simulations cherchent à calculer l'état d'équilibre *statique* d'un système. On peut éventuellement atteindre cet état après plusieurs étapes de simulation, mais la simulation reste quasi-statique. La simulation dynamique est beaucoup moins étudiée, car souvent beaucoup trop gourmande en temps de calcul et l'on manque donc d'expérience en ce domaine.

La recherche d'une simulation temps-réel nous conduit également à grandement limiter le nombre de points d'échantillonage utilisés. Dans ces conditions, la qualité théorique des méthodes peut être prise en défaut, celles-ci s'appuyant parfois sur un phénomène d'échelle pour modéliser le comportement.

On a donc à trouver un compromis difficile entre deux buts antagonistes. On cherche d'une part à obtenir des comportements dynamiques très proches les uns des autres à différentes échelles, ce qui suppose une précision des calculs. On est d'autre part prêt à sacrifier un peu au réalisme de l'animation, pourvu que celle-ci soit suffisamment rapide.

On voit donc que multirésolution et rapidité d'exécution sont difficilement compatibles, et qu'il reste beaucoup de travail à faire avant de tirer pleinement parti des méthodes multirésolution dans le cadre de la simulation temps-réel.

Toutes ces pistes non explorées, ces choix non justifiés par des expériences, font que ce simulateur reste un prototype et pose au moins autant de nouvelles questions qu'il ne répond à d'autres.

Pour conclure

Nous aimerions terminer ce document par quelques réflexions d'ordre plus général et faire que cette thèse mérite son titre, au sens étymologique du terme tout du moins. L'animation en synthèse d'image est un domaine hybride entre l'image et la physique, et cette ambivalence peut poser problème.

On est amené, lorsque l'on cherche à imiter le réel, à utiliser des modèles physiques pour générer les mouvements. Une possibilité est de se plonger dans des ouvrages de physiques pour en extraire les équations adéquates. Reste alors à habiller les résultats de la simulation, en utilisant tout l'attrait de mouvements fluides et colorés en trois dimensions ou en invoquant au contraire la lourdeur ou la complexité des calculs mis en œuvre pour justifier la piètre qualité graphique des images générées.

Ce type d'approche n'est pas très enrichissante dans la mesure où la recherche en animation de synthèse semble alors se limiter à habiller des résultats connus des physiciens pour les présenter sous leur meilleur jour. Un récent auteur d'une publication à *Siggraph* me confiait ainsi avoir été troublé en réalisant que la première moitié de son article se composait d'une simple recopie d'équations complexes de la physique des fluides, sa contribution se limitant au rendu attractif des résultats numériques [YOH00].

La recherche en animation, si elle veut être autre chose qu'une "bête" mise en application de connaissances pratiques en création d'images numériques, doit chercher à tirer parti de sa spécificité. Ce n'est pas tant de précision que l'on a besoin, mais plus d'un effet visuel, d'un contrôle intuitif ou de rapidité de calcul.

À titre d'anecdote, lors de *Siggraph'99*, deux articles simulant les mouvements des fluides furent présentés. L'un utilisait les équations différentielles les plus pointues et nécessitait des heures de calcul [Wit99] alors que

l'autre faisait simplement en sorte de reproduire un comportement fluide, l'animation se déroulant en temps-réel [Sta99]. L'assistance a catégoriquement privilégié la seconde approche, les démonstrations ludiques et interactives emportant largement l'adhésion.

Il faut finalement plus chercher à *imiter* la nature pour laisser place à la nouveauté, à la simplification et pourquoi pas au caractère artistique. Il est certes difficile de copier ainsi l'essence du mouvement, mais c'est là que réside la partie recherche de notre domaine.

ANNEXE A

Rappels sur les opérateurs différentiels

Nous rappelons ici la définition des différents opérateurs différentiels que l'on peut appliquer à un champ vectoriel. Les définitions sont données pour un champ vectoriel dans l'espace, mais s'étendent de façon naturelle aux champs planaires par exemple.

On peut classer ces opérateurs d'après l'ordre (premier ou second) de leurs dérivées, ainsi que par la variable (temps ou espace) par rapport à laquelle on dérive.

1 Dérivées par rapport au temps

Ces dérivées sont simples et intuitives.

1.1 Opérateurs du premier ordre

Dériver une fois par rapport au temps transforme une position en vitesse, une vitesse en accélération.

$$\mathbf{p}_{,t} = \frac{\partial \mathbf{p}}{\partial t} = \begin{pmatrix} \frac{\partial p_x}{\partial t} \\ \frac{\partial p_y}{\partial t} \\ \frac{\partial p_z}{\partial t} \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \mathbf{v} \quad , \quad \mathbf{v}_{,t} = \frac{\partial \mathbf{v}}{\partial t} = \mathbf{a} \tag{A.1}$$

1.2 Opérateurs du second ordre

L'accélération est la dérivée seconde de la position par rapport au temps : $\mathbf{p}_{,tt} = \frac{\partial \mathbf{p}}{\partial^2 t} = \mathbf{a}$.

2 Dérivées par rapport à l'espace

2.1 Opérateurs du premier ordre

Ces opérateurs utilisent les dérivées premières du champ vectoriel. On peut dériver chacune des composantes d'un champ vectoriel par rapport à n'importe laquelle des directions. On a donc, en trois dimensions, 9 dérivées spatiales premières pour un champ vectoriel $\mathbf{v}$:

$$\mathbf{v}_{i,j} = \frac{\partial v_i}{\partial j} \quad , \quad (i,j) \in (x,y,z) \tag{A.2}$$

Parmi ces dérivées et leurs combinaisons, il en est trois particulières qui ont des interprétations précises.

– Le *gradient* d'un champ scalaire s est un vecteur défini par :

$$\mathbf{grad}\, s = \begin{pmatrix} s_{,x} \\ s_{,y} \\ s_{,z} \end{pmatrix} \tag{A.3}$$

Ce vecteur n'est autre qu'une extension de la classique dérivée d'une fonction à un espace de dimension supérieure. Il indique donc la pente locale de la fonction, le vecteur obtenu étant dirigé le long de la plus grande pente au champ s.
Par extension, on définit le gradient d'un champ vectoriel $\mathbf{v}$ comme la matrice 3×3 dont chaque ligne contient le gradient de la composante associée de $\mathbf{v}$:

$$\mathbf{grad}\, \mathbf{v} = \begin{pmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} & \frac{\partial v_x}{\partial z} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} & \frac{\partial v_y}{\partial z} \\ \frac{\partial v_z}{\partial x} & \frac{\partial v_z}{\partial y} & \frac{\partial v_z}{\partial z} \end{pmatrix} \tag{A.4}$$

– La *divergence* d'un champ vectoriel $\mathbf{v}$ est un scalaire défini par :

$$\mathrm{div}\, \mathbf{v} = v_{x,x} + v_{y,y} + v_{z,z} \tag{A.5}$$

Sommant les dérivées des composantes de $\mathbf{v}$ dans les trois directions, il peut s'interpréter comme un terme de mesure locale de l'expension (ou de la dilatation) du champ. Si on considère en effet un petit cube centré autour d'un point, le terme $v_{i,i}$ mesure comment les faces alignées avec l'axe i se sont déplacées dans la direction i. La divergence est une somme sur les trois directions de ces dilatations locales.
Par extension, on définit la divergence d'un tenseur α (matrice 3×3 symétrique) comme le vecteur de taille 3 dont chaque composante est la divergence de la ligne correspondante de la matrice :

$$\mathbf{div} \begin{pmatrix} \alpha_{1x} & \alpha_{1y} & \alpha_{1z} \\ \alpha_{2x} & \alpha_{2y} & \alpha_{2z} \\ \alpha_{3x} & \alpha_{3y} & \alpha_{3z} \end{pmatrix} = \mathbf{div} \begin{pmatrix} \boldsymbol{\alpha_1} \\ \boldsymbol{\alpha_2} \\ \boldsymbol{\alpha_3} \end{pmatrix} = \begin{pmatrix} \mathrm{div}\,(\boldsymbol{\alpha_1}) \\ \mathrm{div}\,(\boldsymbol{\alpha_2}) \\ \mathrm{div}\,(\boldsymbol{\alpha_3}) \end{pmatrix} \tag{A.6}$$

– Le *rotationnel* d'un champ vectoriel $\mathbf{v}$ est un vecteur défini par :

$$\mathbf{rot}\, \mathbf{v} = \begin{pmatrix} v_{y,z} - v_{z,y} \\ v_{z,x} - v_{x,z} \\ v_{x,y} - v_{y,x} \end{pmatrix} \tag{A.7}$$

Ce terme, aparenté à un produit vectoriel, est une approximation de l'axe de rotation des déformations locales du champ $\mathbf{v}$. On l'interprète facilement comme l'axe d'un tourbillon en mécanique des fluides, le champ $\mathbf{v}$ représentant la vitesse du liquide. Ce rotationnel est nul pour tous les champs $\mathbf{v}$ linéaires en les variables x, y et z.

2.2 Opérateur Nabla

L'opérateur nabla (∇) est pratique lorsqu'il s'agit d'écrire des dérivées premières. ∇ est un vecteur de coordonnées $(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})^T$. Les opérateurs précédents sont alors exprimés simplement par :

– $\mathbf{grad}\, s = \nabla s$
– $\mathrm{div}\ \mathbf{v} = \nabla \cdot \mathbf{v}$
– $\mathbf{rot}\, \mathbf{v} = \nabla \wedge \mathbf{v}$

où $\cdot$ et $\wedge$ représentent respectivement le produit scalaire et le produit vectoriel.

2.3 Opérateurs du second ordre

Les opérateurs du second ordre sont généralement des combinaisons de ceux du premier ordre. Nous utiliserons en particulier $\mathbf{grad}(\mathrm{div}\, \mathbf{v})$ et $\Delta \mathbf{v} = \mathrm{div}\, \mathbf{grad}\, \mathbf{v}$ (Δ est appelé l'opérateur *laplacien*). La $i^{ème}$ composante de ces opérateurs s'exprime par :

$$\mathbf{grad}(\mathrm{div}\, \mathbf{v})_i = v_{x,xi} + v_{y,yi} + v_{z,zi} \tag{A.8}$$

$$(\Delta v)_i = v_{i,xx} + v_{i,yy} + v_{i,zz} \tag{A.9}$$

ANNEXE B

Les méthodes d'intégration

Les méthodes présentées permettent de calculer dans une configuration donnée les forces en chaque point. Il existe différentes façons de les *intégrer* dans le temps et nous en présentons ici quelques unes.

1 Difficultés

On peut employer différentes méthodes d'intégration, toutes demandent d'adapter le pas de temps d'intégration à la valeur des accélérations. Si le pas de temps est trop petit, on perdra du temps de calcul alors que s'il est trop grand le système divergera.

La difficulté à intégrer dépend de l'amplitude de l'accélération *et* de sa direction. Un schéma de Newton-Cotes intégrera ainsi parfaitement une accélération constante (la gravité par exemple), quelle que soit son intensité. Que la direction de celle-ci change un peu au cours du temps et il faudra réduire de façon drastique le pas de temps.

2 Dépendance de l'intégration

En élasticité linéaire, on a, d'après le principe fondamental de la dynamique, une loi de la forme :

$$\mathbf{a} = \frac{\mathbf{f}}{m} = \frac{k.x}{m}$$

où k est la raideur du matériau et x une mesure de déplacement.

Un développement de Taylor donne la nouvelle position p' d'un point p comme égale à

$$\mathbf{p}' = \mathbf{p} + \mathbf{v}\,dt + \mathbf{a}\,dt^2 + O(dt^3) \tag{B.1}$$

pour un pas de temps dt donné.

La vitesse $\mathbf{v}$ de ce moment de la simulation est ce qu'elle est, mais on peut jouer sur les autres paramètres pour diminuer l'écart de position $\delta p = p' - p$ qui va être généré par l'intégration. Plus explicitement :

$$\delta p \simeq \mathbf{a}\,dt^2 = \frac{k\,x\,dt^2}{m} \tag{B.2}$$

Pour réduire cet écart et donc ne pas aller trop vite dans l'intégration, on voit donc que le plus efficace est de baisser le pas de temps puisque ce terme est quadratique dans l'équation précédente. L'autre moyen,

moins efficace consiste à prendre des matériaux mous et donc des raideurs faibles, à ne bouger que faiblement les points (ce qui peut revenir à modifier les unités pour avoir un objet de très petite taille) pour limiter les déplacements ou enfin à prendre des objets lourds avec plus d'inertie.

Une autre remarque tirée de [Mes97] est que si l'on augmente le nombre de points de la discrétisation, on va accroître l'instabilité. La masse de chaque point devra en effet diminuer pour que la masse totale soit conservée. Or la fréquence d'oscillation correspondant à la solution de l'Équation B.2 est $\sqrt{\frac{k}{m}}$ et va donc s'en trouver augmentée, ce qui demandera un plus faible pas de temps pour une intégration correcte.

Cette remarque rejoint le critère de stabilité de Courant-Friedrichs-Lewy, qui stipule que la distance parcourue par une onde se propageant dans le matériau doit être inférieure à la distance entre deux points de discrétisation. On comprend intuitivement que la propagation de cette onde ne pourrait être bien calculée si elle pouvait ainsi "sauter" une maille.

En augmentant le nombre de points, on réduit la taille du maillage et il faut donc diminuer la distance parcourue par l'onde (dont la vitesse est proportionelle à $\sqrt{\frac{k}{m}}$) entre deux intégrations et donc baisser le pas de temps.

3 Intégration explicite

Les méthodes d'intégration explicites sont les plus simples et s'appuient directement sur le développement de Taylor donné plus haut (Eq. B.1). La plus simple d'entre elles est la méthode d'*Euler*, qui n'utilise que des développements limités d'ordre 1 pour écrire :

$$\mathbf{v}_{t+dt} = \mathbf{v}_t + \mathbf{a}_t\, dt \tag{B.3}$$
$$\mathbf{p}_{t+dt} = \mathbf{p}_t + \mathbf{v}_t\, dt \tag{B.4}$$

Simpliste, elle offre néanmoins parfois de bons résultats.

En faisant dépendre la nouvelle position de la *nouvelle* vitesse et non plus de celle du pas précédent, on obtient le schéma d'*Euler modifié*. En pratique beaucoup plus stable que le précédent. Il a été démontré comme étant d'ordre 4 par Provost dans sa thèse [Pro97] et donne souvent des résultats supérieurs à ceux de Runge-Kutta 2 (voir *supra*).

$$\mathbf{v}_{t+dt} = \mathbf{v}_t + \mathbf{a}_t\, dt \tag{B.5}$$
$$\mathbf{p}_{t+dt} = \mathbf{p}_t + \mathbf{v}_{t+dt}\, dt \tag{B.6}$$

On pourra parfois lui préférer la méthode de *Newton-Cotes* :

$$\mathbf{v}_{t+dt} = \mathbf{v}_t + \mathbf{a}_t\, dt \tag{B.7}$$
$$\mathbf{p}_{t+dt} = \mathbf{p}_t + \mathbf{v}_t\, dt + \mathbf{a}\, dt^2 \tag{B.8}$$

qui présente l'intéressant avantage d'intégrer correctement les accélérations constantes (la gravité par exemple).

Citons aussi la méthode de *Stoermer* qui donne de bons résultats, même avec de fortes raideurs :

$$\mathbf{b}_{t+dt} = \mathbf{b}_t + \mathbf{a}_t\, dt^2 \tag{B.9}$$
$$\mathbf{p}_{t+dt} = \mathbf{p}_t + \mathbf{b} \tag{B.10}$$

où **b** est un vecteur servant d'accumulateur initialisé à 0 au départ.

Il existe des méthodes d'ordres plus élevés, offrant plus de précision, mais nécessitant en contrepartie le calcul de la force pour des positions intermédiaires entre t et $t+dt$. L'ordre de ces méthodes dépend du nombre de calculs de forces supplémentaires à effectuer. Les plus utilisées sont celles de Runge-Kutta, d'ordre 2 et 4 (voir [PTVF92] pour plus de détails). Plus chères, elles sont également mal appropriées aux simulations dynamiques faisant apparaître des collisions et où apparaissent des discontinuités dans le mouvement.

4 Intégration implicite

L'intégration implicite fut remise au goût du jour par Baraff et Witkin dans [BW98]. Elle donne d'excellents résultats pour des raideurs élevées que l'intégration explicite ne sait gérer, ou au prix d'un pas de temps rédhibitoire.

L'idée consiste à ne plus trouver la nouvelle position en fonction des forces connues à un instant, mais à chercher la position où les forces à $t + dt$ *auraient conduits*, si on les avait intégrées en partant de la position courante. Plutôt qu'un bond en aveugle dans le futur, on cherche un état où les forces sont compatibles avec la position. Mathématiquement, Euler deviendrait :

$$\mathbf{v}_{t+dt} = \mathbf{v}_t + \mathbf{a}_{t+\mathbf{dt}}\, dt \tag{B.11}$$

$$\mathbf{p}_{t+dt} = \mathbf{p}_t + \mathbf{v}_{t+dt}\, dt \tag{B.12}$$

Cela demande donc de connaître $\mathbf{a}_{t+dt}$ pour trouver $\mathbf{p}_{t+dt}$, ce qui parait impossible puisque la force dépend elle même de la position.

On écrit alors un développement limité à l'ordre 1 de l'expression de la force en fonction de la position

$$\mathbf{f}' = \mathbf{f} + \frac{\partial \mathbf{f}}{\partial \mathbf{p}}\, d\mathbf{p}$$

qui va permettre de connaître approximativement la force d'une position proche. Puisque la force dépend de la position d'un point *et* de celle de ses voisins, l'expression $\frac{\partial \mathbf{f}}{\partial \mathbf{p}}$ va être représentée par une matrice qu'il va falloir inverser.

Le coût de l'inversion de la matrice à chaque pas de temps est largement compensé par le gain en stabilité de la méthode et par le pas de temps qui peut alors être employé, en comparaison avec le pas de temps infime nécessaire avec de fortes raideurs dans une intégration explicite.

Il est à noter que les méthodes implicites avaient déjà été utilisée par Terzopoulos *et al.* dans leur article de 1987 [TPBF87].

Citons pour information deux articles qui ont été cités comme utilisant des éléments finis et une intégration implicite [OPT83, HLW83], mais publiés dans des revues de physique, non disponibles sur la toile et que nous n'avons pu nous procurer.

Desbrun a repris cette méthode en l'accélérant jusqu'à l'obtention de temps réel en approximant la matrice jacobienne $\frac{\partial \mathbf{f}}{\partial \mathbf{p}}$ comme constante (ce qui revient à considérer les ressorts comme ayant une longueur à vide nulle) et donc pré-inversible [DSB99]. Desbrun montre que cela revient à ajouter une viscosité artificielle (force inversement proportionnelle aux vitesses, modulée par un coefficient dépendant de la raideur et du pas de temps) et à appliquer un filtrage sur les forces appliquées aux particules.

Les simplifications entraînées par cette hypothèse sont corrigées en ajoutant des forces qui vont corriger le moment d'inertie de l'objet, suivies d'un post-traitement itératif limitant la longueur de chacun des ressorts.

Une meilleure gestion des collisions est introduite dans [MDDB00], permettant une simulation très fluide du mouvement d'un tissus. L'intégration implicite est une méthode à la mode et d'autres développements, que nous jugeons plus douteux (faux) dans leurs fondements mathématiques, ont été présentés dans [EEH00].

ANNEXE C

Green-Lagrange en pratique

Nous décrivons ici brièvement les formules nécessaires à l'implémentation d'un modèle déformable utilisant le formalisme de Green-Lagrange, dans un cadre d'éléments finis explicites. L'objet est supposé maillé à l'aide de tétraèdres, que l'on aura pris soin de générer aussi réguliers que possible pour des raisons d'efficacité et de stabilité numérique.

Après un rappel rapide de notions d'élasticité, nous donnons les formules qui permettront d'implémenter ce formalisme.

1 Rappels d'élasticité

Nous reprenons ici les principales équations décrivant l'élasticité des objets déformables. On se reportera à la Section 1.2 du Chapitre 2 pour une description plus complète.

Le *tenseur des déformations* ε est défini en tout point du matériau. Le terme situé en (i, j) a pour expression, dans le cas de Green-Lagrange :

$$(\varepsilon)_{ij} = \left(\frac{\partial \mathbf{x}}{\partial \Omega_i} \cdot \frac{\partial \mathbf{x}}{\partial \Omega_j} \right) - \delta_{ij} \tag{C.1}$$

où δ_{ij} est le symbole de Kronecker ($\delta_{ij} = 1$ si $i = j$ et 0 sinon).

Ω représente le vecteur de $\Re^3$ représentant les coordonnées d'un point dans le repère lié au matériau. $\mathbf{x}(\Omega)$ représente la position d'un point dans le repère du monde en fonction de ses coordonnées matérielles. Les deux systèmes de coordonnées se correspondent (à une transformation rigide près, voir la Section 1.2 du Chapitre 2) lorsque l'objet est au repos, créant ainsi un tenseur des déformations nul.

$\mathbf{x}(\Omega)$ est une fonction continue qui varie en fonction des déformations de l'objet (les coordonnées matérielles Ω d'un point sont fixes, mais sa position dans l'espace $\mathbf{x}$ variera au cours du temps).

Le *tenseur des contraintes* σ représente la distribution de force à l'intérieur du matériau. Tout comme ε, on peut l'approximer au premier ordre par une matrice 3×3 . La force $\mathbf{f}$ agissant sur une surface élémentaire dS, de normale unitaire sortante $\mathbf{n}$ est : $\mathbf{f} = \sigma\, \mathbf{n}\ \ dS$.

La théorie de l'élasticité linéaire suppose que déformations et contraintes sont linéairement liées (comme dans un ressort idéal). Si le matériau est considéré comme isotrope, des raisons de symétrie font que seuls 2

coefficients décrivent le comportement du matériau :

$$\sigma \;=\; \lambda \,\mathrm{tr}\,(\varepsilon)\, I_3 \;+\; 2\,\mu\,\varepsilon \tag{C.2}$$

où I_3 est la matrice identité 3×3 et tr (ε) la trace de ε. μ et λ sont les coefficients de Lamé : μ représente la rigidité du matériau tandis que λ mesure son incompressibilité (un matériau incompressible a théoriquement un λ infini).

L'énergie potentielle élastique E est définie en tout point du matériau par :

$$E = \frac{1}{2}\sum_{i=1}^{3}\sum_{j=1}^{3} \sigma_{ij}\,\varepsilon_{ij} \tag{C.3}$$

Sa dérivée par rapport à la position du point créera la force qui sera appliquée en chaque point.

2 Forces dissipatives

L'ajout de forces dissipatives est simple et cohérent avec la description faite des précédents tenseurs. Le *tenseur des taux de déformations* ν est la dérivée par rapport au temps de ε (Eq. C.1) :

$$\nu = \frac{\partial\varepsilon}{\partial t}, \qquad (\nu)_{ij} = \left(\frac{\partial\mathbf{v}}{\partial\Omega_i}\cdot\frac{\partial\mathbf{p}}{\partial\Omega_j}\right) + \left(\frac{\partial\mathbf{p}}{\partial\Omega_i}\cdot\frac{\partial\mathbf{v}}{\partial\Omega_j}\right) \tag{C.4}$$

où $\mathbf{v} = \frac{\partial\mathbf{x}}{\partial t}$ est la vitesse d'un point.

σ est mis à jour grâce à une équations identique à celle utilisée pour ε (Eq. C.2) :

$$\sigma \mathrel{+}= \phi\,\mathrm{tr}\,(\nu)\, I_3 \;+\; 2\,\psi\,\nu \tag{C.5}$$

où ϕ et ψ contrôlent la vitesse avec laquelle le matériau perd son énergie cinétique. Un mouvement rigide de l'objet ne sera pas atténué avec cette formulation ($\nu \equiv 0_3$) qui ne dissipe que les vibrations internes.

3 Éléments finis explicites

Nous décrivons ici les équations des éléments finis explicites, appliqués à des maillages tétraédriques d'objets 3D. Au lieu de regrouper toutes les équations relatives à tous les points dans un grand système matriciel, comme dans les éléments finis classiques, les éléments finis *explicites* résolvent les équations relatives à chaque élément *séparément*. Cette approximation locale réduit énormément la taille du système à résoudre et donc le temps de calcul, au prix d'une perte en précision. Pour qu'une information (un déplacement imposé par l'utilisateur par exemple) se propage entre deux éléments, il faudra en particulier maintenant autant d'itérations qu'il y a d'éléments entre ces deux éléments. L'obtention de méthodes de calcul temps-réel doit se faire au prix de ces approximations.

Nous avons choisi des éléments tétraédriques et des fonctions de bases linéaires (voir Chapitre 1, Section 7) pour leur très bon compromis vitesse-efficacité. Pour chaque élément E, la fonction de base linéaire $a^i x + b^i y + c^i z + d^i$ associée à un sommet i de E peut être représentée [1] comme un vecteur $\mathbf{L^i} = (a^i\ b^i\ c^i\ d^i)$.

Par définition des fonctions de base linéaires, appliquer $\mathbf{L^i}$ à un point situé à l'intérieur de l'élément tétraédrique E donne ses coordonnées barycentriques à l'intérieur du tétraèdre. La coordonnée barycentrique d'un sommet i valant 1 au somment i et 0 aux autres sommets, on a :

$$\mathbf{L^i}\cdot\begin{pmatrix}\Omega^j\\ 1\end{pmatrix} = \delta_{ij} \;\; \forall\, i,j \in 1..4 \tag{C.6}$$

Pour chaque somment i, ces quatre équations ($j \in 1..4$) forment un système linéaire qui peut être inversé si l'élément E n'est pas dégénéré (c'est à dire aplati, ce qui n'est pas le cas avec notre méthode par construction).

[1] Les exposants représentent la valeur en un sommet donné ($\mathbf{c^i}, \Omega^j$) tandis que les indices représentent les composantes d'un vecteur ($\Omega_1 = \Omega_x, \Omega_2 = \Omega_y, \Omega_3 = \Omega_z$).

La fonction d'interpolation résultante $\mathbf{L^i}$ va nous permettre de calculer les dérivées exprimées dans le repère associé au matériau.

Tout champ vectoriel $\mathbf{c}$ peut être linéairement interpolé sur l'élément E en utilisant les coordonnées barycentriques. Pour un point situé à l'intérieur de l'élément de coordonnées Ω, la valeur de $\mathbf{c}(\Omega)$ est :

$$\mathbf{c}(\Omega) = \sum_{i=1}^{4} \mathbf{c^i} \ \ \mathbf{L^i} \cdot \begin{pmatrix} \Omega \\ 1 \end{pmatrix} \tag{C.7}$$

Les dérivées (dans le repère lié au matériau) peuvent alors être exprimées par :

$$\frac{\partial \mathbf{c}}{\partial \Omega_j} = \sum_{i=1}^{4} \mathbf{c^i} \ \ (\mathbf{L^i})_j \tag{C.8}$$

En utilisant cette équation aux champs vectoriels position et vitesse, on est capable de calculer en chaque sommet $\frac{\partial \mathbf{x}}{\partial \Omega_i}$ et $\frac{\partial \mathbf{v}}{\partial \Omega_i}$, et donc la valeur des tenseurs ε, ν et σ décrits précédemment.

La force au sommet i résultant de la déformation de E s'exprime par (voir [OH99]) :

$$\mathbf{f^i} = -\frac{vol^{\,E}}{2} \ \sum_{j=1}^{4} \mathbf{x^j} \ \sum_{k=1}^{3} \sum_{l=1}^{3} \ (\mathbf{L^i})_k \ (\mathbf{L^j})_l \ (\sigma)_{kl} \tag{C.9}$$

où $vol^{\,E}$ est le volume du tétraèdre E.

Tous les calculs présentés l'ont été pour un élément E donné, et s'appliquent à ses quatre sommets. Il suffit ensuite de répéter ces calculs pour chacun des éléments qui composent l'objet, chaque sommet *sommant* les contributions (*i.e.* la force résultante) de tous les tétraèdres auquel il appartient.

TABLE DES MATIÈRES

TABLE DES FIGURES

BIBLIOGRAPHIE

[AH97] Olivier R. Astley and Vincent Hayward. Real-time finite-elements simulation of general visco-elastic materials for haptic presentation. In *International Conference on Intelligent Robots and Systems*, volume IROS'97 September 7-11, Grenoble, pages 52–57, 1997. (référencé en page 29)

[Arn88] Bruno Arnaldi. *Conception du noyau d'un système d'animation de scènes tridimentionelles intégrant des lois de la mécanique*. PhD thesis, Université de Rennes 1, 1988. (référencé en page 55)

[Bar84] Alan H. Barr. Global and local deformations of solid primitives. *Computer Graphics*, 18(3) :21–29, 1984. (référencé aux pages 12, 14)

[Bat73] G. K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, 1973. (référencé en page 18)

[Bat96] K-J. Bathe. *Finite Element Proceedures*. Prentice Hall, Englewood Cliffs, NJ, 1996. (référencé en page 20)

[BB89] R. Bartels and J. Beaty. A technique for the direct manipulation of splines curves. In *Proceedings of Graphics Interface*, 1989. (référencé en page 12)

[BBB87] R. Bartels, J. Beaty, and B. Barski. *An introduction to splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, Los Altos, 1987. (référencé en page 11)

[BC00] David Bourguignon and Marie-Paule Cani. Controlling anisotropy in mass-spring systems. In *Proceedings of the 11th Eurographics Workshop on Animation and Simulation*, pages 113–123. Springer-Verlag, August 2000. (référencé en page 15)

[BdCL00] François Boux de Casson and Christian Laugier. Simulating 2D tearing phenomena for interactive medical surgery simulators. In *Proceedings of Computer Animation*, Philadelphia, PA (US), May 2000. (référencé en page 15)

[Ber97] Roman Berka. Reduction of computations in physics-based animation using level of detail. In *Proceedings of the 1997 Spring Conference on Computer Graphics*, pages 69–76, 1997. (référencé en page 28)

[BHS+98] C. Basdogan, C. Ho, M.A. Srinivasan, S.D. Small, and S.L. Dawson. Force interactions in laparoscopic simulations : Haptic rendering of soft tissues. In *Medicine Meets Virtual Reality (MMVR'6) Conference, San Diego, CA*, pages 19–22, Jan 1998. (référencé en page 26)

[Blo87] J. Bloomenthal. Polygonization of implicit surfaces. *Xerox Technical Report CSL-87-2*, pages 1–19, May 1987. (référencé en page 13)

[Blo88] Jules Bloomenthal. Polygonisation of implicit surfaces. *Computer Aided Geometric Design*, 5 :341–355, 1988. (référencé en page 13)

[BNC96] Morten Bro-Nielsen and Stéphane Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Eurographics*, pages 21–30, 1996. (référencé aux pages 22, 24, 26)

[BS95] Jean Braun and Malcolm Sambridge. A numerical method for solving partial differential equations on highly irregular evolving grids. *Nature*, 376 :655–660, August 1995. (référencé en page 64)

[BW76] Nicholas Burtnyk and Mark Wein. Interactive skeleton technique for enhancing motion dynamics in key frame animation. *Communications of the ACM*, 19(10) :564–569, October 1976. (référencé en page 25)

[BW90] Jules Bloomenthal and Brian Wyvill. Interactive techniques for implicit modeling. *Proceedings of Symposium on Interactive 3D Graphics*, 24(2) :109–116, March 1990. (référencé en page 13)

[BW92] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. *Proceedings of SIGGRAPH'92 (Chicago, Illinois, July 1992)*, 26(2) :303–308, July 1992. (référencé en page 14)

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. In Michael Cohen, editor, *SIGGRAPH'98 Conference Proceedings*, Annual Conference Series, pages 43–54. ACM SIGGRAPH, Addison Wesley, July 1998. (référencé aux pages 7, 15, 26, 121)

[CCOS91] J. Collier, B. Collier, G. O'Toole, and S. Sargand. Drape prediction by means of finite elements analysis. In *Journal of the Textile Institute*, volume 82(1), pages 96–107, 1991. (référencé en page 22)

[CDA96] Stéphane Cotin, Hervé Delingette, and Nicolas Ayache. Real time volumetric deformable models for surgery simulation. In Lectures Notes in Computer Science, editor, *Proceedings of Visualization in Biomedical Computing*, volume 11, September 1996. (référencé en page 23)

[CDA98] Stéphane Cotin, Hervé Delingette, and Nicolas Ayache. Efficient linear elastic models of soft tissues for real-time surgery simulation. Technical Report RR-3510, INRIA, 1998. (référencé en page 23)

[CDA99] Stéphane Cotin, Hervé Delingette, and Nicolas Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions On Visualization and Computer Graphics*, 5(1) :62–73, January-March 1999. (référencé en page 23)

[CEO93] Steven A. Cover, Norberto F. Ezquerra, and James F. O'Brien. Interactively deformable models for surgery simulation. *IEEE Computer Graphics and Application*, 13(6) :68–75, November 1993. (référencé en page 26)

[CF97] Stephen Chenney and David Forsyth. View-dependent culling of dynamics systems in virtual environments. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 55–58, 1997. (référencé en page 28)

[CF00] Stephen Chenney and D. Forsyth. Sampling plausible solutions to multi-body constraint problems. *Proceedings of SIGGRAPH'2000 (New-Orleans, Louisiana*, pages 209–218, July 2000. (référencé en page 7)

[CGD97] Marie-Paule Cani-Gascuel and Mathieu Desbrun. Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 3(1) :39–50, March 1997. (référencé aux pages 17, 25, 25)

[CH97] Deborah Carlson and Jessica K. Hodgins. Simulation levels of detail for real-time animation. In *Proceedings of Graphic Interface*, pages 1–8, 1997. (référencé aux pages 27, 28)

[CHP89] John E. Chadwick, David R. Haumann, and Richard E. Parent. Layered construction for deformable animated characters. *Computer Graphics*, 23(3) :243–252, July 1989. (référencé aux pages 15, 25)

[Cia85] Ciarlet. *Elasticité Tridimensionnelle*. Masson, 1985. (référencé en page 33)

[CMP89] Robert D. Cook, David S. Malkus, and Michael E. Plesha. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, January 1989. ISBN : 0471847887. (référencé en page 33)

[Coq90] Sabine Coquillart. Extending free-form deformation : a sculpting tool for 3d geometric modeling. *Proceedings of SIGGRAPH'90*, pages 187–196, 1990. (référencé en page 12)

[Cot97] Stéphane Cotin. *Modèles anatomiques déformables en temps-réel*. PhD thesis, Epidaure-Sophia Antipolis, 1997. (référencé aux pages 11, 23, 24, 26, 69, 77, 107, 113)

[Coz96] Rémi Cozot. *Environnement de Simulation de systèmes physiques : Modèle et langage*. PhD thesis, Université de Rennes, Juillet 1996. (référencé aux pages 17, 22)

[CP88] John .E. Chadwick and Eric Parent. Critter construction : Developping characters for computer animation. In *Proceedings of PIXIM'88*, pages 283–305, Paris, France, October 1988. (référencé en page 25)

[CZ92] David T. Chen and David Zeltzer. Pump it up : Computer animation of a biomechanically based model of muscle using the finite element method. In Edwin E. Catmull, editor, *SIGGRAPH'92 Conference Proceedings*, Annual Conference Series, pages 89–98. ACM SIGGRAPH, Addison Wesley, July 1992. (référencé en page 22)

[Dar95] Denis Dartus. *Elasticité linéaire*. Cepadues-Editions, 1995. (référencé en page 33)

[dBvKOS97] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry*. Verlag, 1997. (référencé en page 64)

[DC99a] Gilles Debunne and Marie-Paule Cani. Animation multiresolution interactive d'objets déformables. In *12ème journées de l'Association Française d'Informatique Graphique, Reims*, nov 1999. (référencé en page 61)

[DC99b] Mathieu Desbrun and Marie-Paule Cani. Space-time adaptive simulation of highly deformable substances. *INRIA Technical Report*, RR-3829 (`http ://www.inria.fr/RRRT/RR-3829.html`), December 1999. (référencé aux pages 18, 19, 29)

[DCA99] Hervé Delingette, Stéphane Cotin, and Nicolas Ayache. A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. In *Computer Animation*, Geneva Switzerland, May 26-28 1999. (référencé en page 24)

[DCG96] Mathieu Desbrun and Marie-Paule Cani-Gascuel. Smoothed particles : A new approach for animating highly deformable bodies. In Springer Computer Science, editor, *7th Eurographics Workshop on Animation and Simulation*, pages 61–76, Poitiers, France, September 1996. (référencé aux pages 29, 49)

[DDBC99] Gilles Debunne, Mathieu Desbrun, Alan Barr, and Marie-Paule Cani. Interactive multiresolution animation of deformable models. In *10th Eurographics Workshop on Computer Animation and Simulation, Milano, Italy*, 1999. (référencé aux pages 45, 61)

[DDCB00] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan Barr. Adaptive simulation of soft bodies in real-time. In *Computer Animation, Philadelphia, USA*, may 2000. (référencé aux pages 69, 83, 91)

[DDCB01] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space and time adaptive sampling. In *Computer Graphics Proceedings*, Annual Conference Series. ACM Press / ACM SIGGRAPH, aug 2001. Proceedings of SIGGRAPH'01. (référencé en page 91)

[Des97] Mathieu Desbrun. *Modélisation et animation d'objets hautement déformables à l'aide de surfaces implicites*. PhD thesis, Institut National Polytechnique, Grenoble, December 1997. (référencé aux pages 11, 17, 18, 29, 55)

[Dil96] Gary A. Dilts. Equivalence of the SPH Method and a Space-Time Galerkin Moving Particle Method. Technical Report LA-UR 96-134, Los Alamos National Laboratory, January 1996. (référencé en page 19)

[DKT95] Oliver Deussen, Leif Kobbelt, and Peter Tucke. Using simulated annealing to obtain good nodal approximations of deformable objects. In *Proceedings of the 6th Eurographics Workshop on Animation and Simulation*, pages 30–43. Springer-Verlag, September 1995. (référencé en page 16)

[DMSB99] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH 99 Conference Proceedings*, 1999. Los Angeles, CA. (référencé en page 46)

[DP90] Libersky L. D. and A. G. Petschek. Smooth particle hydrodynamics with strength of materials, advances in the free-lagrange method. *Trease, Fritts, and Crowley, eds., Springer Verlag*, pages 248–257, 1990. (référencé en page 31)

[DSB99] Mathieu Desbrun, Peter Schröder, and Alan Barr. Interactive animation of structured deformable objects. In *Graphics Interface'99 proceedings*, pages 1–8, June 1999. (référencé aux pages 26, 98, 121)

[DTG96] Mathieu Desbrun, Nicolas Tsingos, and Marie-Paule Gascuel. Adaptive sampling of implicit surfaces for interactive modelling and animation. *Computer Graphics Forum*, 15(5) :319–327, December 1996. A preliminary version of this paper appeared in *Implicit Surfaces'95*, Grenoble, France, may 1995. (référencé en page 13)

[EEH00] Bernhard Eberhardt, Olaf Etzmuß, and Michael Hauth. Implicit-explicit schemes for fast animation with particle systems. In N. Magnenat-Thalmann, D. Thalmann, and B. Arnaldi, editors, *11th Eurographics Workshop on Computer Animation and Simulation, Interlaken, Swizerlan*, August 2000. ISBN 3-211-83549-0. (référencé en page 121)

[Far90] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press Inc., San Diego, 1990. Second edition. (référencé en page 11)

[FM96] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Proceedings of Graphics Interface*, pages 204–212, May 1996. (référencé en page 7)

[For88] Bengt Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Math. Comput.*, 51 :699–706, 1988. (référencé en page 46)

[FS94] A. Finkelstein and D. Salezin. Multiresolution curves. In *Proceedings of SIGGRAPH'94*, Computer Graphics Proceedings, Annual Conference Series, pages 262–268, 1994. (référencé en page 12)

[Fuj95] Koji Fujiwara. Eigenvalues of laplacians on a closed riemannian manifold and its nets. In *Proceedings of the AMS 123*, pages 2585–2594, 1995. (référencé en page 46)

[Fun65] Y.C. Fung. *Foundations of Solids Mechanics*. Prentice-Hall, 1965. (référencé en page 33)

[Gas90] Marie-Paule Gascuel. Déformations de surfaces complexes : techniques de haut niveau pour la modélisation et l'animation. *Thèse de doctorat*, Université Paris XI, October 1990. (référencé en page 25)

[GCD+98] Jean-Dominique Gascuel, Marie-Paule Cani, Mathieu Desbrun, Eric Leroi, and Carola Mirgon. Simulating landslides for natural disaster prevention. In *Proceedings of the 8th Eurographics Workshop on Animation and Simulation*, 1998. (référencé en page 29)

[GCMS00] Fabio Ganovelli, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. A multiresolution model for soft objects supporting interactive cuts and lacerations. In *Proceedings of Eurographics 2000*, 2000. (référencé en page 29)

[GCS99] Fabio Ganovelli, Paolo Cignoni, and Roberto Scopigno. Introducing multiresolution representation in deformable object modeling. In *Proceedings of SCCG99*, pages 149–158, 1999. (référencé en page 29)

[Gel98] Allen Van Gelder. Approximate simulation of elastic membranes by triangulated spring meshes. *Journal of Graphics Tools*, 1998. (référencé aux pages 29, 30, 84, 86, 88, 133)

[Ger62] Paul Germain. *Mécanique des milieux continus*. Masson, 1962. (référencé en page 33)

[GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In Turner Whitted, editor, *SIGGRAPH'97 Conference Proceedings*, Annual Conference Series, pages 209–216. Addison Wesley, August 1997. ISBN 0-89791-896-7, http ://www.cs.cmu.edu/g̃arland/multires/my-work.html. (référencé en page 92)

[Gib97] Sarah F. Gibson. 3d chainmail : a fast algorithm for deforming volumetric objects. In Michael Cohen and David Zeltzer, editors, *Symposium on interactive 3D graphics*, pages 149–154, April 1997. ACM SIGGRAPH. (référencé en page 26)

[GM97] Sarah F. Gibson and Brian Mirtich. A survey of deformable modeling in computer graphics. Technical report, MERL, 1997. (référencé en page 11)

[GMTT89] Jean-Paul Gourret, Nadia Magnenat-Thalmann, and Daniel Thalmann. Simulation of object and human skin deformation in a grasping task. *Proceedings of SIGGRAPH'89 (Boston, MA)*, pages 21–30, July 1989. (référencé aux pages 22, 23)

[GVP90] Marie-Paule Gascuel, Anne Verroust, and Claude Puech. Animation with collisions of deformable articulated bodies. In *Eurographics Workshop on Animation and Simulation*, September 1990. (référencé en page 25)

[GVP91] Marie-Paule Gascuel, Anne Verroust, and Claude Puech. A modeling system for complex deformable bodies suited to animation and collision processing. *Journal of Visualization and Computer Animation*, 2(3) :82–91, August 1991. A shorter version of this paper appeared in *Graphics Interface'91*. (référencé en page 25)

[HHK92] W. Hsu, J. Hughes, and H. Kaufmann. Direct manipulation of free-form deformations. *Proceedings of SIGGRAPH'92*, pages 177–184, 1992. (référencé en page 12)

[HK89] Lars Hernquist and Neal Katz. Treesph : A unification of sph with the hierarchical tree method. *Astronony Astrophysics Suppl. Series*, 70 :419–446, June 1989. (référencé en page 18)

[HLW83] T. J. R. Hughes, I. Levit, and J. Winget. An element-by-element solution algorithm for problems of structural and solid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 36(2) :241–254, 1983. (référencé en page 121)

[HPH96] Dave Hutchinson, Martin Preston, and Terry Hewitt. Adaptative refinement for mass/spring simulations. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation*, pages 31–45, Poitiers, France, sep 1996. (référencé aux pages 16, 27, 29, 91)

[INR] INRIA. Aisim - caesare. http ://www-sop.inria.fr/epidaure/AISIM/. (référencé aux pages 26, 105)

[Jim93] Stéphane Jimenez. Modélisation et simulation physique d'objets volumiques déformables complexes. Thèse de doctorat, Institut National Polytechnique de Grenoble, November 1993. (référencé en page 16)

[Jou96] Ammar Joukhadar. Adaptive time step for fast converging dynamic simulation system. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, volume 2, pages 418–424, November 1996. (référencé aux pages 15, 55)

[JP99] Doug James and Dinesh Pai. Art defo - accurate real time deformable objects. *Proceedings of SIGGRAPH'99 (Los Angeles, California)*, pages 65–72, August 1999. (référencé aux pages 23, 24, 25, 26, 77)

[Kas97] Michael Kass. An introduction to continuum dynamics for computer graphics. In *SIGGRAPH Course Notes*. ACM SIGGRAPH, 1997. (référencé en page 20)

[KWT87] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes : active contour models. In *International Journal Computer Vision*, volume 1(4), pages 321–332, 1987. (référencé en page 19)

[Las87] John Lasseter. Principles of traditional animation applied to 3d computer animation. *Proceedings of SIGGRAPH'87 (Anaheim, California)*, 21(4) :35–43, July 1987. (référencé en page 7)

[LC86] Annie Luciani and Claude Cadoz. Utilisation de modèles mécaniques et géométriques pour la synthèse et le contrôle d'images animées. In *Deuxième Colloque Image, CESTA*, Nice, April 1986. (référencé en page 16)

[LCN99] Jean-Christophe Lombardo, Marie-Paule Cani, and Fabrice Neyret. Real-time collision detection for virtual surgery. In *Proceedings of Computer Animation*, May 1999. (référencé en page 105)

[LHVD95] Annie Luciani, Arach Habibi, A. Vapillon, and Y. Duroc. A physical model of turbulent fluids. In *6th Eurographics Workshop on Animation and Simulation*, Maastricht, Netherlands, September 1995. (référencé en page 16)

[LJR+91] Annie Luciani, Stéphane Jimenez, Olivier Raoult, Claude Cadoz, and Jean-Loup Florens. A unified view of multitude behaviour, flexibility, plasticity, and fractures : balls, bubbles and agglomerates. In *IFIP WG 5.10 Working Conference*, Tokyo, Japan, April 1991. (référencé en page 16)

[LL59] L.D. Landeau and E.M. Lifshitz. *Theory of Elasticity*. Pergamon Press, 1959. (référencé en page 33)

[LP95] Jean-Christophe Lombardo and Claude Puech. Oriented particles : A tool for shape memory objects modelling. In *Graphics Interface'95*, Quebec, Canada, May 1995. (référencé en page 16)

[LPC95] Jean Louchet, Xavier Provot, and David Crochemore. Evolutionary identification of cloth animation models. In *Proceedings of the 6th Eurographics Workshop on Animation and Simulation*, pages 44–54. Springer-Verlag, September 1995. (référencé en page 16)

[LT94] Patrick Lascaux and Raymond Théodor. *Analyse numérique matricielle appliquée à l'art de l'ingénieur, méthodes directes*. Masson, seconde edition, 1994. (référencé aux pages 67, 79)

[LTW95] Y. Lee, Demetri Terzopoulos, and K. Waters. Realistic modeling for facial animation. *Proceedings of SIGGRAPH'95*, pages 55–62, 1995. (référencé en page 15)

[Luc85] Annie Luciani. Un outil informatique de création d'images animées. *Thèse de docteur ingénieur d'électronique*, Institut National Polytechnique de Grenoble, November 1985. (référencé aux pages 15, 30)

[MC96a] Philippe Meseure and C. Chaillou. Modélisation mécanique pour la simulation d'actes chirurgicaux. In *Actes du 4ème séminaire du groupe de travail Animation et Simulation, Strasbourg*, 1996. (référencé en page 26)

[MC96b] Philippe Meseure and C. Chaillou. Simulations de corps déformables par subdivision adaptative compatibles avec les découpes. In *Actes des 4èmes journées de l'AFIG, Dijon, France, 27-29 novembre 1996*, 1996. (référencé en page 26)

[McC89] S. F. McCormick. *Multilevel Adaptive Methods for Partial Differential Equations. Chapter 2 : The Finite Volume Method*. Vol. 6, SIAM, Philadelphia, 1989. (référencé en page 64)

[MDDB00] Mark Meyer, Gilles Debunne, Mathieu Desbrun, and Alan H. Barr. Interactive animation of cloth-like objects in virtual reality. In *Journal of Visualization and Computer Animation*, 2000. (référencé en page 121)

[Mes97] Philippe Meseure. *Modélisation de corps déformables pour la simulation d'actes chirurgicaux*. PhD thesis, Université des Sciences et Technologies de Lille, janvier 1997. (référencé aux pages 26, 120)

[Mil88] Gavin Miller. The motion dynamics of snakes and worms. *Proceedings of SIGGRAPH'88 (Atlanta, Georgia)*, 22(4) :169–177, August 1988. (référencé en page 15)

[Mil95] Roger B. Milne. An adaptive level-set method. *PHD Thesis*, University of California, Berkeley Lab, December 1995. (référencé en page 46)

[MJ96] R. MacCracken and K. Joy. Free-form deformations with lattices of arbitrary topology. *Proceedings of SIGGRAPH'96*, pages 181–188, 1996. (référencé en page 12)

[Mon92] J. J. Monaghan. Smoothed particle hydrodynamics. *Annual Review on Astronomy and Astrophysics*, 543(30) :543–574, 1992. (référencé aux pages 17, 49, 50)

[MP89] Gavin Miller and Andrew Pearce. Globular dynamics : A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3) :305–309, 89. This paper also appeared in SIGGRAPH'89 Course notes number 30. (référencé en page 16)

[MPT99] William A. McNeely, Kevin D. Puterbaugh, and James J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. In *SIGGRAPH 99 Conference Proceedings*, 1999. Los Angeles, CA. (référencé en page 107)

[MT92] Dimitri Metaxas and Demetri Terzopoulos. Dynamic deformation of solid primitives with constraints. *Computer Graphics*, 26(2) :309–312, July 1992. (référencé aux pages 14, 25)

[MWTT98] W. Maurel, Y. Wu, N. Magnenat Thalmann, and D. Thalmann. *Biomechanical Models for Soft Tissue Simulation*. Springer, 1998. (référencé en page 33)

[NG96] H. Ng and R. Grimsdale. Computer graphics techniques for modelling cloth. In *IEEE Computer Graphics and Applications*, volume 16(5), pages 28–41, 1996. (référencé en page 11)

[OH99] James O'Brien and Jessica Hodgins. Graphical models and animation of brittle fracture. In *Proceedings of SIGGRAPH'99*, pages 137–146, 1999. (référencé aux pages 24, 27, 29, 50, 77, 91, 125)

[OPT83] M. Ortiz, P.M. Pinsky, and R.L. Taylor. Unconditionally stable element-by-element algorithms for dynamic problems. *Computer Methods in Applied Mechanics and Engineering*, 36(2) :223–239, 1983. (référencé en page 121)

[PB81] S. Platt and Norman Badler. Animating facial expressions. *Proceedings of SIGGRAPH'81*, 15(3) :245–252, July 1981. (référencé en page 15)

[PB88] John Platt and Alan Barr. Constraint methods for flexible models. *Proceedings of SIGGRAPH'88 (Atlanta, Georgia)*, 22(4) :279–288, August 1988. (référencé en page 19)

[PLL98] Julien Péquignot, Jean-Christophe Lombardo, and Stéphane Lantéri. Optimisation géométrique locale de maillages tétraédriques. Technical report, ESSI3 - CSI - INRIA, 1998. (référencé en page 91)

[Pro95] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface*, pages 147–154, June 1995. (référencé en page 15)

[Pro97] Xavier Provot. *Animation Réaliste de Vêtements.* Thèse de doctorat, Synthim - INRIA, France, December 1997. (référencé en page 120)

[PSE+00] Joseph Popović, Steven M. Seitz, Michael Erdmann, Zoran Popović, and Andrew Witkin. Interactive manipulation of rigid body simulations. *Proceedings of SIGGRAPH'2000 (New-Orleans, Louisiana)*, pages 201–208, July 2000. (référencé en page 7)

[PTVF92] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. *Numerical Recipes in C, second edition*. Cambridge University Press, New York, USA, 1992. (référencé en page 120)

[PW89] Alex Pentland and John Williams. Good vibrations : Modal dynamics for graphics and animation. *Proceedings of SIGGRAPH'89 (Boston, MA)*, 23(3) :215–222, July 1989. (référencé aux pages 13, 14, 29)

[Ree83] W. T. Reeves. Particle systems—a technique for modeling a class of fuzzy objects. *Computer Graphics*, 17(3) :359–376, 1983. (référencé en page 16)

[Rey87] Craig W. Reynolds. Flocks, herds and schools : A distributed behavioral model. *Proceedings of SIGGRAPH'87*, 21(4) :25–34, juillet 1987. (référencé en page 16)

[Rey97] Hugh Reynolds. An alternative inter-particle force model for coupled system flexible body dynamics. In *8th Eurographics Workshop on Computer Animation and Simulation, Budapest, Hungria*, 1997. (référencé en page 16)

[Rob98a] B. Robertson. Antz-piration. *Computer Graphics World*, 21, 1998. (référencé en page 7)

[Rob98b] B. Robertson. Meet Geri : The new face of animation. *Computer Graphics World*, 21, 1998. (référencé en page 7)

[SBG96] Barry Smith, Petter Bjørstad, and William Gropp. *Domain Decomposition : Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge Univ Pr, May 1996. (référencé en page 33)

[Seg84] L. Segerlind. *Applied Finite Element Analysis*. John Wiley and Sons, New York, 1984. (référencé en page 20)

[SF93] Jos Stam and Eugene Fiume. Turbulent wind fields for gaseous phenomena. *Proceedings of SIGGRAPH'93*, 27 :369–376, August 1993. (référencé en page 17)

[SF95] Jos Stam and Eugene Fiume. Depicting fire and other gaseous phenomena using diffusion processes. *Proceedings of SIGGRAPH'95 (Los Angeles)*, pages 129–136, August 1995. (référencé en page 17)

[SIM] SIMULOG. Ghs3d. (référencé en page 92)

[SP86] Thomas W. Sedeberg and Scott R. Parry. Free-form deformations of solid geometric models. *Computer Graphics*, 20(4) :151–160, 1986. (référencé en page 12)

[ST92] Richard Szeliski and David Tonnesen. Surface modeling with oriented particle systems. *Proceedings of SIGGRAPH'92 (Chicago)*, 26(2) :185–194, July 1992. (référencé en page 16)

[ST95] Jianhua Shen and Daniel Thalmann. Interactive shape design using metaballs and splines. In *Implicit Surfaces'95—the First Eurographics Workshop on Implicit Surfaces*, pages 187–195, Grenoble, France, April 1995. (référencé en page 25)

[Sta99] Jos Stam. Stable fluids. *Proceedings of SIGGRAPH'99*, pages 121–128, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California. (référencé en page 115)

[Tec] SensAble Technologies. Système à retour d'effort "Phantom". http ://www.sensable.com. (référencé aux pages 26, 107)

[TF88] Demetri Terzopoulos and Kurt Fleischer. Modeling inelasctic deformation : Viscoelasticity, plasticity, fracture. *Proceedings of SIGGRAPH'88 (Atlanta, Georgia)*, 22 :269–278, August 1988. (référencé aux pages 19, 25)

[TG70] S.P. Timoshenko and J.N. Goodier. *Theory of Elasticity*. McGraw-Hill, 1970. (référencé en page 33)

[Ton91] David Tonnesen. Modeling liquids and solids using thermal particles. In *Graphics Interface'91*, pages 255–262, Calgary, AL, June 1991. (référencé en page 16)

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *Proceedings of SIGGRAPH'87 (Anaheim, California)*, 21 :205–214, July 1987. (référencé aux pages 19, 20, 25, 121)

[TPF89] Demetri Terzopoulos, John Platt, and Kurt Fleisher. Heating and melting deformable models (from goop to glop). In *Graphics Interface'89*, pages 219–226, London, Ontario, June 1989. (référencé aux pages 16, 25)

[TT93] Russel Turner and Daniel Thalmann. The elastic surface layer model for animated character construction. In Springer Verlag, editor, *Computer Graphics International*, 1993. (référencé en page 25)

[Tur95] Russel Turner. Leman : A system for constructing and animating layered elastic characters. In *Computer Graphics - Developments in Virtual Environments*, pages 185–203, Academic Press, San Diego, CA, June 1995. (référencé en page 25)

[TW88] Demetri Terzopoulos and Andrew Witkin. Physically based model with rigid and deformable component. *IEEE Computer Graphics and Applications*, pages 41–51, décembre 1988. (référencé aux pages 19, 25)

[TW90] Demetri Terzopoulos and K. Waters. Physically based facial modeling analysis and animation. In *Journal of Visualization and Computer Animation*, pages 73–80, 1990. (référencé en page 15)

[Val99] Bernard Valton. *Gestion de la complexité de scènes animées et interactives : contributions à la conception et à la représentation*. PhD thesis, Université de Rennes 1, 1999. (référencé en page 28)

[VB87] Brian Von Herzen and Alan H. Barr. Accurate triangulations of deformed, intersecting surfaces. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 103–110, July 1987. (référencé en page 51)

[WFB87] Andrew Witkin, Kurt Fleisher, and Alan H. Barr. Energy constraint on parametrized models. *Computer Graphics*, 21(4) :225–232, July 1987. (référencé en page 19)

[WH94] Andrew Witkin and Paul Heckbert. Using particles to sample and control implicit surfaces. *Proceedings of SIGGRAPH'94*, pages 269–278, July 1994. (référencé en page 13)

[Wit99] Patrick Witting. Computational fluid dynamics in a traditional animation environment. *Proceedings of SIGGRAPH 99*, pages 129–136, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California. (référencé en page 114)

[WMW86] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4) :227–234, August 1986. (référencé en page 13)

[WW90] Andrew Witkin and William Welch. Fast animation and control for non-rigid structures. *Proceedings of SIGGRAPH'90 (Dallas, Texas)*, 24(4) :243–252, August 1990. (référencé aux pages 14, 25)

[YOH00] Gary D. Yngve, James F. O'Brien, and Jessica K. Hodgins. Animating explosions. *Proceedings of SIGGRAPH 2000*, pages 29–36, July 2000. ISBN 1-58113-208-5. (référencé en page 114)

[Zhu00] Yan Zhuang. *Real-time Simulation of Physically-Realistic Global Deformations.* PhD thesis, UC Berkeley, Department of Electrical Engineering and Computer Science, Fall 2000. http ://www.cs.berkeley.edu/ yzhuang/publications.html. (référencé en page 29)

[ZT91] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method*. McGraw-Hill, 1991. (référencé en page 20)