Modèle déformable 1D pour la simulation physique temps réel

(version modifiée du 28/01/05)

THÈSE

présentée et soutenue publiquement le 23 novembre 2004

pour l'obtention du

Doctorat de l'Université des Sciences et Technologies de Lille

(spécialité informatique)

par

Julien Lenoir

Composition du jury

Président :	Sophie Tison
Rapporteurs :	Marie-Paule Cani Hervé Delingette Dinesh K. Pai
Examinateur :	Yannick Rémion
Directeur :	Christophe Chaillou
Encadrants :	Philippe Meseure Laurent Grisoni

UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE
Laboratoire d'Informatique Fondamentale de Lille — UPRESA 8022
U.F.R. d'I.E.E.A. – Bât. M3 – 59655 VILLENEUVE D'ASCQ CEDEX
Tél. : +33 (0)3 28 77 85 41 – Télécopie : +33 (0)3 28 77 85 37 – email : `direction@lifl.fr`

Remerciements

Je tiens tout d'abord à remercier Christophe Chaillou pour m'avoir donné la possibilité de réaliser cette thèse et pour m'avoir accueilli au sein de l'équipe GRAPHIX.

Je remercie tout particulièrement Philippe Meseure pour m'avoir encadré dès le DEA et m'avoir donné l'envie de poursuivre en thèse. Son enthousiasme communicatif, sa disponibilité et sa grande maîtrise du sujet ont eu raison des doutes sur une éventuelle poursuite en thèse.

Je remercie également Laurent Grisoni pour m'avoir co-encadré et soutenu pendant ces trois années. Son expertise scientifique et ses qualités humaines ont été des atouts précieux dans l'évolution de cette thèse.

Merci à Sophie Tison qui m'a fait l'honneur de présider mon jury.

Merci à Marie-Paule Cani, Hervé Delingette et Dinesh Pai d'avoir accepté la difficile et lourde tâche de rapporter ma thèse. Je les remercie pour leurs commentaires avisés et leur participation à l'établissement du document final.

Merci à Yannick Rémion d'avoir accepté d'être examinateur de ma thèse, mais aussi pour avoir partagé son expertise et ses connaissances depuis quatre ans.

Merci à toutes les personnes rencontrées au cours de mes enseignements.

Merci à Stéphane Cotin qui me donne la possibilité de poursuivre mes recherches en Post-Doc.

Un grand merci à tous les membres et ex-membres de l'équipe Graphix pour les nombreux moments de fou rire et de détente mais aussi d'aide et de soutien : Fred 1 et 2, Lol, Hollowman, Steph, Fab, JP, Sam, SB, Patricia, Quicksilver, Bilboox, Jay, Damien (mais aussi Iovka pour son soutien moral), Nioc-Nioc, Sylvain 1 et 2, Nico, Géry, Bibi, Wizzer, Lilith (qui m'a initié aux joies de la 3D en 1ère année de DEUG et que j'ai retrouvé avec plaisir un peu plus tard), Lulu (à prononcer Loulou!! pour son soutien et sa présence réconfortante), Zoue (pour sa joie de vivre, son optimisme et sa confiance).

Merci à ma famille qui m'a soutenue pendant ses huit longues années d'études et plus particulièrement pendant ses trois années de thèse.

Merci à Annick qui est arrivée au bon moment dans ma vie et qui m'a montré que d'un malheur pouvait naître un bonheur.

Enfin, un merci tout particulier à Laure...

Résumé

Cette thèse s'inscrit dans un cadre de simulation temps réel basée sur la physique. Le but premier de ce travail est de proposer un modèle déformable 1D. Les applications d'un tel modèle sont nombreuses en réalité virtuelle ou en animation pour la simulation d'objets déformables longilignes, tels que les cordes, ficelles ou lacets.

Nous proposons un modèle déformable 1D temps réel basé sur une géométrie de type spline et animé par les équations physiques de Lagrange. Nous nous sommes appuyés pour cela sur les travaux de Yannick Rémion et de son équipe au LERI. Ce modèle se révèle particulièrement adapté à la simulation chirurgicale pour la représentation de fil de suture ou d'organes (intestin grêle, trompes de Fallope,...).

Pour certaines applications, il peut être intéressant de demander au modèle déformable de vérifier certaines conditions exprimées sous formes d'équations de contraintes. La prise en compte de ces contraintes s'effectue, dans le système dynamique, à l'aide de la méthode des multiplicateurs de Lagrange. Dans ce contexte de contraintes pour la simulation dynamique, l'une des contributions majeures de cette thèse est la proposition d'une nouvelle classe de contraintes, appelée *contraintes glissantes*. Elles permettent, par exemple, d'imposer à un fil de passer par un point de l'espace sans imposer de valeur paramétrique correspondante. Ce type de contrainte est particulièrement utile pour la simulation de suture dans un contexte chirurgicale, mais répond aussi à des besoins d'animations spécifiques (lacet de chaussure, nœuds coulant, ...).

Certaines applications, comme la suture d'organe, mettent en jeu plusieurs modèles dynamiques liés ensemble. Pour ce type de simulation, nous proposons une architecture logicielle permettant de simuler des articulations d'objets quelconques (rigides ou déformables) quel que soit le formalisme physique employé pour chacun d'eux. Cette proposition logicielle trouve diverses applications notamment en simulation chirurgicale mais permet aussi de simuler dynamiquement toute articulation d'objets hétérogènes.

Certaines manipulations requièrent une souplesse du modèle à des endroits précis, sachant que ces zones peuvent se déplacer lors d'une simulation, par exemple pendant le serrage d'un nœud. Pour cela, nous proposons une multi-résolution géométrique et mécanique sur notre modèle qui vise à adapter localement sa résolution afin qu'il puisse s'adapter aux interactions tout en offrant des bonnes performances générales en calcul. On concentre alors le plus gros du temps de calcul sur les zones d'intérêt et on limite ce temps dans les autres zones du modèle. Un critère d'adaptation de la résolution en fonction de la courbure est proposé. Cette technique est particulièrement adaptée à la simulation de nœuds en permettant à la spline d'augmenter le nombre de degrés de liberté et ainsi en lui fournissant une grande souplesse de définition géométrique dans la zone de serrage.

Mots-clés: Simulation physique temps réel, Dynamique, Modèle déformable 1D, Contraintes, Multi-résolution, Découpe.

A ma famille
A mes amis

Table des matières

Table des figures

Notation

Ce document s'appuie sur les notations suivantes :

Notations générales :

t désigne le temps.
s une abscisse paramétrique.
P un point.
T une tangente.
C une courbure ou un amortissement.
V une vitesse.
A une accélération.
F une force.
M ou m une masse.
$A \oplus B = C$ une somme directe. (i.e. $\forall x \in C, \exists! x_A \in A, \exists! x_B \in B$ telque $x = x_A + x_B$)

Notations des dérivées :

$\dot{q}(t)$ la dérivée première de q par rapport au temps t : $\frac{dq}{dt}$. Désigne la vitesse de q.
$\ddot{q}(t)$ la dérivée seconde de q par rapport au temps t : $\frac{d^2q}{dt^2}$. Désigne l'accélération de q.
La notation $\dot{x}$ désigne toujours une dérivée totale temporelle, par exemple : $\dot{q}(t,s) = \frac{dq}{dt}(t,s)$.

Notation scalaire :

t un scalaire.

Notations vectorielles :

$\mathbf{v}$ un vecteur (noté $\vec{v}$ dans certain cas, le lecteur en est alors averti).
$\mathbf{AB}$ est le vecteur reliant le point $\mathbf{A}$ au point $\mathbf{B}$: $\mathbf{AB} = \mathbf{B} - \mathbf{A}$.
$\mathbf{A_t}$ un vecteur indicé.
$\mathbf{A_{(n)}}$ un vecteur de dimension n.
$\mathbf{P}(a)$ un point dépendant d'un paramètre scalaire a.
$\mathbf{g}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t)$ une fonction vectorielle de trois paramètres : $\mathbf{q}$ et $\dot{\mathbf{q}}$ deux vecteurs et t un scalaire.
$\mathbf{u}.\mathbf{v}$ le produit scalaire des vecteurs $\mathbf{u}$ et $\mathbf{v}$.
$\mathbf{u} \wedge \mathbf{v}$ le produit vectoriel des vecteurs $\mathbf{u}$ et $\mathbf{v}$.
$|\mathbf{u}|$ représente la norme euclidienne du vecteur $\mathbf{u}$. En dimension 2, on a donc $|\mathbf{u}| = \sqrt{u_x^2 + u_y^2}$.

Notations matricielles :

M une matrice.
Id la matrice identitée.
$M_{(n \times m)}$ une matrice de dimension $n \times m$ (n lignes et m colonnes).
M^{-1} la matrice inverse de M.

M^T la matrice transposé de M.
$M.N$ dénote le produit de la matrice M par la matrice N.
$M.\mathbf{n}$ dénote le produit de la matrice M par le vecteur $\mathbf{n}$.

Notations tensorielles :

$\mathbf{u} \otimes \mathbf{v}$ est le produit tensoriel des vecteurs $\mathbf{u}$ et $\mathbf{v}$.

Introduction

L'animation est un domaine de l'informatique graphique en pleine effervescence. Ceci est dû à l'utilisation de plus en plus fréquente d'images virtuelles pour les effets spéciaux de films ou de publicité. On trouve également des images de synthèse de plus en plus réalistes dans les jeux vidéo et les simulateurs (aviation, conduite automobile ou chirurgie).

On peut scinder ces applications en deux groupes, celui constitué des jeux vidéo et des simulateurs qui ont pour dénominateur commun l'impératif du temps réel ou interactif. Le second groupe contient les applications en temps différé, utilisées par le septième art et la publicité (film d'animation ou effets spéciaux). Dans ce second groupe, les outils de CAO (Conception Assistée par Ordinateur) sont largement utilisés pour définir l'animation souhaitée, avant de passer à la phase finale de calcul des images de synthèse. Les *designers* en charge de l'animation utilisent des outils dédiés permettant de définir de manière simple, rapide et intuitive la séquence animée. Différentes techniques sont donc proposées pour animer des objets, comme le key-framing, la cinématique inverse, l'animation comportementale...

L'objectif de l'animation par ordinateur est de limiter l'investissement du concepteur pour lui permettre de se concentrer sur le scénario ou simplement de réduire le temps de production de l'animation. Dans cet objectif d'abstraction des paramètres de l'animation, le fait de les définir au début de l'animation et de laisser la méthode calculer la séquence animée est un avantage. Dans ce contexte, l'utilisation des lois physiques est une approche naturelle pour animer des objets issus de la réalité. Pour cela, des modèles sont proposés et leur mise en application est réalisée par *simulation*.

Ce type d'animation basée sur des lois physiques est employé dans certains jeux vidéo[1] mais aussi dans les simulateurs. Dans ce contexte, l'un des domaines porteurs est la simulation chirurgicale qui implique la restitution du comportement d'objets mous. En effet, la plupart des opérations chirurgicales simulées ont trait à un organe en particulier, or ces objets sont des tissus organiques plus ou moins déformables. D'autres opérations chirurgicales mettent en jeu des interactions entre divers modèles, un exemple est donné par la suture d'un organe, l'aiguille est liée à un fil de chirurgie qui est lui-même en interaction avec l'organe suturé. Ce type d'intervention requiert une attention particulère au niveau des interactions entre les modèles. Ce domaine fait, à l'heure actuelle, l'objet de nombreuses recherches et quelques sociétés comme [KIS, SIM, Sur] proposent des simulateurs chirurgicaux pour certains protocoles opératoires.

Dans la grande majorité des opérations chirurgicales, le praticien est amené, à un moment ou à un autre, à manipuler un fil chirurgical. Cette manipulation prend place généralement lors de la suture d'un organe ou d'une plaie, et se termine toujours par la création d'un nœud avec le fil. Pour simuler cela, il faut disposer d'un modèle qui offre suffisament de souplesse pour être manipulé de façon intuitive. De plus, il doit intégrer des fonctionalités automatiques d'adaptation en fonctions des manipulations. Il doit être capable, par exemple, d'adapter sa résolution pour permettre la réalisation et le serrage d'un nœud. Afin d'offrir des fonctionalités supplémentaire à l'utilisateur, le modèle doit également être pourvu d'un jeu de contraintes utiles.

Nous proposons un modèle déformable 1D temps réel capable de simuler le comportement d'une large gamme d'objets filiformes. Notre modèle est capable de gérer ses déformations en flexion au travers d'une énergie de flexion continue proposée. Ce modèle est muni d'un jeu de contraintes avec notamment la proposition d'un nouveau type de contraintes dites *contraintes glissantes* très utile en simulation et en animation de manière générale. Une proposition d'architecture d'objets articulés est également faite pour permettre de définir des contraintes entre des objets quelconques. Enfin, l'une des contributions importantes de cette thèse est la multi-résolution du modèle mécanique. De ce fait, le modèle 1D est capable d'adapter automatiquement sa résolution en fonction des interactions et offre, en plus, la possibilité de le découper.

Nous commençons par détailler les divers points qui ont trait à notre sujet en développant, pour chacun d'eux, quelques propositions relevées dans la littérature. Ces points vont de la simulation physique à la gestion de contraintes, en passant par le cas particulier de la simulation d'objets 1D. Ce tour d'horizon des travaux existants dans les domaines relatifs à notre sujet nous permet ensuite de faire des choix éclairés lors de la mise en place de notre modèle.

[1] http ://www.havok.com

Ensuite, nous détaillons notre modèle d'objet 1D déformable en définissant sa géométrie à base de spline, animée par le formalisme physique de Lagrange. De ce formalisme découle les équations qui permettent de mettre en place la simulation. Une résolution efficace de ces équations permet à notre modèle d'être simulé en temps réel. Quant à l'aspect déformable de l'objet, il se traduit par l'introduction d'énergies de déformation en élongation et en flexion. Aussi, pour simuler une large gamme de comportements, plusieurs possibilités sont offertes au travers d'énergies discrètes et continues. Afin de représenter des objets de nature différente, nous proposons d'habiller notre modèle physique à l'aide d'une multitude de techniques, allant d'une simple ligne brisée à une surface à convolution en passant par un cylindre généralisé ou encore une surface implicite à squelette ponctuel. L'utilisation de ces techniques d'habillage nous permet de mettre en valeur notre modèle dans diverses applications, comme la simulation de fil, de corde ou d'organes longilignes (intestin grêle) dans un contexte chirurgical. Cependant, certaines applications nécessitent des modèles déformables surfaciques ou volumiques, aussi nous détaillons les possibilités d'extensions de notre proposition aux dimensions supérieures. Cependant, il arrive assez fréquemment qu'un objet soit composé de plusieurs parties contraintes les unes aux autres, le tout formant un objet complexe dont la simulation requiert une attention particulière.

Dans ce contexte d'objets dits articulés, nous détaillons notre proposition d'architecture logicielle capable d'intégrer plusieurs modèles de natures différentes et de définir des contraintes entre ces modèles. Cette proposition trouve des applications, que nous décrivons, comme la simulation d'une balançoire, ou en simulation chirurgical au travers d'une simulation de pré-découpe d'une trompe de Fallope. Nous détaillons également une application avec la mise place d'une méthode d'interaction via un périphérique spécifique. Cette proposition d'articulation d'objets montre l'intérêt de définir des contraintes sur des modèles, mais notre modèle déformable 1D peut lui aussi être muni de contraintes. Nous détaillons donc un jeu de contraintes, pris en compte par la méthode des multiplicateurs de Lagrange, permettant par exemple de fixer un point du modèle, une tangente, une courbure... De plus, nous définissons une nouvelle classe de contraintes dites *contraintes glissantes* en détaillant la théorie nécessaire à leur introduction dans notre modèle. Ceci permet de créer des contraintes qui se déplacent le long du modèle 1D, apportant une souplesse supplémentaire. Ce type de contraintes trouve des applications dans de nombreux domaines comme la suture d'organe, la modélisation variationnelle (lacet de chaussure, nœud coulant,...),... Ces propositions offrent de multiples possibilités au modèle, cependant il est nécessaire d'adapter les déformations du modèle pour des interventions extrêmes comme la création d'un nœud.

Dans ce souci d'adaptation, nous proposons de démultiplier le nombre de degrés de liberté aux endroits où cela est nécessaire. Afin de mieux percevoir les différentes techniques possibles, nous exposons quelques travaux de la littérature qui ont trait à la multi-résolution. Cela nous permet de placer notre proposition de modèle mécanique multi-résolution par rapport aux travaux existants. Pour cela, nous présentons les techniques que nous avons choisi de mettre en place pour ajouter ou supprimer des degrés de liberté. Sachant que cette proposition de multi-résolution est tout à fait compatible avec les propositions faites dans le chapitre précédent sur les contraintes. La mise en place du modèle multi-résolution s'effectue par le biais de critères qui permettent d'évaluer les zones du modèle susceptibles d'avoir besoin d'une meilleure résolution. Une fois le changement de résolution effectué, certaines parties du modèle ont subi des modifications qui requièrent une étude dédiée à la stabilité de notre modèle multi-résolution. Enfin, des tests permettent de montrer l'efficacité de la proposition. Et, une conséquence originale de la multi-résolution est la découpe de notre modèle 1D.

Enfin, ce document se termine par un récapitulatif des diverses propositions faites pour établir le modèle proposé, avant de conclure et d'exposer quelques perspectives à venir.

Chapitre 1

Généralités sur la simulation physique, les modèles 1D et les contraintes

Sommaire

Notre but premier est de proposer un modèle déformable d'objet 1D simulé à l'aide de lois physiques. Aussi, il est important d'avoir une vue assez large des divers travaux qui ont pu être développés sur ce sujet. De ce fait, nous commençons par présenter quelques travaux qui ont traits à la simulation physique en générale. Une fois le contexte des simulations physiques bien définie, nous discutons de certains travaux plus proche de notre sujet, à savoir des modèles 1D.

Ensuite, le second but de notre modèle 1D est d'offrir suffisament de souplesse pour être utilisé dans une large gamme d'applications. Pour cela, notre modèle doit proposer un ensemble de contraintes étendues. Afin de mieux percevoir les différents techniques possible pour intégrer des contraintes au sein d'un modèle mécanique, nous proposons de détailler ici certaines d'entre elles.

1.1 Simulation physique

L'algorithme utilisé pour la simulation physique d'un objet dépend grandement de la représentation mathématique de cet objet. Tous les objets rencontrés en réalité sont plus ou moins déformables, mais lorsque les propriétés physiques du matériau contraignent les déformations internes à des amplitudes extrêmement faibles, l'objet est qualifié de *rigide*. Ce type d'objet permet une gestion différente de la dynamique, ainsi en simulation physique, on traite séparément ces objets rigides en tant qu'objets indéformables.

La plupart des objets déformables possèdent une configuration de référence permettant de mesurer les déformations courantes. Pour cela, l'objet est muni d'une structure interne permettant de faire cette comparaison et d'en déduire les déformations. Cependant, certains travaux traitent d'objets déformables ne possédant aucune structure interne particulière. Ainsi les déformations ne peuvent plus être déduites d'une comparaison avec une configuration de référence, mais à l'aide d'informations locales propres à chaque élément infinitésimal constituant l'objet. Ce type de simulation permet donc d'animer des objets de type fluide comme les gaz, la fumée, les liquides... Pour simuler ce type d'objet [Hab97], il est possible de déterminer la trajectoire des particules en adoptant une approche Lagrangienne[MCG03, FOA03, DC96] ou encore de déterminer le flux continu du domaine par une approche Eulérienne[GBO04, FF01, FM97b, FM97a, Sta03, FSJ01, Sta99, Sta00, MST+04].

Dans la suite, on considère toujours des objets possédant une structure interne, les objets déformables démunis d'une telle structure sont donc exclus. On distingue par la suite les objets rigides, ou objets à géométrie fixe, des objets déformables. Pour cela, un premier paragraphe traite des objets rigides puis un second paragraphe détaille divers travaux menés sur les objets déformables et enfin, un troisième paragraphe expose quelques travaux propres au cas particulier des modèles 1D.

1.1.1 Objets rigides

Les objets rigides sont pourvus de contraintes internes ne leur permettant aucune déformation. Ainsi, la géométrie de l'objet est fixée au cours du temps. Ce type d'objet se simule extrêmement bien à l'aide du formalisme de Newton-Euler.Dans ce formalisme, un objet de masse m est défini par les coordonnées de son centre de masse $\mathbf{G}$ (barycentre des points $\mathbf{P_i}$ du solide pondérés par leur masse m_i dans le cas discret ou leur densité volumique de masse ρ dans le cas continu)

$$\mathbf{OG} = \begin{cases} \frac{1}{m}\sum_i m_i \mathbf{OP_i} & \textit{cas discret} \\ \frac{1}{m}\int_V \rho \mathbf{OP} dv & \textit{cas continu} \end{cases}$$

ainsi qu'un vecteur orientation $\mathbf{\Omega}$ (généralement noté sous forme d'un quaternion $\mathbf{q}$). La dynamique fait intervenir les dérivées temporelles de ces deux composantes, ce qui apporte :

– à l'ordre un, le vecteur vitesse $\dot{\mathbf{G}}$ du centre de masse et le vecteur vitesse angulaire $\boldsymbol{\omega}$:

$$\begin{cases} \dot{\mathbf{OG}} &= \frac{d\mathbf{OG}}{dt} \\ \boldsymbol{\omega} &= \dot{\mathbf{\Omega}} = \frac{d\mathbf{\Omega}}{dt} \text{ (ou } \dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\omega}\mathbf{q} \text{)} \end{cases}$$

la relation $\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\omega}\mathbf{q}$ permet de faire le lien entre le quaternion, qui oriente instantanément l'objet, et le vecteur vitesse angulaire.

– à l'ordre deux, le vecteur accélération $\ddot{\mathbf{G}}$ du centre de masse et le vecteur accélération angulaire $\dot{\boldsymbol{\omega}}$:

$$\begin{cases} \ddot{\mathbf{OG}} &= \dfrac{d^2\mathbf{OG}}{dt^2} \\ \dot{\boldsymbol{\omega}} &= \ddot{\boldsymbol{\Omega}} = \dfrac{d^2\boldsymbol{\Omega}}{dt^2} = \dfrac{d\boldsymbol{\omega}}{dt} \end{cases}$$

A partir de ces définitions, la dynamique du corps rigide est régie par l'équation :

$$\begin{cases} \displaystyle\sum_{i=1}^{n} \mathbf{F_i} = m\ddot{\mathbf{G}} \\ \displaystyle\sum_{i=1}^{n} \mathbf{M_i} = \sum_{i=1}^{n} \mathbf{GA_i} \wedge \mathbf{F_i} = \frac{I\boldsymbol{\omega}}{dt} \end{cases} \tag{1.1}$$

avec $\mathbf{F_i}$ les n forces qui s'appliquent sur le solide, $\mathbf{A_i}$ le point d'application (sur le solide) de la force $\mathbf{F_i}$, $\mathbf{M_i}$ représente le moment dû à la force $\mathbf{F_i}$ appliquée en $\mathbf{A_i}$ et I la matrice d'inertie de l'objet.

La matrice d'inertie I traduit la distribution de masse dans le volume de l'objet pour les trois directions de l'espace $\mathbb{R}^3$. Pour un objet continu, décrivant un volume V avec une densité de masse définie par la fonction spatiale ρ, la matrice d'inertie associée est :

$$I = \begin{pmatrix} \int_V \rho(y^2+z^2)\,dv & -\int_V \rho xy\,dv & -\int_V \rho xz\,dv \\ -\int_V \rho xy\,dv & \int_V \rho(x^2+z^2)\,dv & -\int_V \rho yz\,dv \\ -\int_V \rho xz\,dv & -\int_V \rho yz\,dv & \int_V \rho(x^2+y^2)\,dv \end{pmatrix}$$

pour un modèle discret, la matrice se définit par des sommes discrètes :

$$I = \begin{pmatrix} \sum_i m_i(y_i^2+z_i^2) & -\sum_i m_i x_i y_i & -\sum_i m_i x_i z_i \\ -\sum_i m_i x_i y_i & \sum_i m_i(x_i^2+z_i^2) & -\sum_i m_i y_i z_i \\ -\sum_i m_i x_i z_i & -\sum_i m_i y_i z_i & \sum_i m_i(x_i^2+y_i^2) \end{pmatrix}$$

Ainsi, lorsqu'une force $\mathbf{F}$ s'applique sur l'objet, elle induit un déplacement de l'objet au travers de son action sur la dynamique du centre de masse mais aussi un changement d'orientation suivant son point d'application et l'inertie de l'objet.

Partant de ce formalisme, le réalisme de la simulation peut être augmenté au travers d'ajouts de forces particulières. [Hah88] propose d'ajouter des impulsions au niveau des impacts lors de collision entre objets, ainsi qu'une force de friction dérivant de la loi de Coulomb. Ainsi, les impacts sont gérés à des temps discrets de la simulation et le contact continu de deux objets est approximé par une série de contacts instantanés.

[Bar94] propose un modèle de contact basé sur les frictions statiques et dynamiques avec de multiples points de contacts. Cette technique consiste à exprimer pour chaque point de contact $\mathbf{p_i}$, l'accélération relative $\mathbf{a_i}$ des deux objets. Ainsi, une accélération négative traduit une interpénétration des deux objets alors qu'une accélération nulle traduit un contact. [Bar94] propose une technique de résolution du problème multi-contraint.

1.1.2 Objets déformables

La simulation d'objets déformables fait l'objet de nombreux travaux comme le font remarquer [GM97, Mes02] ou, plus spécifiquement dans le contexte chirurgical, [MT96].

Au travers de l'étude des travaux existants dans le domaine, il apparaît deux manières distinctes de représenter un objet déformable pour en étudier la dynamique :

L'approche Eulérienne :

Dans l'approche Eulérienne, la dynamique est observée en chaque point de l'espace. Ainsi, l'étude porte sur les flux à l'intérieur d'un domaine. Il n'est alors pas possible de suivre le mouvement

d'une particule donnée de l'objet. En effet, cette particule se déplace d'une zone de l'espace à une autre mais ce sont ces zones de l'espace qui informent du déplacement local sans pour autant savoir quelle est la particule de l'objet présent à cet instant. Cette approche ne permet donc pas de suivre l'évolution des points caractéristiques d'un objet, elle n'est donc pas adaptée à l'étude de la dynamique des objets structurés. L'approche Eulérienne convient aux objets non structurés comme les fluides.

L'approche Lagrangienne :

L'approche Lagrangienne consiste à étudier la dynamique d'une particule liée à l'objet. Cette approche est adaptée à tout objet structuré puisqu'elle permet de suivre les déplacements de chaque point caractéristique de l'objet (de sa structure).

Ne considérant pas le cas des fluides dans la suite de l'exposé, l'approche considérée est toujours Lagrangienne.

L'animation dynamique d'un objet déformable débute par le choix d'un formalisme physique qui apporte des propriétés particulières au modèle mécanique. Les modèles physiques sont tous simulés via un processeur réalisant des calculs discrets, ce qui implique que les modèles simulés passent tous par une étape de discrétisation. Cependant, suivant le moment où intervient cette étape, le modèle peut être qualifié de discret ou de continu.

Si la modélisation physique est réalisée par un ensemble fini de points, alors la discrétisation s'opère dès le début de la conception du modèle, on peut ainsi parler de modèle discret. Mais, si le modèle est conçu à l'aide d'une sommation infinitésimale, d'équations de continuité ou de toute autre technique permettant de conserver la continuité intrinséque du modèle, il peut être alors qualifié de continu.

Cependant, la discrétisation d'un modèle continu peut s'effectuer de deux manières différentes suivant le formalisme choisi, l'une est spatiale (l'objet est décomposé en sous-éléments qui ont une réalité géométrique et physique) l'autre est *conceptuelle* (l'objet est défini à l'aide de degrés de liberté issus d'une description mathématique du modèle).

Cette section commence donc par décrire les outils principaux qui ont trait aux modèles discrets, puis s'attarde sur les modèles continus à discrétisation spatiale avant de terminer sur des travaux propres aux modèles continus à degrés de liberté quelconques.

1.1.2.1 Modèles discrets (Masses-Ressorts)

Un des modèles les plus étudiés en matière de simulation physique d'objets déformables est le modèle masses-ressorts[TGG00, Pro95, LPC95, BC00, NT98]. Ce modèle a l'avantage d'être extrêmement simple à comprendre et à réaliser. Le principe est de disposer des masses ponctuelles sur l'objet à simuler en les reliant à l'aide de ressorts plus ou moins élaborés. Ce modèle propose simplement de définir un objet de masse m par le biais d'un ensemble fini de points spatiaux ponctuels ($\mathbf{P} = \{\mathbf{P_i}(x, y, z) | i \in \{1..n\}\}$) chacun d'entre eux étant muni d'une masse m_i et de définir, sur cet ensemble de points, des liaisons à l'aide de ressorts plus ou moins élaborés. La dynamique du modèle est alors assurée par le formalisme physique de Newton connu comme le principe fondamental de la dynamique (PFD) :

$$\sum_j \mathbf{F_j} = m\mathbf{a} \tag{1.2}$$

où $\mathbf{a}$ est l'accélération du point considéré, m sa masse et $\mathbf{F_j}$ les forces appliquées à ce point à l'instant t. L'application du *PFD* au masses-ressorts nous donne :

$$\forall i, \sum_{j=1}^{n_i} \mathbf{F_j^i} = m_i \ddot{\mathbf{P}}_\mathbf{i} \tag{1.3}$$

où $\ddot{\mathbf{P}}_\mathbf{i} = \frac{d^2\mathbf{P_i}}{dt^2}$ est l'accélération de la particule $\mathbf{P_i}$ et $\mathbf{F_j^i}$ représente les n_i forces appliquées au point $\mathbf{P_i}$. Ainsi, dans le bilan de force de chaque particule, vient s'ajouter, aux forces usuelles (gravité, frottements visqueux...), les forces dues aux ressorts incidents à la particule. Afin de vérifier le principe physique de conservation de l'énergie d'un système, chaque ressort doit vérifier la propriété suivante : la somme des

forces qu'il induit est nulle, de même pour les moments (ce qui se traduit par le principe d'action-réaction). Lorsqu'il n'y a que deux acteurs (ressort binaire), cela revient au principe physique d'action/réaction (i.e. les deux forces sont colinéaires, de même intensité mais de sens opposé).

Les ressorts permettent d'insuffler au modèle des énergies de déformation qui ont pour tâche de définir le comportement de l'objet. Ainsi, le choix du type de ressort est des plus importants dans l'élaboration d'un modèle masses-ressorts. Pour cela, voici une liste non exhaustive de quelques types de ressorts utiles :

Ressort d'élongation :

Ce type de ressort se définit entre deux points **A** et **B** à l'aide d'une longueur au repos l_0 et d'un coefficient de raideur k relatif à l'élasticité du matériau simulé. La force exercée par **B** sur **A** est :

$$\mathbf{F_{B->A}} = k\left(|\mathbf{AB}| - l_0\right).\frac{\mathbf{AB}}{|\mathbf{AB}|}$$

Par le principe physique d'action/réaction, la force exercée par **A** sur **B** est donnée par

$$\mathbf{F_{A->B}} = -\mathbf{F_{B->A}}$$

Ressort d'élongation amorti :

Un ressort d'élongation amorti avec les mêmes propriétés que le ressort possède en plus un coefficient d'amortissement a, ce qui permet d'amortir la force en fonction de la vitesse des particules :

$$\mathbf{F_{B->A}} = \left(k(|\mathbf{AB}| - l_0) + a\frac{\dot{\mathbf{AB}}.\mathbf{AB}}{|\mathbf{AB}|}\right).\frac{\mathbf{AB}}{|\mathbf{AB}|}$$

où $\dot{\mathbf{AB}}$ représente la vitesse relative des deux points **A** et **B** : $\dot{\mathbf{AB}} = \dot{\mathbf{B}} - \dot{\mathbf{A}}$.

La force exercée par **A** sur **B** est toujours définie par la relation $\mathbf{F_{A->B}} = -\mathbf{F_{B->A}}$.

Ressorts angulaires :

Ce type de ressort est basé sur trois points ponctuels et permet de définir une force relative à l'angle formé par les deux vecteurs adjacents. Ainsi, pour trois points ordonnés **A**, **B** et **C**, un ressort angulaire de raideur k et d'angle au repos α_0 produit les forces suivantes (formule linéarisée [BC00]) :

$$\begin{aligned}
\mathbf{F_A} &= -k\left[\frac{\mathbf{BA}.\mathbf{BC}}{|\mathbf{BA}||\mathbf{BC}|} - cos(\alpha_0)\right]\frac{\mathbf{BC}}{|\mathbf{BC}|} \\
\mathbf{F_B} &= k\left[\frac{\mathbf{BA}.\mathbf{BC}}{|\mathbf{BA}||\mathbf{BC}|} - cos(\alpha_0)\right]\left(\frac{\mathbf{BC}}{|\mathbf{BC}|} + \frac{\mathbf{BA}}{|\mathbf{BA}|}\right) \\
\mathbf{F_C} &= -k\left[\frac{\mathbf{BA}.\mathbf{BC}}{|\mathbf{BA}||\mathbf{BC}|} - cos(\alpha_0)\right]\frac{\mathbf{BA}}{|\mathbf{BA}|}
\end{aligned}$$

Ce type de ressorts permet de contraindre trois points consécutifs à former un angle proche de celui au repos. Ainsi, la force générée est une force de flexion puisque son calcul se base sur l'angle de courbure formé par les trois points.

On remarque au passage que la somme des trois forces est bien nulle, ce qui vérifie la propriété de conservation d'énergie.

Le comportement d'un modèle masses-ressorts est directement lié à ces déformations, autrement dit aux ressorts employés. Ainsi, la plupart des ressorts décrits ci-dessus ne sont pas suffisants pour obtenir le résultat escompté. Pour cela, de nombreux travaux mettent en jeux des techniques permettant de simuler des comportements données en introduisant des termes spécifiques ou des contraintes sur les ressorts.

Par exemple, [Pro95] propose de définir une contrainte sur l'élongation des ressorts. Les ressorts sont munis d'une élongation maximale qui permet de limiter le comportement trop élastique des simulations masses-ressorts sans pour autant augmenter de manière démesurée le coefficient de raideur des ressorts. Ainsi le modèle proposé permet de simuler des tissus très peu élastiques sans soulever de problèmes d'intégration numérique (problème raide). La prise en compte de la contrainte est réalisée après l'intégration

numérique par un processus de dynamique inverse. Ce processus consiste à déplacer (rapprocher sur le segment) les deux nœuds reliés par le ressort dans le but d'atteindre la déformation maximale autorisée afin de vérifier la contrainte. En ce qui concerne le déplacement des extrémités, il est effectif sur les nœuds libres du maillage. Le déplacement est donc partagé entre deux extrémités libres, supporté par l'extrémité libre si l'autre est fixe et inexistant si les deux extrémités sont fixées. Le résultat comparatif obtenu par Provot est présenté sur la figure (1.1).

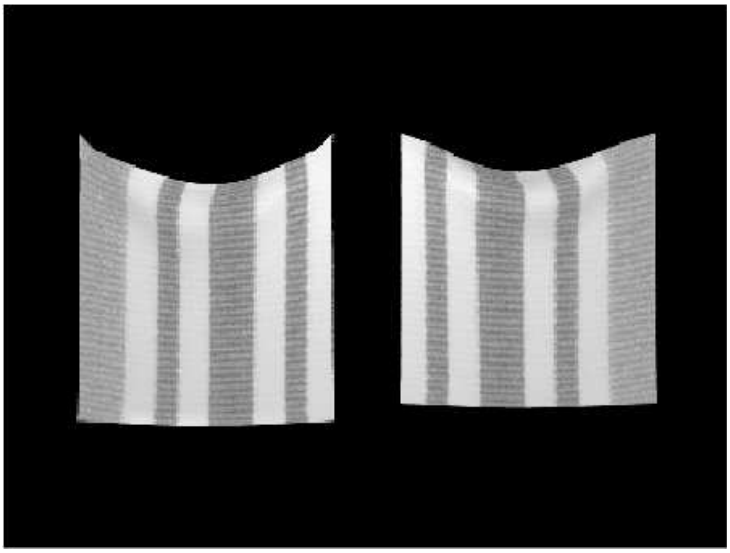


FIG. 1.1 – (Gauche) Modèle élastique (problème raide). (Droite) Modèle de Provot. (Extrait de [Pro95])

[DRG03, NNB04, NN03] proposent d'introduire un terme d'hystérésis dans le cadre de simulations textiles. Ce terme permet d'introduire des retards dans les déformations et les retours de déformation et également des phénomènes plastiques. Denise et al.[DRG03] présentent leurs résultats sur des simulations masses-ressorts (modèle de [Pro95]) de tissus.

Il est possible de doter un modèle masses-ressorts d'énergies de déformation plus élaborées comme le proposent Bourguignon et Cani [BC00] en contrôlant l'anisotropie des déformations. Pour cela, les propriétés élastiques du matériau peuvent être exprimées en tout point de l'objet sur différents axes. Les forces résultantes sont alors distribuées sur les sommets des éléments (tétraèdres ou héxaèdres) à l'aide de coordonnées barycentriques. Des énergies contrôlant la préservation du volume peuvent également être introduites. En ce sens, Bourguignon et Cani [BC00] proposent de placer pour chaque élément (tétraèdre ou héxaèdre) de l'objet des ressorts reliant un sommet à l'iso-barycentre de l'élément.

Certains travaux permettent de simuler un modèle masses-ressorts sur des architectures distribuées comme le propose Zara et al. [ZFV02].

Ce modèle physique d'objets déformables est très largement utilisé dans certains domaines comme la simulation de vêtements [BWK03, BHG92, BHW94b, BHW94a, HB00, EWW96, Pro95, BW98, VMT00, VCMT95, LPC95], la simulation chirurgicale [TGG00] ou l'animation faciale [NT98, ZPS01].

Cependant, le modèle masses-ressorts est difficile à identifier au réel. Van Gelder et Wilhelms[VGW97] ont montré qu'il n'existait pas de paramètres de modèles discrets surfaciques à base de triangles, permettant de reproduire exactement le comportement d'une membrane. De plus, l'utilisation de ressorts introduit des problèmes d'oscillation autour de la position d'équilibre. Et, le fait de définir un même objet avec plus de ressorts montés en série impose d'augmenter la raideur de chaque ressort de manière à assurer une raideur globale constante. Ceci demande de choisir un pas de temps assez petit pour l'intégration numérique et mène à un problème raide (*stiff problem*).

1.1.2.2 Modèles continus à discrétisation spatiale

Les modèles continus permettent de prendre en considération les déformations de l'objet en tout point du volume. Ainsi, la simulation se rapproche de la réalité en essayant de rendre compte des déformations continues de la matière. Plus précisément, les modèles continus à discrétisation spatiale permettent d'étudier de manière plus précise des zones spatiales du matériau en utilisant une discrétisation plus fines dans ces zones. Ainsi ce type de modèle est régulièrement employé en résistance des matériaux pour mesurer les déformations internes d'un objet et donc prévoir les limites de résistance du matériau ainsi que les points de rupture de l'objet.

Le modèle continu à discrétisation spatiale le plus utilisé est le modèle élément fini, cependant il existe beaucoup de méthodes dérivées possédant leurs spécificités. La première méthode est la méthode des différences finies qui consiste à discrétiser un problème continu soit sur la dimension spatiale (définition d'une grille régulière), soit sur la dimension temporelle (pour réaliser l'intégration numérique). Les méthodes suivantes sont les méthodes éléments finis ou dérivées des éléments finis (simulation des déformations du volume ou de la surface). Enfin, les deux dernièrs modèles sont les masses-tenseurs et le modèle hybride qui sont des modèles issus des élements finis avec cependant des caractéristiques communes au modèle discret (masses-ressorts).

1.1.2.2.1 Méthode des différences finies (*Finite Difference Method*) La méthode des différences finies permet d'approximer la variation locale d'une fonction f (i.e. évaluer f') à l'aide d'une discrétisation de la fonction f. Effectivement, le développement limité de f (formule de Taylor) permet d'obtenir à l'ordre un :

$$f'(x) = \frac{f(x+\delta) - f(x)}{\delta} + \mathcal{O}(\delta) \tag{1.4}$$

en notant $f^{(n)}_{i+j} = f^{(n)}(x + j\delta)$ et en conservant uniquement les termes d'ordre un, l'équation (1.4) devient $f'_i = \frac{f_{i+1} - f_i}{\delta}$. Cette équation est appelée la *first forward difference*. On définit de la même manière la *first backward difference* comme étant : $f'_i = \frac{f_i - f_{i-1}}{\delta}$.

Par ce principe, il est possible d'exprimer toutes les dérivées successives de la fonction f à l'aide d'évaluation discrète de f. Par exemple, à l'ordre deux, on trouve la *second forward difference* :

$$f''_i = \frac{f'_{i+1} - f'_i}{\delta} = \frac{f_{i+2} - 2f_{i+1} + f_i}{\delta^2}$$

et la *second backward difference* :

$$f''_i = \frac{f'_i - f'_{i-1}}{\delta} = \frac{f_{i-2} - 2f_{i-1} + f_i}{\delta^2}$$

Terzopoulos et al.[TPBF87] propose de simuler des modèles continus (paramétriques) par les équations physiques locales (donc continues) et de discrétiser ces équations à l'aide de la méthode des différences finies.

Pour cela, ils découpent l'espace paramétrique à l'aide d'une grille régulière définissant ainsi des nœuds discrétisant le modèle. Chaque nœud possède des informations physiques relatives au matériau et regroupe également toutes les informations relatives à des fonctions continues du modèle. Autrement dit, toute fonction continue relative au modèle est discrétisée par son évaluation en chaque nœud de la grille.

Ainsi, l'énergie continue de déformation, qui est une fonction continue du modèle, est approximée au niveau de chaque nœud. Comme l'expression de cette énergie fait intervenir les dérivées (spatiales) première et seconde de la fonction position, cette approximation est effectuée à l'aide de différences finies d'ordres un et deux.

La discrétisation de l'énergie de déformation impose aux équations physiques du mouvement d'être également discrétisées en ces points pour pouvoir résoudre le système dynamique.

Les auteurs proposent également une intégration numérique, des équations différentielles ordinaires obtenues, par la méthode des différences finies appliqué non plus sur la dimension spatiale mais sur la dimension temporelle.

L'utilisation de la méthode des différences finies pour intégrer les équations dynamiques sur la dimension temporelle est aussi utilisée par Qin et Terzopoulos[QT96]. Cependant, les équations dynamiques du modèle ne sont pas obtenues par la méthode des éléments finis mais par les équations de Lagrange (cf. section 1.1.2.3.2).

Debunne et al.[DDBC99] proposent de simuler les déformations d'un objet non plus à l'aide d'une grille régulière comme peut le proposer la méthode des différences finies, mais à l'aide d'une grille irrégulière. De ce fait, les auteurs ont besoin d'opérateurs de dérivation adaptés. Ainsi, ils proposent une approximation discrète des opérateurs *Laplacien* et *gradient de la divergence* pour simuler les déformations d'un objet sur une grille non régulière.

1.1.2.2.2 Méthode des éléments finis (*Finite Element Method*) La méthode des éléments finis est un outil mathématique permettant de résoudre un problème aux équations différentielles partielles, comme par exemple l'étude des déformations d'un objet continu. Le principe de la méthode est de discrétiser le domaine de continuité en une partition de sous-domaines discrets et un jeu de fonctions d'interpolation permettant d'évaluer un champ continu à partir des éléments discrets.

Dans l'étude des déformations d'un objet continu, la loi physique de constitution du matériau est utilisée pour déterminer les énergies de déformation de l'objet (cf. annexe A). Ceci apporte les équations de déformation locale à l'objet. Ensuite, la méthode des éléments finis opère une discrétisation de l'objet en sous-éléments sur lesquels les équations s'expriment. Ainsi, la formulation obtenue est dite faible puisque les équations finales ne sont plus locales mais relatives à l'objet (discrétisé) tout entier.

Quelque soit l'énergie de déformation choisie (loi de constitution linéaire ou non), la résolution d'un système physique par la méthode des éléments finis peut être statique ou dynamique.

Simulation statique :

Une résolution statique cherche simplement l'état d'équilibre du système par la relation

$$K(\mathbf{U})\mathbf{U} = \mathbf{F}$$

où $K(\mathbf{U})$ est la matrice de raideur relative au matériau simulé, $\mathbf{F}$ le bilan de force et $\mathbf{U}$ le vecteur déplacement de l'ensemble des nœuds de l'objet discrétisé.

Si les déformations sont non-linéaires par rapport aux déplacements, la matrice K dépend du vecteur $\mathbf{U}$ des déplacements. Alors que si les déformations sont linéaires par rapport aux déplacements, K est indépendante du vecteur des déplacements $\mathbf{U}$.

Déformations linéaires :

Si la loi de constitution choisie induit des déformations linéaires, la matrice de rigidité K du système statique ne dépend pas du vecteur des déplacements $\mathbf{U}$. Ainsi, le système d'équations obtenu est linéaire et peut donc être résolu par différentes méthodes :

Méthodes directes par inversion de la matrice K, par décomposition LU ou QR.

Méthodes itératives relaxation (Jacobi ou Gauss-Seidel), projection (gradient (bi-)conjugué, méthode de Lanczos).

La simulation statique en déformation linéaire apporte une matrice de rigidité K définie et positive. Or, un système d'équations linéaires possédant de telles propriétés se résoud très facilement par la méthode du gradient conjugué [NvdS00, NvdS01].

Déformations non-linéaires :

Si la loi de constitution choisie induit des déformations non-linéaires, la matrice de rigidité K du système statique dépend du vecteur des déplacements $\mathbf{U}$. Ainsi, le système d'équations obtenu est non-linéaire et la résolution peut alors être réalisée par des **méthodes itératives** comme Newton-Raphson, la méthode de la sécante, la méthode du point fixe, la méthode d'Aitken, la méthode de Broyden.

Déformations quasi non-linéaires :

Cotin [CDA99] souligne que le comportement biomédical d'un tissu mou se rapproche d'une loi constitutive visco-élastique non linéaire. Or, une telle loi de déformation est difficilement utilisable en temps réel à cause des temps de calcul prohibitifs.

Pour cela, les auteurs proposent une solution en deux étapes :

– D'abord, utiliser la méthode des éléments finis avec une loi de constitution linéaire pour simuler de manière statique l'objet. Ceci permet d'obtenir le système :

$$K\mathbf{U} = \mathbf{F}$$

Les auteurs s'appuient sur l'aspect linéaire des déformations pour les mettre en forme à l'aide de tenseurs et ainsi effectuer des précalculs pour un jeu de paramètres et de contraintes donné. Ainsi, lors de la simulation, les déplacements sont déterminés de manière rapide et efficace à l'aide des tenseurs précalculés.

– Ensuite, une seconde passe est effectuée à l'aide d'un feedback d'informations pour déterminer un comportement quasi non-linéaire. Ce comportement est obtenu en décomposant le déplacement $\mathbf{u}$ (et la force $\mathbf{f}$) en une partie normale $\mathbf{u_n}$ ($\mathbf{f_n}$) et une partie tangentielle $\mathbf{u_t}$ ($\mathbf{f_t}$). Les auteurs déterminent alors un déplacement relatif à un comportement non-linéaire comme étant

$$\hat{\mathbf{u}} = \mathbf{u_n} + P(|\mathbf{u_t}|)\frac{\mathbf{u_t}}{|\mathbf{u_t}|}$$

où P est une approximation polynomiale du déplacement radial par rapport au déplacement axial.

De cette manière, les auteurs offrent la possibilité de simuler, en temps réel, des tissus mous au comportement quasi non-linéaire.

La méthode des éléments finis statique est employée par exemple par Gourret et al.[GMTT89] pour la simulation d'objets et la déformation de corps humain.

Cotin [CDA96] utilise également cette méthode mais dans un cadre de simulation chirurgicale avec des déformations linéaires (donc en petits déplacements). Les auteurs proposent d'effectuer le précalcul des déformations de chaque nœud pour un déplacement élémentaire. Le résultat est alors stocké sous forme d'un tenseur 3×3. Lors de la simulation, les déformations effectives des nœuds sont alors calculées à l'aide de ces tenseurs par une simple combinaison linéaire.

Bro-Nielsen et Cotin [BNC96] proposent d'utiliser une technique dite de *condensation* qui consiste, une fois le système formé, à supprimer l'ensemble des nœuds internes à l'objet pour ne conserver dans les équations que les nœuds de surface. Cette astuce permet de diminuer la taille du système sans changer la simulation. Pour cela, le système est ré-écrit en séparant les nœuds de surface (indicés par s) et les nœuds internes (indicés par i) :

$$\begin{pmatrix} K_{ss} & K_{si} \\ K_{is} & K_{ii} \end{pmatrix} \begin{pmatrix} \mathbf{U_s} \\ \mathbf{U_i} \end{pmatrix} = \begin{pmatrix} \mathbf{F_s} \\ \mathbf{F_i} \end{pmatrix}$$

En substituant $\mathbf{U_i}$ dans la première équation matricielle avec son expression obtenue à l'aide de la seconde équation matricielle, il en résulte :

$$\left(K_{ss} - K_{si}K_{ii}^{-1}K_{is}\right)\mathbf{U_s} = \mathbf{F_s} - K_{si}K_{ii}^{-1}\mathbf{F_i}$$

Au final, le système est restreint aux nœuds de surface. Mais il est toujours possible de retrouver le déplacement d'un nœud du volume grâce à l'équation matricielle (qui a permis d'effectuer auparavant le changement de variable) :

$$\mathbf{U_i} = K_{ii}^{-1}\left(\mathbf{F_i} - K_{is}\mathbf{U_s}\right)$$

Wu et Heng[WH04] proposent d'utiliser cette technique de condensation non plus dans le but de séparer les nœuds de surface des nœuds volumiques mais, les nœuds appartenant à une zone particulière des autres nœuds. Par exemple, en simulation chirurgicale, une opération porte généralement sur une petite partie d'un organe. C'est donc les déformations de cette zone qui sont les plus intéressantes. Pour cela, les auteurs proposent de définir trois types de nœuds, les nœuds de la zone d'opération, les nœuds en dehors de cette zone et les nœuds à la frontière des deux zones. Dans un premier temps, les auteurs emploient la méthode de condensation pour obtenir les déplacements des nœuds de la zone d'opération et de la frontière. Enfin, une seconde application de la méthode de condensation permet d'obtenir les déplacements des nœuds de surface de la zone hors-opération.

Simulation dynamique :

En simulation dynamique, les phénomènes transitoires, les oscillations et les propagations d'ondes sont recherchés en introduisant une partie dynamique et cinétique dans les équations du mouvement

$$M\frac{d^2\mathbf{U}}{dt^2} + C\frac{d\mathbf{U}}{dt} + K(\mathbf{U})\mathbf{U} = \mathbf{F}$$

où M est la matrice des masses et C la matrice d'amortissement. Il est à noter que les déformations peuvent être linéaires ou non-linéaires, cela n'influe pas sur la méthode employée pour résoudre le système dynamique.

Bro-Nielsen et Cotin[BNC96] proposent plusieurs modèles éléments finis en commencant par une description statique puis, dynamique. Mais ils résolvent les équations dynamiques obtenues en utilisant les differences finies sur la dimension temporelle. En procédant ainsi, les auteurs obtiennent un système d'équations linéaires dont les inconnues sont les déplacements. En effet, la discrétisation de la dimension temporelle supprime des équations toute notion de temps et donc les vitesses et les accélérations des déplacements.

Pour accélérer la résolution du système d'équations, il est envisageable de concentrer la masse (technique du *mass lumping*) de l'objet sur les nœuds discrets du maillage dans le but d'accélérer les calculs lors de la résolution du système comme le proposent Bro-Nielsen et Cotin[BNC96].

La méthode de condensation permet de ne résoudre le système d'équations que pour les éléments de surface. Il existe une méthode qui définit les déformations d'un objet en ne considérant que sa surface dès l'établissement des équations : la méthode des éléments de surface.

1.1.2.2.3 Méthode des éléments de surface (*Boundary Element Method*) La méthode des éléments de surface est la méthode des éléments finis exprimée non plus dans le volume mais sur la surface. Cette transition se fait, sans perte d'informations, à l'aide du théorème de la divergence (théorème de *Gauss* ou formule d'*Ostrogradski*) :
Soit un objet de volume V (et de surface S) et $\vec{u}$ [2] un champ vectoriel [3] :

$$\int_S \vec{u}.\,d\vec{s} = \int_V div(\vec{u})\,dv \tag{1.5}$$

En utilisant ce théorème, il est possible de passer d'un problème intégral volumique en un problème intégral surfacique tout en conservant les mêmes caractéristiques et donc en résolvant le même problème.

Ainsi, la méthode des éléments de surface découle directement de la méthode des éléments finis à ceci près qu'elle ne considère que les nœuds de surface du modèle. Ainsi, les déformations de l'objet ne se propagent plus de proche en proche dans le volume mais affectent directement sa surface. Et, l'ensemble des nœuds du maillage du modèle physique se trouve en périphérie de l'objet. Donc tous les nœuds du maillage simulés sont directement observables et apportent une information visuelle au contraire des FEM qui définissent des nœuds dans le volume qui n'ont d'effets observables que par propagation sur un nœud de surface. Les BEM permettent donc de simuler un objet avec moins de nœuds que les FEM (puisque les nœuds appartenant au volume ne sont plus considérés) donc moins d'équations. Cependant, chaque nœud d'un maillage BEM comporte un vecteur déplacement (comme pour les FEM) mais aussi un vecteur traction. Donc pour une discrétisation spatiale en n^s nœuds de surfaces, le problème est de calculer $6n$ inconnues (au lieu de $3n^v$ pour les FEM, avec n^v le nombre de nœuds discrétisant le volume). De plus, les matrices généralement mises en oeuvres ne sont pas creuses au contraire des FEM.

James et Pai[JP99] proposent d'utiliser la méthode des BEM pour simuler de manière quasi-statique des objets déformables possèdant des propriétés physiques isotropes et homogènes, avec une élasticité linéaire. De tels objets ont un comportement dicté par l'équation de Navier :

$$(N\mathbf{u})(\mathbf{x}) + \mathbf{b}(\mathbf{x}) = \mathbf{0} \qquad \forall \mathbf{x} \in \Omega$$

[2] la notation fléchée apporte plus de clarté dans le théorème de Gauss

[3] $div(\vec{u}) = \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}$

où Ω représente le volume de l'objet (Γ sa surface), N l'opérateur différentiel linéaire de second ordre, $\mathbf{u}$ le champ déplacement et $\mathbf{b}$ le terme dû aux forces appliquées à l'objet, comme le champ de gravité. Le principe des BEM est schématisé sur la figure (1.2). Sur cette figure, on distingue les zones surfaciques Γ_u (en contact) dont le déplacement est contraint, des zones surfaciques libres Γ_p dont le déplacement est déterminé par la simulation.

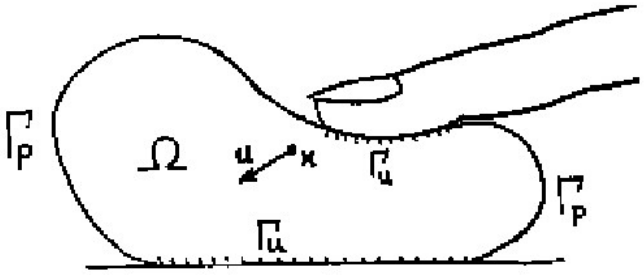


FIG. 1.2 – *Boundary Element Method* notation (Extrait de [JP99])

Cette équation forme, avec les conditions aux limites, le problème aux valeurs limites (*Boundary Value Problem* : BVP). Pour déterminer les champs déplacements $\mathbf{u}$ et tractions $\mathbf{p}$ sur la surface Γ de l'objet, les auteurs utilisent une formulation intégrale surfacique de l'équation de Navier :

$$\forall \mathbf{x} \in \Gamma \ , \ c(\mathbf{x})\mathbf{u}(\mathbf{x}) + \int_\Gamma p^*(\mathbf{x},\mathbf{y})\mathbf{u}(\mathbf{y})d\Gamma(\mathbf{y}) = \int_\Gamma u^*(\mathbf{x},\mathbf{y})\mathbf{p}(\mathbf{y})d\Gamma(\mathbf{y}) + \int_\Omega u^*(\mathbf{x},\mathbf{y})\mathbf{b}(\mathbf{y})d\Omega(\mathbf{y}) \qquad (1.6)$$

où c est une matrice $[c_{ij}]$ relative à la continuité de la surface (qui n'a pas besoin d'être calculée explicitement (voir [JP99] pour plus de détails)), la matrice $u^*(x,y)$ est définie par $[u^*_{ij}]$ qui représente le déplacement dans la direction j au point $\mathbf{y}$ du champ déplacement. Ce déplacement est dû à un effort unitaire appliqué dans chacune des i directions au point $\mathbf{x}$. $p^*(\mathbf{x},\mathbf{y})$ est définie de la même manière à l'aide, cette fois-ci, du champ traction.

Les auteurs considèrent alors qu'aucune force ne s'applique sur l'objet donc la fonction $\mathbf{b}$ est nulle. Ainsi, il ne reste plus que des intégrales surfaciques dans l'équation 1.6.

La mise en place des BEM s'effectue alors en 3 étapes :

- Discrétiser la surface Γ en une partition de N éléments de surface polygonaux définissant n sommets. Chacun de ces sommets possède une information de déplacement et de traction. Ainsi, on retrouve les champs déplacement $\mathbf{u}$ et traction $\mathbf{t}$ sur l'ensemble de la surface à l'aide de fonctions d'interpolation appliquées aux sommets du maillage.
- Appliquer les équations physiques intégrales (cf. équation 1.6) sur chaque élément de surface. En introduisant les fonctions d'interpolation, ces équations se définissent à l'aide des n points du maillage. On obtient donc un système de $3n$ équations avec $6n$ inconnues (le déplacement et la traction de chaque sommet du maillage) de la forme :

  $$H\mathbf{u} = G\mathbf{t}$$

  où H et G sont des matrices $3n \times 3n$ denses.
- Appliquer les conditions aux limites du problème considéré (BVP). Cela consiste simplement à fixer la valeur de certains degrés de liberté qu'ils soient en déplacement ou en traction (ces contraintes correspondent aux nœuds surfaciques des zones Γ_u sur la figure (1.2)). Ainsi, ces degrés de liberté possèdent une valeur et ne sont donc plus des inconnues du système d'équations. En fixant $3n$ valeurs, on diminue donc le nombre d'inconnues à $3n$ et le système devient ainsi consistant (i.e. le même nombre d'équations et d'inconnues). Le système résultant est alors noté :

  $$A\mathbf{V} = \mathbf{Z} \qquad (1.7)$$

  où les $3n$ inconnues $\mathbf{V}$ sont regroupées dans la partie gauche de l'équation.

Ainsi, les nœuds appartenant aux zones Γ_u sont fixés par les conditions aux limites, et les nœuds dans les zones Γ_p sont déplacés à l'aide de leur équation d'équilibre.

James et Pai[JP99] proposent d'utiliser les BEM dans une simulation temps-réel, pour cela ils utilisent la cohérence temporelle du système en précalculant l'ensemble des matrices nécessaires à la résolution du système. Et, dès qu'une interaction apparaît, les conditions aux limites du problème BVP changent, mais le système n'est pas recalculé dans son intégralité, il est simplement mis à jour en utilisant la formule de Shermann-Morrison-Woodbury [GVL96]. Cette formule permet de calculer rapidement l'inverse d'une matrice à la condition que celle-ci s'écrive sous une forme particulière :

$$\begin{aligned} \text{Si } A &= B + CD^T \\ \text{Alors } A^{-1} &= B^{-1} - B^{-1}C\left[I_d + D^T B^{-1} C\right]^{-1} D^T B^{-1} \end{aligned}$$

Dans l'hypothèse où l'on peut définir des mises à jour, d'une matrice A, sous la forme $C.D^T$, cette formule permet alors de calculer rapidement les mises à jour nécessaires de la matrice inverse de A. Les auteurs utilisent la formule de Shermann-Morrison-Woodbury pour résoudre directement le système d'équations linéaires (1.7) afin de déterminer efficacement les inconnues du problème (déplacements et tractions) en temps-réel.

Les BEM fournissent à la fois les déplacements et les tractions de la surface ainsi, un retour d'effort peut être mis en place avec une plus grande précision que dans le cas des FEM où les forces sont calculées numériquement par dérivation des déplacements. Cette mise en place est décrite plus en détails dans [JP01].

1.1.2.2.4 Méthode des éléments longs (*Long Element Method*) Costa et Balaniuk[CB01a, CB01b] proposent une méthode des éléments finis basée sur une décomposition du volume d'un objet en N éléments parallélépipèdiques, la méthode des éléments longs (*Long Element Method* LEM). Cette méthode considère les pressions excercées sur les éléments pour former l'équation d'équilibre statique.

Remarque : Les éléments longs ne se déforment que sur une seule direction (celle de la longueur), ainsi, toute les données vectorielles (forces, pressions...) peuvent s'exprimer sous forme scalaire.

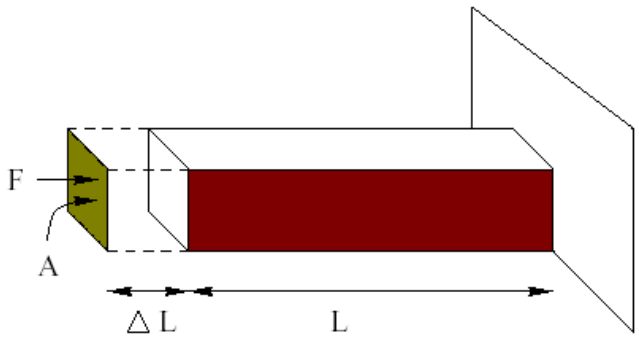


Fig. 1.3 – Un élément long (LEM) (Extrait de [SLC01])

La pression est d'abord définie comme étant le rapport d'une force par unité d'aire (cf. figure (1.3)) :

$$P = \frac{F}{A}$$

Cette pression produit une déformation. Or, pour de petits déplacements, cette déformation est linéaire par rapport au déplacement : $\epsilon = E\frac{\Delta L}{L}$ avec E l'élasticité du matériau. Comme la déformation est relative à l'inverse de la longueur L, la force est relative au déplacement ΔL : $F = K\Delta L$ avec $K = \frac{AE}{L}$. Il est à noter que K n'est pas une constante mais dépend de L.

La simulation statique de ce modèle recherche simplement l'état d'équilibre, à savoir l'égalité des pressions externe et interne. Or la pression externe se décompose en une pression atmosphérique et une

pression dûe à la déformation si elle existe. De même, la pression interne se décompose par la pression du fluide et la pression sous l'effet de la gravité :

$$\begin{array}{rcc} P_{int} & = & P_{ext} \\ \Leftrightarrow \quad P_{fluide} + dgh & = & P_{atm} + E\frac{\Delta L}{L} \\ \Leftrightarrow \quad E\frac{\Delta L}{L} - \Delta P & = & dgh \end{array}$$

où d est la densité du fluide incompressible, g la gravité, h la distance entre la partie haute du fluide et le point où la pression est calculée (Principe de Pascal : la pression d'un fluide sur un corps est proportionnelle à la profondeur d'immersion de ce corps), P_{fluide} la pression du fluide, P_{atm} la pression atmosphérique et $\Delta P = P_{fluide} - P_{atm}$.

En dissociant les éléments longs manipulés (et donc contraints en déformation) des éléments libres, on obtient finalement l'équation d'équilibre pour chaque élément long i :

$$\begin{cases} E_i\frac{\Delta L_i}{L_i} - \Delta P_i = d_i g_i h_i & \text{si l'élément i est libre} \\ \Delta L_i = y_i & \text{sinon} \end{cases} \tag{1.8}$$

Ceci procure les équations de déformation de chaque élément, il suffit maintenant de définir les connexions entre les éléments longs. Ceci se réalise par le biais de deux conditions :

- En utilisant le principe de Pascal, qui stipule que la pression excercée sur un fluide confiné dans un récipient fermé est répartie de façon homogène dans le fluide, il en résulte l'équation :

  $$\Delta P_i = \Delta P_j \quad \forall (i,j)$$

  il est alors possible de supprimer l'indice i sur le terme ΔP_i dans l'équation (1.8).
- Le fluide est considéré comme incompressible, ce qui veut dire que l'on doit avoir la propriété de conservation du volume :

  $$\sum_{i=1}^{N} A_i \Delta L_i = 0 \tag{1.9}$$

  Cette équation définit la conservation du volume pour les éléments longs donc également pour l'objet approximé par des éléments longs. Mais les deux volumes ne sont pas forcément égaux.

Les auteurs proposent alors de compléter le modèle en introduisant dans l'équation statique des éléments libres (cf équation (1.8)) un terme de tension surfacique :

$$P = \sum_j k_j (\Delta L_i - \Delta L_j) A_i$$

qui lie les déformations d'un élément i à celles de ses voisins j.

On regroupe alors les N équations d'équilibre (1.8) de chaque élément long et l'équation de conservation du volume (1.9) pour obtenir un système d'équations linéaires de $N+1$ équations à $N+1$ inconnues (les N ΔL_i et la pression ΔP) :

$$A\mathbf{x} = \mathbf{B}$$

La matrice A est liée à l'environnement et ne peut donc faire l'objet de précalculs. Les auteurs proposent donc d'utiliser la méthode itérative du gradient bi-conjugué pour approcher la solution du système.

Les FEM définissent des conditions aux limites du problème qui fixent certains nœuds du maillage. Dans le cas des LEM, l'objet est entièrement défini par des éléments longs et par des conditions aux limites de pression et de conservation du volume qui s'expriment directement sur ces éléments longs. Donc la méthode des LEM n'a plus besoin d'un maillage de l'objet. Mais les LEM n'en restent pas moins une méthode de simulation de corps déformable à discrétisation spatiale. En effet, l'ensemble des éléments longs approxime spatialement le volume de l'objet simulé.

Sundaraj et al.[SLC01] utilisent les LEM statiques pour un objet polygonal quelconque. Pour cela, ils proposent un algorithme générique qui discrétise le volume de l'objet en un ensemble d'éléments longs. Cet algorithme se base directement sur les sommets du maillage polyèdrique et sur des directions de recherche. Un lancer de rayon est alors effectué pour déterminer la facette formant l'autre extrémité de l'élément long.

Sundaraj et Laugier[SL02] proposent d'étendre ce principe pour des grands déplacements. Pour cela, ils reprennent le principe de conservation du volume et considèrent alors que la surface A_i d'un élément i peut varier (alors qu'elle est considérée comme constante en petits déplacements). Ainsi, la nouvelle équation de conservation du volume est :

$$\sum_{i=1}^{N}(A_i\Delta L_i + L_i\Delta A_i) = 0$$

Or, cette équation unique se trouve sur la dernière ligne de la matrice A du système global. Il est donc possible de mettre en place un mécanisme de mise à jour de la matrice (et de son inverse), comme le proposent les auteurs, au travers de la formule de Sherman-Morrison [GVL96]. Si la mise à jour de A peut s'écrire sous la forme :

$$A \rightarrow A + \mathbf{u} \otimes \mathbf{v}$$

où $\mathbf{u}$ et $\mathbf{v}$ sont deux vecteurs, alors la mise à jour de l'inverse de A est dictée par :

$$A^{-1} \rightarrow A^{-1} + \frac{\mathbf{z} \otimes \mathbf{w}}{1 + \lambda}$$

où $\mathbf{z} = A^{-1}.\mathbf{u}$, $\mathbf{w} = (A^{-1})^T.\mathbf{v}$ et $\lambda = \mathbf{v}.\mathbf{z}$

En utilisant une méthode itérative pour résoudre les équations, les auteurs obtiennent une simulation quasi-dynamique qui recherche la solution du problème non plus instantanément mais sur un certain nombre d'itérations pendant lesquelles les résultats intermédiaires sont utilisés pour afficher l'objet.

1.1.2.2.5 Méthode de distribution de volume (*Volume Distribution Method*) Cette méthode récente a été proposée par Kenneth Sundaraj [Sun04, SLB03] et découle directement de la méthode des LEM.

Le principe est, comme pour les LEM, de définir les équations d'équilibre d'un objet volumique rempli d'un fluide incompressible. Ces équations sont établies également à l'aide du principe de Pascal et de la conservation du volume. A ceci près que la méthode des VDM ne s'appuie pas sur une discrétisation en éléments longs du volume mais directement sur le maillage triangulaire surfacique de l'objet (soit l'objet est défini directement à l'aide d'un tel maillage, soit les nœuds du maillage doivent être définis dans une première étape).

Les pressions ne sont plus définies par rapport à une élongation mais à des variations de volumes. Ainsi, le volume est distribué sur les nœuds du maillage (de même pour l'aire de la surface). Donc, chaque nœud i du maillage possède une information de surface A_i et de volume V_i (l'aire d'un triangle est distribuée homogènement sur les trois sommets, puis le volume est distribué sur les nœuds dans les mêmes proportions que l'aire). Ainsi, le déplacement ΔL_i du nœud i induit une variation du volume V_i de ΔV_i.

L'auteur définit le module de déformation B_i et le module de connectivité des déformations B_{ij} comme des paramétres physiques, définis en chaque nœud, liés au matériau.

L'effort de pression volumique au nœud i est donné par

$$P_{vp} = B_i\frac{\Delta V_i}{V_i}$$

et l'effort, en ce nœud i, dû à la tension volumique est donné par

$$P_{vt} = \sum_{j=1}^{N} B_{ij}\frac{\Delta V_i - \Delta V_j}{V_i}$$

La simulation statique est alors dirigée par l'équilibre des pressions interne et externe sur chaque nœud i du maillage :

$$\begin{aligned} & P_{ext} & = & P_{int} \\ \Leftrightarrow \quad & P_{ep} + P_{vp} + P_{vt} & = & P_{fluid} + P_{gravity} \\ \Leftrightarrow \quad & B_i \frac{\Delta V_i}{V_i} + \sum_{j=1}^{N} B_{ij} \frac{\Delta V_i - \Delta V_j}{V_i} - \Delta P_i & = & \rho_i g \delta_i \end{aligned} \tag{1.10}$$

avec $\Delta P_i = P_{fluid} - P_{ep}$ et P_{ep} la pression de l'environnement (atmosphérique), ρ_i la densité du fluide incompressible contenu dans le volume V_i, g la gravité et δ_i la mesure hydrostatique de la distance du nœud i due à la présence du fluide (profondeur d'immersion du nœud i=distance minimale du nœud à la surface du fluide). Un schéma explicatif des différents termes introduits ci-dessus est présenté sur la figure (1.4).

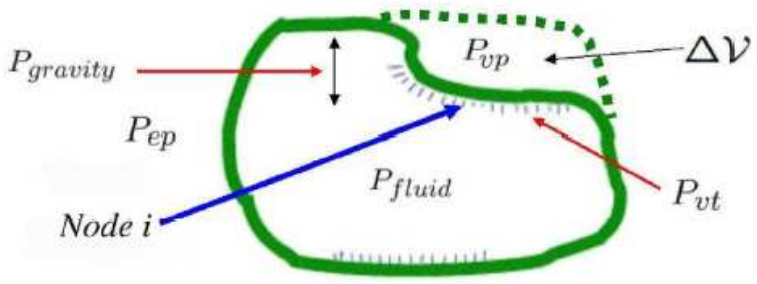


FIG. 1.4 – *Volume Distribution Method* - notation (Extrait de [Sun04])

Ensuite, on utilise les mêmes conditions aux limites que dans le cas des LEM, à savoir le principe de Pascal (qui impose, ici aussi, que le terme ΔP_i soit le même ΔP en chaque nœud) et la conservation du volume qui s'exprime ici sous la forme :

$$\sum_{i=1}^{N} \Delta V_i = 0 \tag{1.11}$$

sachant que la variation du volume lié à un nœud i est définie par rapport au déplacement de ce nœud par la relation $\Delta V_i = A_i \Delta L_i + L_i \Delta A_i$, le problème s'exprime alors en fonction des N déplacements ΔL_i des nœuds du maillage et de la variation de pression ΔP. On retrouve donc un système linéaire de $N+1$ équations (cf. équations (1.10) et (1.11)) à $N+1$ inconnues.

Le modèle VDM offre donc les mêmes caractéristiques physiques que le modèle LEM sans avoir à établir une discrétisation en éléments longs du volume. Il suffit d'établir une discrétisation spatiale de la surface de l'objet pour définir le maillage nodal animé.

Les méthodes éléments finis et différences finies traitent des déformations d'un objet de manière continue alors qu'un maillage masses-ressorts ne considère que des éléments discrets du volume. Cependant, la manipulation d'un maillage masses-ressorts est plus facile qu'un maillage éléments finis. Le modèle masses-tenseurs décrit ci-après permet d'allier les avantages de chacune des méthodes : les déformations continues de l'objet en considérant un maillage discret de masses.

1.1.2.2.6 Masses-Tenseurs La méthode des masses-tenseurs a été proposée par Cotin[Cot97] afin d'allier la simplicité des modèles masses-ressorts et la robustesse des modèles éléments finis et de permettre de modifier la topologie du maillage en temps réel (utile pour les simulations chirurgicales par exemple).

L'objet est discrétisé pour former un maillage tétraédrique, l'ensemble de la masse de l'objet est alors considérée sur les nœuds du maillage (technique du mass lumping), en donnant la même propriété aux

frottements visqueux (à savoir que le frottement visqueux d'un nœud dépend uniquement de ce nœud du maillage), cela permet de définir les équations dynamiques sur les nœuds indépendamment les uns des autres. Les équations dynamiques du mouvements sont celles de Newton :

$$M_i \frac{d^2\mathbf{U_i}}{dt^2} + D_i \frac{d\mathbf{U_i}}{dt} + \mathbf{F_i^{int}} + \mathbf{F_i^{ext}} = \mathbf{0}$$

où $\mathbf{U_i}$ est le déplacement du nœud i, M_i sa masse, D_i son coefficient de frottement visqueux, $\mathbf{F_i^{ext}}$ la somme des forces extérieures au modèle appliquées au nœud i et $\mathbf{F_i^{int}}$ les forces internes au modèle appliquées au nœud i (typiquement les forces dues aux déformations de l'objet).

Jusque là, les modèles masses-tenseurs sont équivalents à un système masses-ressorts. La différence vient de la définition de l'énergie de déformation, donc du terme $\mathbf{F_i^{int}}$. En effet, ce terme n'est pas défini à l'aide de simples ressorts mais, il est relatif à la configuration locale et continue du maillage. Le calcul de cette force interne peut être décomposée en quatre étapes :

- Définir les fonctions interpolatrices du maillage qui permettent de définir le champ déplacement en tous points du volume des tétraèdres. Pour cela, une fonction s'appuie sur les déplacements $\mathbf{U_i}$ des quatre nœuds d'un tétraèdre $\mathcal{T}_k$ et définit le déplacement en tous points de ce tétraèdre.
- Définir l'énergie élastique W_k du tétraèdre $\mathcal{T}_k$ comme une fonction des déplacements des quatre sommets.
- Calculer la force élastique $\mathbf{F^{int}_{i,\mathcal{T}_k}}$ produite par le tétraèdre $\mathcal{T}_k$ sur le nœud i en utilisant l'énergie élastique W_k.
- Sommer les contributions $\mathbf{F^{int}_{i,\mathcal{T}_j}}$ produites par les tétraèdres $\mathcal{T}_j$ ayant le nœud i en commun pour obtenir la force $\mathbf{F_i^{int}}$ qui s'excerce sur le nœud i.

Les trois premières étapes permettent de définir la force élastique d'un tétraèdre $\mathcal{T}_k$ sur un nœud i :

$$\mathbf{F^{int}_{i,\mathcal{T}_k}} = \sum_{l=0}^{3} K_{il}^{\mathcal{T}_k} \mathbf{U_{l,\mathcal{T}_k}}$$

où l représente l'indice du nœud considéré dans le tétraèdre $\mathcal{T}_k$ et $K_{il}^{\mathcal{T}_k}$ est une matrice 3×3 (ou tenseur) de raideur représentant la contribution du tétraèdre $\mathcal{T}_k$ au tenseur définissant la raideur du segment reliant les nœuds i et l (ou du nœud i si $i = l$).

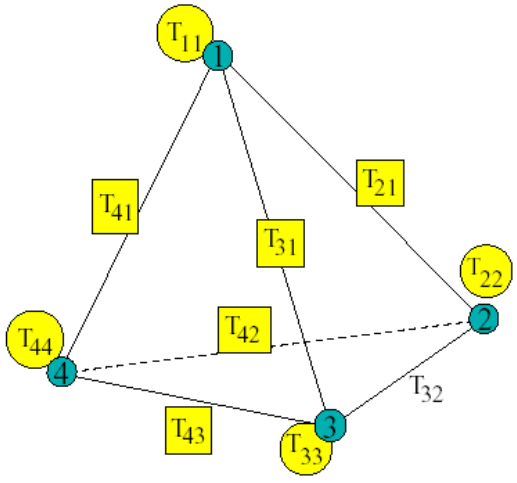


FIG. 1.5 – Schéma de distribution des tenseurs sur un tétraèdre (Extrait de [Cot97])

Pour un tétraèdre donné, il existe donc 6 tenseurs K_{il} pour les segments (avec $i \neq l$) et 4 tenseurs K_{ii} pour les sommets (cf. figure (1.5)). En considérant une élasticité linéaire, ces tenseurs s'écrivent :

$$K_{il}^{\mathcal{T}_k} = \frac{1}{36V(T_k)} \left[\lambda_k \left(\mathbf{n_{l,\mathcal{T}_k}} \otimes \mathbf{n_{i,\mathcal{T}_k}}\right) + \mu_k \left(\mathbf{n_{i,\mathcal{T}_k}} \otimes \mathbf{n_{l,\mathcal{T}_k}}\right) + \mu_k \left(\mathbf{n_{i,\mathcal{T}_k}}.\mathbf{n_{l,\mathcal{T}_k}}\right) Id_{(3\times3)}\right]$$

où $V(T_k)$ représente le volume du tétraèdre T_k et $\mathbf{n}_{l,\mathcal{T}_k}$ le vecteur normal au triangle d'indice l du tétraèdre $\mathcal{T}_k$. Les paramètres λ_k et μ_k sont les coefficients de *Lamé*, ils sont relatifs aux propriétés physiques locales (au tétraèdre $\mathcal{T}_k$) du matériau.

Les masses-tenseurs ne diffèrent des masses-ressorts que dans l'expression des énergies de déformations qui sont continues et ne dépendent pas de la discrétisation au contraire d'un modèle masses-ressorts. De plus, la suppression d'un élément engendre dans un modèle masses-ressorts un recalibrage des raideurs des ressorts pour assurer un comportement élastique cohérent alors qu'avec le modèle masses-tenseurs, il suffit de soustraire les contributions tensorielles de l'élément supprimé.

Picinbono et al.[PLDA00] proposent d'étendre les travaux de Cotin[Cot97] dans le cas d'une élasticité anisotropique toujours en petit déplacement. Et [PDA00, PDA03] proposent d'étendre les masses-tenseurs aux grands déplacements en utilisant le modèle de déformation non linéaire de *St Venant-Kircchoff*. Ce modèle se base sur le tenseur de déformation de *Green-St Venant*, également connu dans la littérature sous le nom de tenseur de *Green-Lagrange*) (cf Annexe A). Les auteurs obtiennent les résultats comparatifs, entre les modèles linéaires et non linéaires, présentés sur la figure (1.6). Les auteurs proposent également d'introduire une force sur chaque nœud qui permet de pénaliser les variations du volume et ainsi assurer une certaine conservation du volume global.

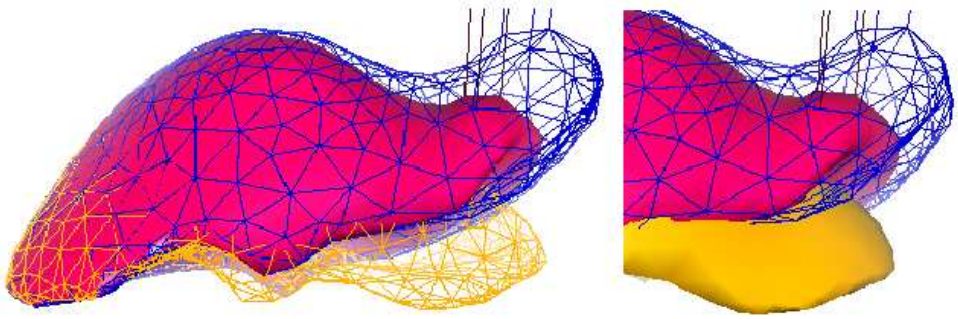


FIG. 1.6 – Déformation linéaire (fil de fer bleu), non-linéaire (solide rouge), au repos (en bas) (Extrait de [PDA00])

Contrairement au modèle FEM, le modèle masses-tenseurs possède la propriété d'être tolérant aux changements topologiques du maillage, ce qui est très utile lors de découpes ou de fractures. D'un autre coté, le modèle quasi-statique des FEM [CDA00] permet de mettre en place des précalculs qui offrent une simulation temps-réel. Cette complémentarité des deux méthodes a donné naissance au modèle hybride.

1.1.2.2.7 Modèle hybride Cotin et al. [CDA00] proposent de mixer les modèles FEM quasi-statiques et masses-tenseurs dynamiques pour allier la rapidité de l'un avec les possibilités de changement topologique du maillage de l'autre.

Pour cela, l'objet simulé est discrétisé à l'aide d'un maillage de nœuds. Deux zones distinctes sont définies sur ce maillage, l'une destinée à être simulée par la méthode des FEM quasi-statiques et l'autre par les masses-tenseurs. Pour assurer une cohérence entre les deux modèles, des nœuds communs définissent la frontière entre les deux zones ce sont les nœuds de connexion. Ces nœuds de connexion définissent des conditions limites pour chacun des deux problèmes (FEM et masses-tenseurs). On choisit alors une loi de constitution identique dans les deux modèles pour obtenir un comportement global cohérent (or les élasticités linaires des deux modèles suivent la même loi physique de constitution, c'est donc celle-ci qui est retenue).

Comme le modèle FEM est statique, il ne considére que des déplacements. Le modèle masses-tenseurs est quant à lui dynamique, il est donc capable de considérer à la fois des déplacements et des forces. Ainsi, les interactions entre les deux modèles et l'utilisateur sont schématisées sur la figure (1.7).

La boucle de simulation commence par déterminer la position d'équilibre du modèle FEM en prenant en compte les conditions aux limites (déplacements extérieurs et déplacements des nœuds de connexion).

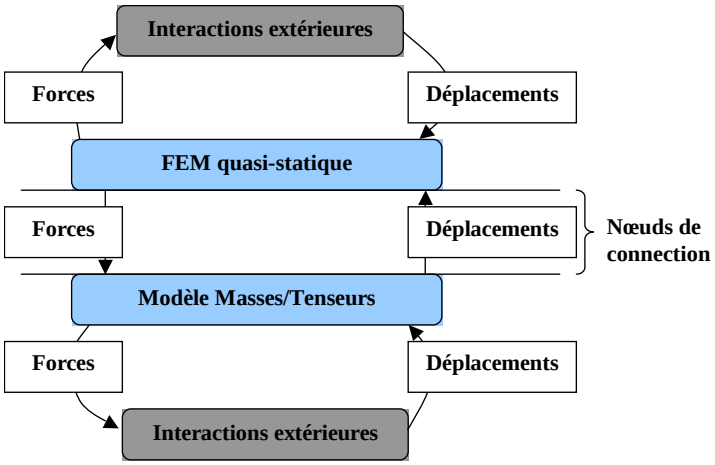


FIG. 1.7 – Boucle d'interaction d'un modèle hybride

Puis, la simulation effectue un pas d'intégration pour le modèle masses-tenseurs en prenant également en compte les conditions aux limites (forces excercées sur les nœuds de connexion et déplacements extérieurs).

Les auteurs font remarquer que les maillages des deux modèles ne sont pas forcément définis au même grain. Par exemple, le modèle FEM étant rapide à se simuler, son maillage peut être plus fin. Ainsi, à la jonction des deux maillages, certains nœuds du maillage plus fin ne sont pas forcément reliés à un nœud de l'autre maillage. Donc les deux maillages ne sont pas forcément entièrement connectés. Ceci introduit des artéfacts visuels dus à la non continuité des deux maillages comme indiqué sur la figure (1.8).

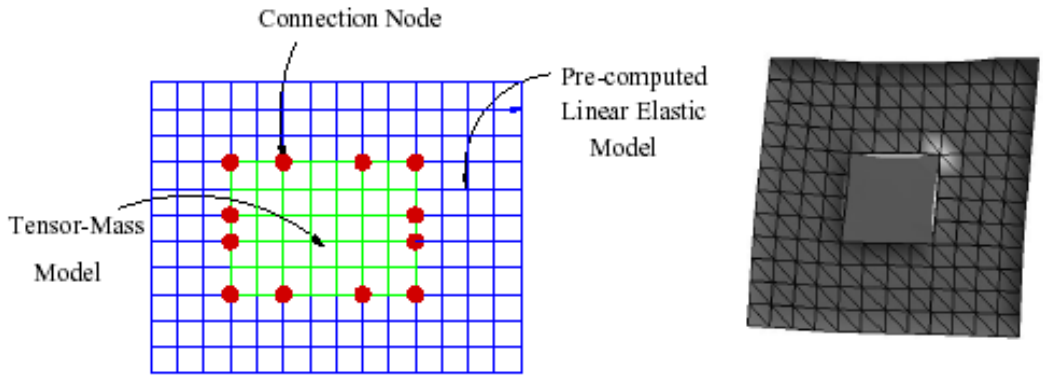


FIG. 1.8 – (Gauche) Définition des nœuds de connexion. (Droite) Modèle hybride avec 8 nœuds de connexion.

Le modèle hybride permet d'allier la rapidité des FEM quasi-statiques avec les changements topologiques rendus possibles par les masses-tenseurs. Ce modèle de simulation trouve des applications dans le domaine de la simulation chirurgicale où des déformations rapides permettent de donner du réalisme et la partie masses-tenseurs offre la possibilité de découper ou déchirer une partie d'un organe.

Les modèles étudiés dans cette section ont la particularité de se baser sur une discrétisation spatiale de l'objet (volumique ou surfacique). Cependant, si une discrétisation est nécessaire au calcul numérique, elle n'est pas forcément spatiale. La section qui suit s'étend sur les modèles continus qui ont pour caractéristique commune, une discrétisation non spatiale.

1.1.2.3 Modèles continus à degrés de liberté quelconques

Ce type de modèles permet de définir le comportement d'un objet déformable non plus au travers d'une discrétisation en points spatiaux mais à l'aide d'un jeu de paramètres renseignant la configuration déformée de l'objet. Ce jeu de paramètres est défini par la méthode choisie (et parfois libre de choix). Par exemple, l'analyse modale étudie les vibrations d'un matériau et déforme l'objet en le faisant vibrer. Il existe également le formalisme physique de Lagrange qui permet de définir un objet à l'aide d'un jeu de paramètres indépendants nécessaires et suffisants pour définir l'objet.

Nous commençons par décrire les modèles de simulation basés sur l'analyse des modes de vibration d'un matériau puis, nous détaillons les modèles fondés sur le formalisme physique de Lagrange.

1.1.2.3.1 Analyse modale pour la simulation physique dynamique Cette approche consiste à étudier les fréquences de vibration du matériau. Ainsi, lorsqu'une déformation est appliquée, on peut déterminer à quelles fréquences elle correspond et déformer l'objet, non plus par déplacements de points spatiaux, mais en faisant vibrer le matériau à certaines fréquences.

Le processus peut être décomposé en plusieurs étapes :

- Dans un premier temps, il s'agit d'exprimer les équations dynamiques d'un matériau en fonction des déplacements des nœuds d'un maillage discrétisant l'objet (par exemple avec les FEM) :

$$M\ddot{\mathbf{U}} + C\dot{\mathbf{U}} + K\mathbf{U} = \mathbf{F} \tag{1.12}$$

  avec $\mathbf{U}$ le vecteur déplacement des nœuds du maillage discret de l'objet, M la matrice des masses, C la matrice d'amortissement, K la matrice de rigidité et $\mathbf{F}$ le vecteur des forces extérieures. On remarque au passage que la matrice K est indépendante de $\mathbf{U}$, ce qui veut simplement dire que les déformations de l'objet sont forcément linéaires [SHGO02].
- Puis, il faut effectuer un changement de base dans le but d'exprimer le système dynamique sur des degrés de liberté, indépendants les uns des autres, correspondant aux différentes fréquences de vibration du matériau. Partant de l'équation dynamique (1.12), il suffit de déterminer les valeurs propres (λ_i) du système et leur vecteur propre associé $(\mathbf{v_i})$. Les vecteurs propres définissent les modes de vibration du système. Si les valeurs propres ont toutes une multiplicité de une, alors les vecteurs propres définissent une base, appelée *base modale* (associée à la transformation ϕ), dans laquelle les matrices M et K sont diagonales. [4] En définissant la matrice d'amortissement C comme une combinaison linéaire de M et K (amortissement de *Rayleigh* (cf. [Nie03])), la transformation de cette matrice par ϕ aboutit également à une matrice diagonale.
  Le système d'équation (1.12) se ré-écrit alors dans la base modale sous la forme :

$$\ddot{\mathbf{U}}^* + C^*\dot{\mathbf{U}}^* + K^*\mathbf{U}^* = \mathbf{F}^*$$

  où C^* et K^* sont deux matrices diagonales (liées aux valeurs propres λ_i), $\mathbf{U}^* = m^{-1}\phi^T\mathbf{U}$ (représente le vecteur des modes de vibration) et $\mathbf{F}^* = m^{-1}\phi^T\mathbf{F}$ (le vecteur des forces projetées sur la base modale) avec $m = \phi^T M\phi$. Voir [Pat00] pour plus de détails.

Comme les équations finales sont décorrélées et relatives à une fréquence de vibration, il est envisageable d'introduire des déformations propres à une fréquence de vibration donnée. De manière générale, un objet simulé par cette méthode est déformé au travers de ses différents modes de vibration qui deviennent donc son jeu de paramètres de déformation.

Patil [Pat00] étudie le découplage, par analyse modale, des équations différentielles du second ordre d'un système dynamique général non conservatif.

Pentland et Williams[PW89] proposent également d'utiliser cette technique d'analyse des modes de vibration d'un matériau pour simuler un objet déformable. Cependant, les auteurs ne se basent pas directement sur une décomposition modale des équations dynamiques mais sur une approximation globale par des fonctions linéaires et quadratiques (comme le font remarquer Shen et al.[SHGO02]).

[4] http ://www.cert.fr/dcsd/THESES/vincent/manuscrit_vincent/node7.html♯SECTION00222100000000000000

James et Pai[JP02] proposent de coupler cette méthode avec une animation d'objets rigides pour simuler les déformations de la peau sur les os. Les os sont simulés ou contrôlés par un mécanisme de capture de mouvement (*motion capture*). La peau est fixée aux os ainsi, dès qu'un os induit un déplacement, il provoque des vibrations dans la structure de peau et l'anime ainsi automatiquement.

1.1.2.3.2 Formalisme physique de Lagrange De nombreux travaux [RNG99, RNG00, RNN00, QT96, WW90] proposent de simuler des objets déformables à l'aide des équations physiques de Lagrange. Le formalisme physique de Lagrange est détaillé par Nocent dans [Noc99], mais nous allons donner les principes généraux de ce formalisme physique afin de mieux en comprendre le fonctionnement.

Le formalisme physique de Lagrange est basé sur le principe des puissances virtuelles de d'Alembert, énoncé en 1743[Noc99], qui permet d'obtenir une équation fonctionnelle à partir d'une équation vectorielle [5] :

$$\vec{A} = \vec{B} \quad \Leftrightarrow \quad \forall \vec{X} \ , \ \vec{A}.\vec{X} = \vec{B}.\vec{X}$$

L'équation résultante est liée au champ vectoriel $\vec{X}$. Par exemple, l'application de ce principe au PFD de Newton (cf. section 1.1.2.1) sur le champ des vitesses virtuelles aboutit à la formulation en puissances virtuelles :

$$\mathcal{P}_{int} + \mathcal{P}_{ext} = 0$$

Le formalisme de Lagrange découle directement du principe fondamental de la dynamique (PFD) énoncé par Newton (cf. section 1.1.2.1). Lagrange utilise le *principe des puissances virtuelles* (sur le champ des déplacements virtuels) sur l'équation vectorielle du PFD pour aboutir à une formulation énergétique, ce qui lui permet de se détacher d'une discrétisation spatiale de l'objet et d'obtenir des équations fonctionnelles (et non plus vectorielles). Puis il détermine la trajectoire de la particule (le déplacement virtuel correspondant à la réalité physique) dans l'ensemble des trajectoires possibles à l'aide du principe physique de *moindre action* (défini ci-après). Le paragraphe suivant décrit de manière plus détaillée l'aboutissement aux équations de Lagrange.

Lagrange définit un jeu de paramètres indépendants, permettant de définir entièrement le système matériel (l'objet ou les objets). Ces paramètres sont appelées les *coordonnées généralisées* (q_i) du système, ils sont indépendants et permettent de définir entièrement le système matériel (ils correspondent aux positions des particules dans un système Newtonien). Lagrange utilise alors le *principe des puissances virtuelles* pour exprimer les énergies cinétiques E_c et potentielles E_p du système matériel en fonction des coordonnées généralisées et de leur vitesse (ces deux termes étant dépendants du temps) et ensuite former un terme caractéristique de la configuration énergétique du système, le *Lagrangien* :

$$\mathcal{L}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) = E_c(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) - E_p(\mathbf{q}(t), \dot{\mathbf{q}}(t), t)$$

Le but de Lagrange est alors de rechercher la configuration énergétique réelle du système matériel dans l'espace des solutions possibles. Pour cela, il se base sur le principe physique de moindre action, énoncé par *Maupertuis* en 1753 [Noc99], qui stipule que [6] *parmi toutes les trajectoires possibles sur une durée donnée, celle choisie par la nature est celle pour laquelle l'action est minimale* (cela revient à rechercher une géodésique dans une métrique basée sur l'énergie).

Or, [6] *l'action est une grandeur physique qui est définie, sur une trajectoire donnée, comme le produit de l'énergie (le lagrangien) par le temps mis pour parcourir cette trajectoire.*

[7] *L'action synthétise le comportement global du système matériel sur un intervalle de temps.* Donc l'action du *Lagrangien* sur l'intervalle de temps $[a, b]$ est donnée par :

$$A = \int_a^b \mathcal{L}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t)\, dt$$

[5] La notation fléchée apporte une meilleure compréhension du principe des puissance virtuelles
[6] extrait de http ://www.reperes-basiques.com/C4.htm
[7] Extrait de [Noc99]

La configuration énergétique réelle est donc déterminée par le minimum de l'action du *Lagrangien*. On peut déterminer ce minimum en étudiant les variations δA de l'action [8] :

$$\begin{aligned}\delta A &= \int_a^b \mathcal{L}(\mathbf{q}(t)+\boldsymbol{\delta}\mathbf{q}, \dot{\mathbf{q}}(t)+\boldsymbol{\delta}\dot{\mathbf{q}}, t) - \mathcal{L}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t)\, dt \\ &= \int_a^b \frac{\partial \mathcal{L}}{\partial \mathbf{q}}.\boldsymbol{\delta}\mathbf{q} + \frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}.\boldsymbol{\delta}\dot{\mathbf{q}}\, dt\end{aligned}$$

On utilise alors une intégration par partie sur le second terme $\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}.\boldsymbol{\delta}\dot{\mathbf{q}}$:

$$\begin{aligned}\delta A &= \int_a^b \frac{\partial \mathcal{L}}{\partial \mathbf{q}}.\boldsymbol{\delta}\mathbf{q}\, dt + \left[\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}.\boldsymbol{\delta}\mathbf{q}\right]_a^b - \int_a^b \left(\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}.\boldsymbol{\delta}\mathbf{q}\right) dt \\ &= \int_a^b \left(\frac{\partial \mathcal{L}}{\partial \mathbf{q}} - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\right).\boldsymbol{\delta}\mathbf{q}\, dt + \left[\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}.\boldsymbol{\delta}\mathbf{q}\right]_a^b\end{aligned}$$

Or, on a $\boldsymbol{\delta}\mathbf{q}(a) = \boldsymbol{\delta}\mathbf{q}(b) = \mathbf{0}$ (cf. [Noc99] pour plus de détails) :

$$\delta A = \int_a^b \left(\frac{\partial \mathcal{L}}{\partial \mathbf{q}} - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\right).\boldsymbol{\delta}\mathbf{q}\, dt$$

Or, nous recherchons la valeur minimale de l'action A, cette valeur minimale est déterminée par un point stationnaire d'où l'on doit observer la relation $\delta A = 0$ pour une petite variation $\boldsymbol{\delta}\mathbf{q}$ des paramètres $\mathbf{q}$, ce qui entraîne la relation :

$$\frac{\partial \mathcal{L}}{\partial \mathbf{q}} - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}} = \mathbf{0} \tag{1.13}$$

En considérant les variables indépendantes q_i une à une, on obtient

$$\forall i\ ,\ \frac{\partial \mathcal{L}}{\partial q_i} - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} = 0 \tag{1.14}$$

qui forment les équations différentielles d'Euler-Lagrange, plus connues sous le nom d'équations de Lagrange.

Witkin et Welch[WW90] proposent de simuler un objet déformable à l'aide des équations de Lagrange non pas en définissant les coordonnées généralisées comme des points de l'espace interpolés ou approximés (cas des splines ci-après), mais comme un jeu de paramètres de déformations.

L'objet est discrétisé par un ensemble de points caractéristiques. Ensuite, lors de la simulation, la configuration déformée de l'objet est définie par les coordonnées des points caractéristiques de l'objet déformé. Alors que la configuration au repos de l'objet est donnée par des informations au niveau des points caractéristiques dans la configuration non déformée (donc ces informations sont indépendantes du temps) :

$$\mathbf{x} = R\mathbf{p} \tag{1.15}$$

où $\mathbf{x}$ est la position d'un point caractéristique de l'objet déformé, R la matrice des déformations (ces valeurs forment l'ensemble des coordonnées généralisées de l'objet) et $\mathbf{p}$ les informations des points caractéristiques dans la configuration non-déformée. Pour des déformations linéaires, $\mathbf{p}(x,y,z) = [1,x,y,z]$ alors que pour pouvoir définir des déformations non-linéaires, il faut des informations du second degré : $\mathbf{p}(x,y,z) = [1,x,y,z,xy,xz,yz,x^2,y^2,z^2]$. De cette manière, il est possible de définir toute déformation s'exprimant sous forme polynômiale.

[8] http ://mathworld.wolfram.com/Euler-LagrangeDifferentialEquation.html
et http ://www.plmsc.psu.edu/∼www/matsc597c-1997/phases/Lecture5/node4.html

La dynamique de l'objet est alors obtenue par les équations de Lagrange. Or, les coordonnées généralisées de l'objet mécanique sont les coefficients de la matrice R qui représentent les paramètres de déformation de l'objet. Donc, l'objet n'est pas simulé dans l'espace tri-dimensionnel euclidien $\mathbb{R}^3$, mais dans l'espace de ses déformations.

La proposition de Witkin et Welch[WW90] permet de simuler tout objet déformable en le contrôlant par ses déformations. Par exemple, un objet possédant une coordonnée généralisée caractéristique de son élongation connaît, à chaque itération, l'élongation qu'il subit et la dynamique de ce paramètre (la variation de l'élongation). Il est alors capable d'en déduire sa configuration spatiale courante grâce à l'équation 1.15. On peut imaginer la même chose avec des paramètres de flexion, cisaillement, torsion...

Metaxas et Terzopoulos[MT92] proposent de définir les déformations d'un objet en le décomposant en une composante rigide et une composante déformable (comme schématisé sur la figure (1.9)) :

$$\mathbf{x} = \mathbf{c} + R\mathbf{p}$$

$\mathbf{c}$ est le centre du repère lié à l'objet déformable, R est la matrice de rotation (changement de base ϕ sur le schéma) orientant le repère de l'objet (introduit 3 angles θ_i) et $\mathbf{p}$ est le point de l'objet déformé, dans le repère local, sachant que $\mathbf{p}$ se décompose en $\mathbf{p} = \mathbf{s} + \mathbf{d}$ avec $\mathbf{s}$ le point de référence et $\mathbf{d}$ le vecteur déplacement.

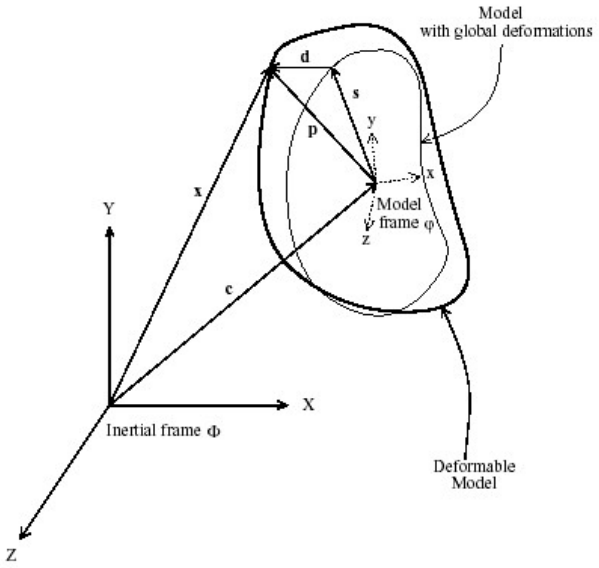


FIG. 1.9 – Schéma d'un objet déformable muni d'un repère local (extrait de [MT92]).

Cette définition apporte un jeu de paramètres étendus (position $\mathbf{c}$ et rotation θ du repère lié à l'objet déformable, paramètres des déformations globales $\mathbf{s}$ et locales $\mathbf{d}$) offrant des possibilités supplémentaires à l'aide de déformations globales et locales.

Rémion et al.[RNG99, RNG00, RNN00] ainsi que Qin et Terzopoulos[QT96] proposent d'utiliser le formalisme de Lagrange sur des modèles géométriques de type spline (cf. section 1.2.1). Comme une spline est entièrement définie par la donnée de ses points de contrôle, les coordonnées généralisées du modèle mécanique sont les coordonnées de ces points de contrôle. Or, les splines se décomposent en deux groupes :

Splines d'interpolation :

Pour les splines interpolatrices (exemple des Hermites ou Catmull-Rom), les points de contrôle sont situés sur la spline elle-même. Dans ce cas, la définition mécanique du modèle spline passe par une discrétisation spatiale.

Splines d'approximation :

Pour les splines d'approximation (exemple des B-splines ou NURBS), les points de contrôle ne sont pas situés sur le modèle géométrique (sauf cas particulier où un nœud possède une multiplicité suffisante pour forcer un point de contrôle à être sur la spline). Pour ce type de spline, les coordonnées généralisées sont donc virtuelles et spatialement décorélées du modèle physique simulé (la spline).

Quelque soit le type de spline choisi, les coordonnées généralisées sont des points virtuels de l'espace tri-dimensionnel euclidien $\mathbb{R}^3$. Ainsi, la simulation se base sur des points de l'espace $\mathbb{R}^3$ bien que la discrétisation de l'objet soit virtuelle (il n'y a pas d'échantillonnage de l'objet). L'utilisation des équations de Lagrange sur une formulation de type spline permet de conserver l'aspect continu du modèle géométrique lors de sa simulation dynamique. Le modèle simulé est bien la spline en tant que modèle continu.

1.2 Modèles 1D

Il existe de nombreuses façon de définir un modèle 1D, où chacune d'elles peut potentiellement apporter des propriétés supplémentaires par rapport aux travaux généraux qui ont été développé dans la section précédente. Aussi, il est clair que les techniques particulières aux modèles 1D doivent être détaillées.

Tout modèle mécanique 1D se base sur un modèle géométrique qui, dans un second temps, se voit être agrémenté de paramètres physiques (masse, frottements visqueux...) et animé par des équations mécaniques pour devenir un modèle dynamique. Donc, nous commençons par établir un état de l'art non exhaustif des modèles géométriques 1D avant de décrire quelques travaux menés sur la simulation physique d'objets 1D.

1.2.1 Choix géométiques pour les modèles 1D

Il existe de nombreuses manières de définir un modèle géométrique 1D, allant de la simple liaison de points ponctuels dans l'espace, jusqu'aux courbes splines rationnelles procurant de multiples propriétés mathématiques (continuité, localité, invariance par projection...). Nous allons donc détailler une sous-partie de ces modèles en suivant une progression logique en termes de propriétés mathématiques et géométriques. Nous commençons par le modèle le moins élaboré : le modèle rectiligne.

1.2.1.1 Modèle rectiligne

Un modèle géométrique rectiligne est simplement défini par un ensemble de points dans l'espace reliés par un réseau de segments rectilignes. Un tel modèle peut être défini à l'aide d'un graphe non orienté $\mathcal{G}(S, A)$ dont l'ensemble S des nœuds est constitué des points ponctuels du modèle géométrique et l'ensemble A des arcs schématise les segments rectilignes reliant des couples de points distincts de l'ensemble S.

En posant des conditions d'adjacences, on peut imposer aux arcs qu'ils soient tous contigus pour ne former qu'une ligne brisée : deux éléments de l'ensemble S n'apparaisent qu'une seule fois dans l'ensemble A, les autres éléments de S apparaissent deux fois dans l'ensemble A. On peut également utiliser les propriétés du modèle pour définir une découpe et obtenir deux courbes distinctes en possédant deux sous-graphes indépendants.

Un modèle discret défini par une ligne brisée est de continuité C^0 comme le montre la figure (1.10), ce qui peut poser des problèmes au niveau du calcul de l'éclairement de l'objet (si celui-ci est doté d'une épaisseur et donc, il définit implicitement un volume filiforme) ou encore lors du tracé des différents segments à l'aide d'une section plane (technique de l'extrusion), ou tout simplement lors du tracé de la ligne brisée, le modèle apparaît tel qu'il est : C^0. Ce modèle pauvre en continuité n'apporterait pas beaucoup de propriétés intéressantes à une simulation dynamique.

Il est donc indispensable de poursuivre cette étude des modèles géométriques sur les modèles possédant des propriétés de continuité plus intéressantes, comme les courbes à subdivision.

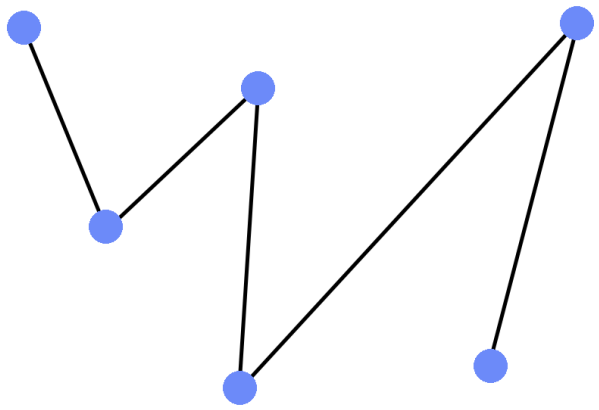


FIG. 1.10 – Exemple d'un modèle géométrique ponctuel

1.2.1.2 Courbe à subdivision

Ce type de courbe est à mi-chemin entre les courbes discrètes et les courbes continues. Une telle courbe est définie à l'aide d'un ensemble fini de points reliés les uns aux autres. En ce sens, ce modèle de courbe est discret. Cependant, un algorithme de subdivision lui est associé, permettant de redéfinir la courbe à une résolution différente, c'est à dire avec plus ou moins de points discrets. Ce schéma de subdivision tend à l'infini à créer une courbe continue basée sur les points de départ. C'est précisément en ce sens que le modèle à subdivision est continu.

La propriété de subdivision du modèle est très intéressante d'un point de vue mécanique puisqu'elle procure la possibilité de choisir la résolution adéquate à chaque instant. Malheureusement, à une résolution donnée, ce modèle reste un modèle discret.

Les propriétés des courbes à subdivision sont liées au masque de subdivision employé. En particulier, une courbe dite B-spline uniforme cubique (qui est définie plus bas) possède un masque de subdivision qui fait de ce type de courbe un modèle à subdivision. Or cette B-spline est une courbe de continuité C^2 possédant des propriétés intéressantes et exploitables. Il existe donc des modèles d'une continuité supérieure à C^0 qui possèdent également les propriétés des courbes à subdivision.

Il apparaît de manière évidente que la continuité C^0 des modèles présentés jusqu'ici pose un certain nombre de problèmes tant au niveau visuel qu'au niveau physique. Pour pallier ce problème, il faut un modèle géométrique 1D qui permet de maîtriser sa continuité. Il suffit de construire ce modèle à partir d'équations mathématiques de continuité désirée : les courbes paramétriques.

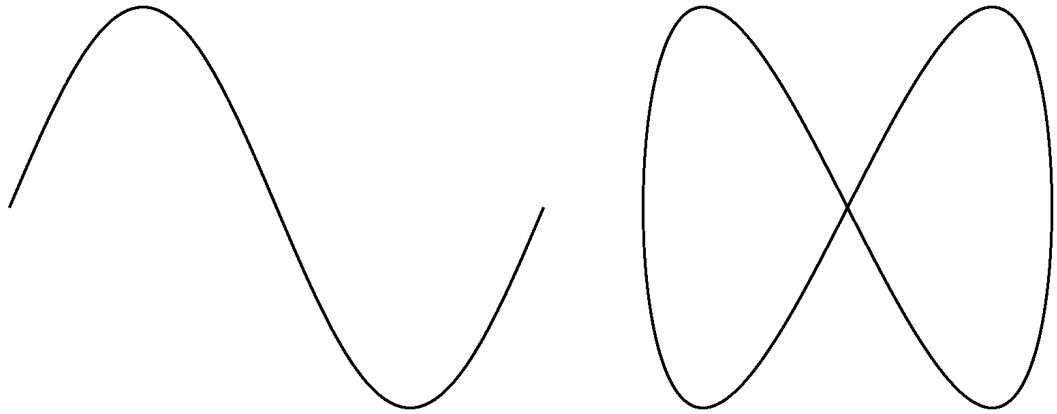
1.2.1.3 Courbe paramétrique

Les courbes paramétriques sont des courbes définies par une abscisse paramétrique et la donnée d'une fonction, de cette abscisse, par axe de l'espace considéré. En 3D, une courbe paramétrique est donc définie par :

$$\mathbf{P}(s) = \begin{pmatrix} x(s) \\ y(s) \\ z(s) \end{pmatrix}$$

avec s l'abscisse paramétrique du point considéré et x, y et z les fonctions de s définissant la courbe dans l'espace 3D. Un exemple est donné par la figure (1.11).

Les propriétés du modèle sont ainsi liées aux propriétés des fonctions $x(s)$,$y(s)$ et $z(s)$. Ainsi, le modèle géométrique obtenu aura pour continuité mathématique la plus petite continuité parmi les fonctions $x(s)$,$y(s)$ et $z(s)$. La courbe peut également être fermée si les fonctions paramétriques utilisées sont toutes périodiques de périodes égales ou multiples comme indiqué sur le schéma (b) de la figure (1.11).



(a) Exemple d'une courbe paramétrique $x(t) = t$ et $y(t) = sin(t)$

(b) Exemple d'une courbe paramétrique fermée $x(t) = sin(t)$ et $y(t) = sin(2t)$. Période commune de 2π

FIG. 1.11 – Exemple de courbe paramétrique en 2D

Ce modèle très théorique reste abstrait et ne permet pas, par exemple, d'avoir une idée de l'allure de la courbe rien qu'avec les paramètres géométriques (i.e. les fonctions $x(s)$,$y(s)$ et $z(s)$).

Une manière de s'approcher d'une solution est de définir la courbe paramétrique à l'aide de points ponctuels de l'espace pondérés par des fonctions paramétriques à valeurs réelles.

1.2.1.4 Spline

Une courbe spline de l'espace $\mathbb{R}^3$ est une courbe paramétrique définie à l'aide de $n+1$ points ponctuels $\mathbf{q_i}$ de l'espace $\mathbb{R}^3$ (appelés points de contrôle de la courbe) pondérés par des fonctions d'influence b_i : $[0,1] \longrightarrow \mathbb{R}$ également appelées fonctions de base :

$$\forall s \in [0,1] \qquad \mathbf{P}(s) = \sum_{i=0}^{n} \mathbf{q_i} b_i(s) \tag{1.16}$$

L'aspect fermé du domaine de définition de la spline n'a d'autre signification que de montrer que la courbe spline est finie. Si le domaine de définition de la courbe est différent, il est toujours possible de le ramener à cet ensemble $[0,1]$ sans perte de généralité (tout intervalle $[a,b]$ se ramène à $[0,1]$ par une transformation affine).

Les fonctions d'influence étant sollicitées régulièrement, elles sont généralement *simples*. Pour cela, les splines utilisées en modélisation géométrique définissent leurs fonctions d'influence comme des polynômes, par morceaux ou non, rationnels ou non. Ainsi, leur évaluation s'en trouve moins coûteuse.

Il peut être noté qu'une courbe spline n'est définie que par la donnée de ses fonctions d'influence. Une classification des splines peut donc être établie, comme le propose Blanc[Bla94], suivant les propriétés vérifiées par les fonctions b_i :

a - Normalité :

Définition : Une courbe spline vérifie la propriété de normalité (elle est alors qualifiée de spline normale) lorsque :

$$\forall s \in [0,1] \qquad \sum_{i=0}^{n} b_i(s) = 1$$

Cette propriété induit simplement le fait que tout point (d'abscisse paramétrique s) de la spline est obtenu par un barycentre des points de contrôle, où les poids sont donnés par les valeurs des fonctions d'influence $b_i(s)$.

Propriétés : De fait, une spline normale possède les deux propriétés suivantes :
- invariance affine : toute transformation affine de la spline s'obtient en appliquant la transformation affine uniquement sur les points de contrôle.
- invariance barycentrique : la composée barycentrique de plusieurs splines normales est obtenue par la composée barycentrique de leurs points de contrôle.

b - Interpolation/Approximation :

Définition : Une spline possède la propriété d'interpolation pour un point de contrôle $\mathbf{q_k}$ lorsque :

$$\exists s_k \in [0,1] \quad / \quad \begin{cases} b_k(s_k) = 1 \\ \forall k' \neq k \qquad b_{k'}(s_k) = 0 \end{cases}$$

Ainsi, le point de la courbe spline d'abscisse paramétrique s_k est $\mathbf{P}(s_k) = \sum_{i=0}^{n} \mathbf{q_i} b_i(s_k) = \mathbf{q_k}$. Donc la spline passe par le point de contrôle $\mathbf{q_k}$.

Définition : Lorsqu'une courbe spline vérifie cette propriété pour tous ses points de contrôle, on parlera d'une spline d'interpolation. Dans les autres cas, on parlera de spline d'approximation.

c - Positivité :

Définition : Une spline possède la propriété de positivité (elle est alors qualifiée de spline positive) si elle vérifie :

$$\forall i \in \{0..n\} \quad \forall s \in [0,1] \quad b_i(s) \geq 0$$

En particulier, une courbe spline normale positive est une courbe obtenue par un barycentre interne au réseau de points de contrôle, ce qui procure la propriété suivante :

Propriété : Toute courbe spline normale positive est entièrement comprise dans l'enveloppe convexe de ses points de contrôle.

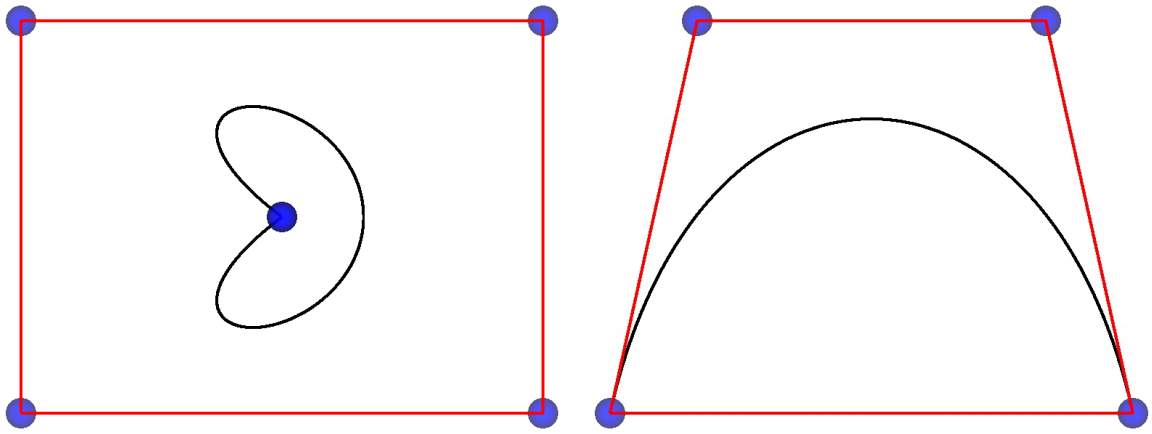


FIG. 1.12 – Exemples d'enveloppe convexe de courbes splines normales positives

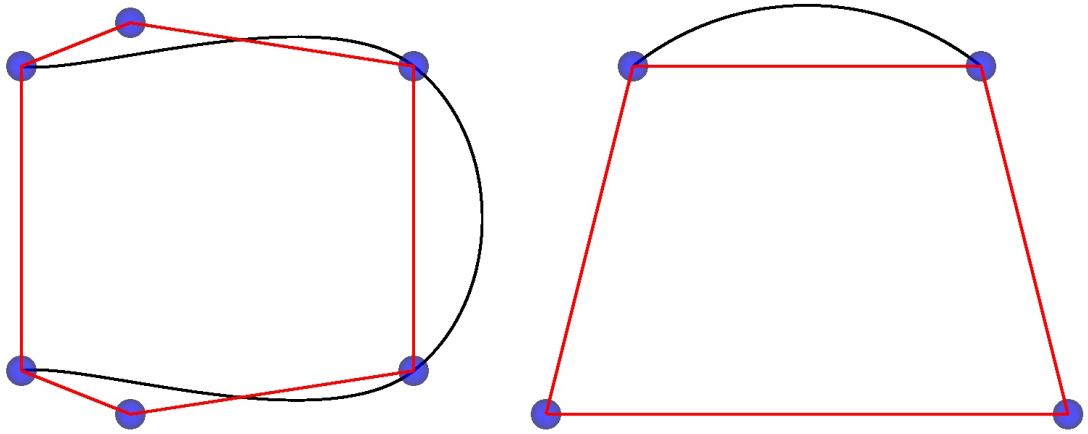


FIG. 1.13 – Exemples d'enveloppe convexe de courbes splines non normales positives

d - Régularité :

Définition : Une spline possède la propriété de régularité (elle est alors qualifiée de spline régulière) si elle vérifie :

$$\forall k \in \{0..n\} \quad \exists s_k \in [0,1] \quad / \quad \left\{ \begin{array}{llll} \forall s < s_k & b'_k(s) \geq 0 & \texttt{et} & \forall k' \in \{k+1..n\} \quad b_{k'}(s) \leq b_k(s) \\ \forall s > s_k & b'_k(s) \leq 0 & \texttt{et} & \forall k' \in \{0..k-1\} \quad b_{k'}(s) \leq b_k(s) \end{array} \right.$$

Autrement dit, toutes les fonctions d'influence $b_k(s)$ possèdent un unique maximum en s_k, et deux fonctions $b_k(s)$ et $b_{k'}(s)$ ne peuvent se croiser qu'entre s_k et $s_{k'}$. Gerald Farin[Far92] a démontré la propriété suivante :

Propriétés : régularisation des oscillations
Le nombre d'intersections entre un plan de $\mathbb{R}^3$ et une courbe spline normale, positive et régulière de $\mathbb{R}^3$ est, au plus, égale au nombre d'intersections entre le plan et le réseau de contrôle de la courbe.

Cette propriété de régularisation des oscillations sous-tend qu'une telle courbe ne peut pas effectuer d'oscillations "parasites" dans l'enveloppe convexe de ses points de contrôle. Autrement dit, chaque point de contrôle influe sur la courbe progressivement jusqu'à un maximum puis décroît progressivement. Dans ces conditions, le réseau formé par les points de contrôle constitue une très bonne approximation de l'allure de la courbe.

e - Localité :

Définition : Une spline possède la propriété de localité (elle est alors qualifiée de spline locale) si elle vérifie :

$$\forall k \in \{0..n\} \quad \exists (s_k^-, s_k^+) \in [0,1]^2 \quad / \quad \left\{ \begin{array}{ll} \forall s < s_k^- & b_k(s) = 0 \\ \forall s > s_k^+ & b_k(s) = 0 \end{array} \right.$$

Cela signifie que chaque fonction $b_k(s)$ n'influence la courbe que localement sur l'intervalle $\Delta_k = [s_k^-, s_k^+]$. Cette propriété est particulièrement intéressante puisque, d'un point de vue modélisation géométrique, elle permet de localiser les modifications de la courbe suite au déplacement d'un point de contrôle et donc de réduire la complexité en temps de cette modification de $\mathcal{O}(n)$ à $\mathcal{O}(1)$. D'un point de vue physique, cette propriété permet de localiser l'ensemble des calculs effectués sur un modèle spline. Si une force est excercée sur une spline locale, non seulement l'équation spline nous permet de distribuer correctement la force sur les points de contrôle, mais de plus, la localité nous indique que seul un certain nombre de ses points sont affectés.

Comme le fait remarquer Blanc[Bla94], les fonctions d'influence d'une spline locale ne peuvent pas être définies par un polynôme unique, puisque celui-ci aurait une infinité de zéro. Habituellement,

les fonctions d'influence sont des polynômes construits par morceaux :

$$\forall k \in \{0..n\} \quad b_k(s) = \begin{cases} \pi_k(s) & \texttt{si} \quad s \in \Delta_k \\ 0 & \texttt{sinon} \end{cases}$$

où $\pi_k(s)$ est un polynôme non nul éventuellement défini par morceaux. Ainsi, la courbe spline est intrinsèquement définie par morceaux

Blanc[Bla94] démontre :

- que les suites $(s_0^-, s_1^-, ..., s_n^-)$ et $(s_0^+, s_1^+, ..., s_n^+)$ sont croissantes pour une spline locale régulière.
- qu'une courbe spline normale, positive, régulière et locale est comprise entièrement dans l'union des enveloppes convexes de chacun de ses segments.

Il est à noter que le nombre maximum de fonctions $b_k(s)$ simultanément non nulles est également le nombre maximum de segments influencés par un point $\mathbf{q_k}$. Ainsi, plus ce nombre sera faible, plus l'influence des points $\mathbf{q_k}$ sera localisée sur la courbe. Ce qui permet de définir la notion suivante :

> **Définition :** Une courbe spline possède une localité d'ordre m lorsqu'un point de contrôle influence au maximum m segments de courbe.

L'ensemble des courbes splines détaillées par la suite se définiront par rapport à cette classification. De plus, comme une spline est une combinaison linéaire de points de contrôle par des fonctions polynômiales (par morceaux ou non, rationelles ou non), l'expression d'un point de la spline peut se mettre sous forme matricielle :

$$\mathbf{P}(s) = \begin{pmatrix} s^n & s^{n-1} & \dots & 1 \end{pmatrix} . M . \mathbf{G}$$

Avec M la matrice de transformation caractéristique du type de spline et $\mathbf{G}$ le vecteur de la géométrie qui définit la spline (i.e. généralement le vecteur des points de contrôle). Les propriétés sur les dérivées successives de la courbe peuvent être déduites plus facilement à l'aide de cette expression matricielle. Dans un souci de lisibilité, on notera $\mathbf{T}$ le vecteur $\begin{pmatrix} s^n & s^{n-1} & \dots & 1 \end{pmatrix}$ généralement, $\mathbf{G}$ sera le vecteur des points de contrôle $\begin{pmatrix} \mathbf{q_0} & \mathbf{q_1} & \dots & \mathbf{q_n} \end{pmatrix}^T$, ainsi la spline est définie par l'équation $\mathbf{P}(s) = \mathbf{T}.M.\mathbf{G}$.

Néanmoins, il reste encore la question de la continuité de la courbe spline. Si celle-ci est définie à l'aide de fonctions d'influence polynômiales, elle est alors C^∞. Si, par contre, les fonctions d'influence sont des polynômes par morceaux, la courbe est C^∞ sur chaque segment spline, mais la continuité aux jonctions des segments dépend des polynôme et donc de la spline.

En géométrie, on définit une continuité géométrique d'ordre n (notée G^n) d'une courbe $\mathbf{P}(s)$ lorsque sa dérivée $d^n\mathbf{P}(s)/dc^n$ est continue, où c est l'abscisse curviligne de la courbe. Autrement dit, une continuité G^1 impose simplement que les tangentes droite et gauche d'une jonction soient colinéaires sans pour autant être de même norme.

Il est possible d'utiliser un produit tensoriel de spline pour définir une nouvelle spline mais dans un espace supérieur. Ainsi, une spline en dimension 2 ou 3 peut être définie. De cette manière, une spline de dimension dim, définie par ses $(n_1 \times .. \times n_{dim})$ points de contrôle $\mathbf{q_{i_1,..,i_{dim}}}$ est donnée par l'équation :

$$\mathbf{P}(s_1, .., s_{dim}) = \sum_{i_1=0}^{n_1} .. \sum_{i_{dim}=0}^{n_{dim}} \left(\mathbf{q_{i_1,..,i_{dim}}} \prod_{j=1}^{dim} b_{i_j} \right)$$

Connaissant les propriétés de base des courbes splines, détaillons maintenant les différents types de spline couramment utilisés.

1.2.1.5 Courbe de Bézier

Dans les années 60, Pierre Bézier définit les courbes de même nom [Béz66a, Béz66b, Béz77, Béz86], ces courbes servent alors à la modélisation de carrosserie automobile chez Renault.

Une spline de $n+1$ points de contrôle est définie par ses $n+1$ fonctions d'influence $b_i(s)$. Si le choix des fonctions se porte sur des polynômes, ils doivent être tous indépendants les uns des autres afin de définir de manière unique la courbe spline. Ainsi, ils définissent une base de l'espace vectoriel des polynômes de degrés n. Or, une base simple et évidente de cet espace vectoriel est la base canonique $(1, s, s^2, ..., s^n)$.

Une spline définie dans cette base ne serait pas très exploitable puisqu'elle ne possède aucune propriété particulière.

Une courbe de Bézier est également une spline dont les fonctions d'influence sont des polynômes seulement, la base des polynômes dans laquelle la spline est définie est différente, puisqu'il s'agit de la base des polynômes de Bernstein d'ordre n :

$$B_i^n(s) = C_n^i s^i (1-s)^{n-i} \quad \text{avec } C_n^i = \frac{n!}{i!(n-i)!} \tag{1.17}$$

Une courbe de Bézier de $n+1$ points de contrôle $\mathbf{q_i}$ est alors définie par :

$$\forall s \in [0,1] \qquad \mathbf{P}(s) = \sum_{i=0}^{n} \mathbf{q_i} B_i^n(s) \tag{1.18}$$

Un exemple de courbe de Bézier est donné sur la figure (1.14) avec ses fonctions d'influence.

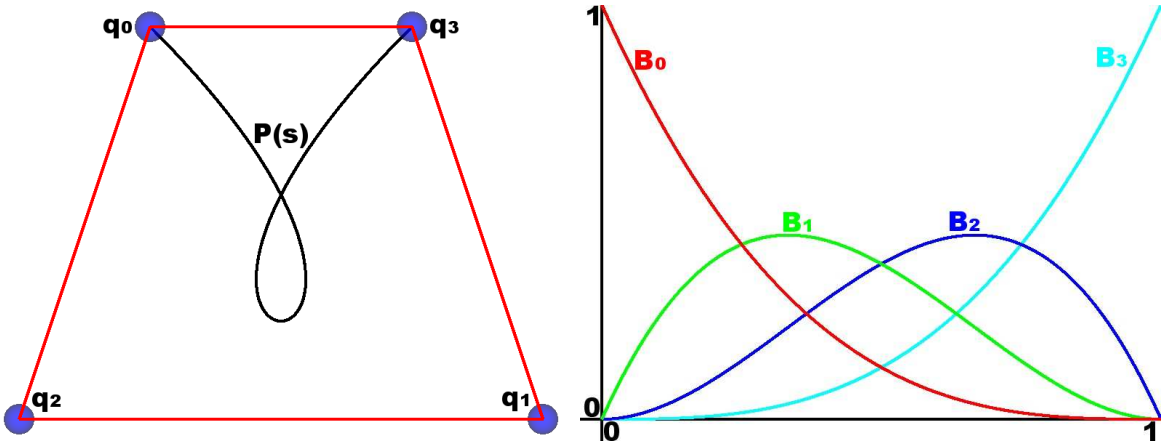


FIG. 1.14 – Exemple d'une courbe de Bézier de degrés 3 (l'enveloppe convexe des points de contrôle est schématisé en rouge) et ses fonctions d'influence.

Une spline de Bézier est normale, positive, régulière, d'interpolation pour les points de contrôle $\mathbf{q_0}$ et $\mathbf{q_n}$ (cf. annexe (B) pour les démonstrations) et de continuité C^∞ puisqu'elle se base sur des polynômes.

Il doit être noté que le degré des polynômes de Bernstein employés correspond directement au nombre de points de contrôle utilisés. Ce qui est une limitation du modèle de Bézier puisque cela introduit des coûts de calcul non négligeables et des problèmes de stabilité numérique lorsque le degré (i.e. le nombre de points de contrôle) augmente. Un autre inconvénient des courbes de Bézier est le manque de localité qui, pour chaque évaluation, impose de considérer tous les points de contrôle, et la moindre modification d'un point de contrôle influe sur toute la courbe. Le manque de localité est marqué par le fait que les fonctions d'influence ont pour support le domaine $[0,1]$ tout entier (cf. figure (1.15)).

La notation matricielle d'une courbe de Bézier dépend du nombre de points de contrôle. Généralement, on trouve dans la littérature des courbes de Bézier de quatre points de contrôle ($n = 3$). Pour une telle courbe, on obtient la matrice suivante :

$$M_3^{Bezier} = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{1.19}$$

De ce fait, la tangente, en l'abscisse paramétrique s, de la courbe de Bézier est définie par l'expression matricielle :

$$\mathbf{T}(s) = \frac{d\mathbf{P}}{ds}(s) = \begin{pmatrix} 3s^2 & 2s & 1 & 0 \end{pmatrix} . M_3^{Bezier} . \mathbf{G}$$

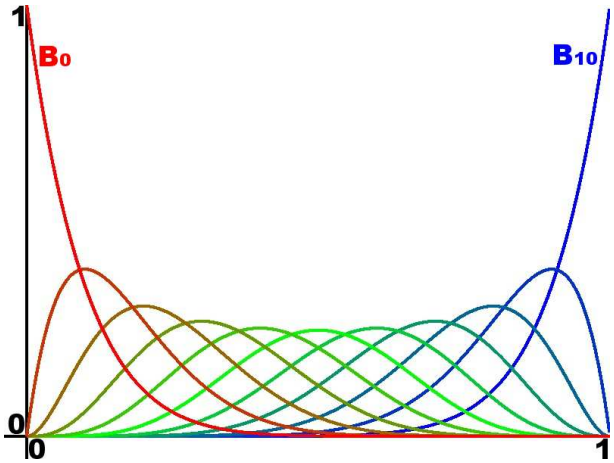


FIG. 1.15 – Polynômes de Bernstein d'ordre 10

On vérifie alors aisément que $\mathbf{T}(0) = 3(\mathbf{q_1} - \mathbf{q_0})$ et que $\mathbf{T}(1) = 3(\mathbf{q_n} - \mathbf{q_{n-1}})$. De manière plus générale, les courbes de Bézier vérifient les relations $\mathbf{T}(0) = n(\mathbf{q_1} - \mathbf{q_0})$ et $\mathbf{T}(1) = n(\mathbf{q_n} - \mathbf{q_{n-1}})$.

L'évaluation de points de la courbe peut être réalisé de manière récursive à l'aide de l'algorithme de De-Casteljau. Cet algorithme tire partie de la propriété des nombres combinatoires : $C_n^i = C_{n-1}^{i-1} + C_{n-1}^i$. Partant de là, De-Casteljau a démontré que l'on pouvait évaluer un point d'une courbe de Bézier par :

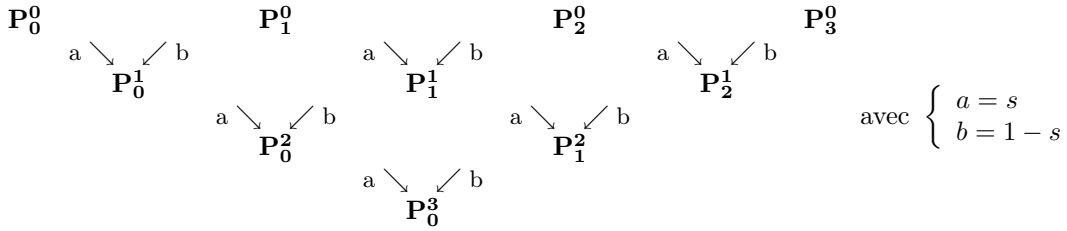
$$\mathbf{P}(s) = \mathbf{P_0^n}(s)$$

où

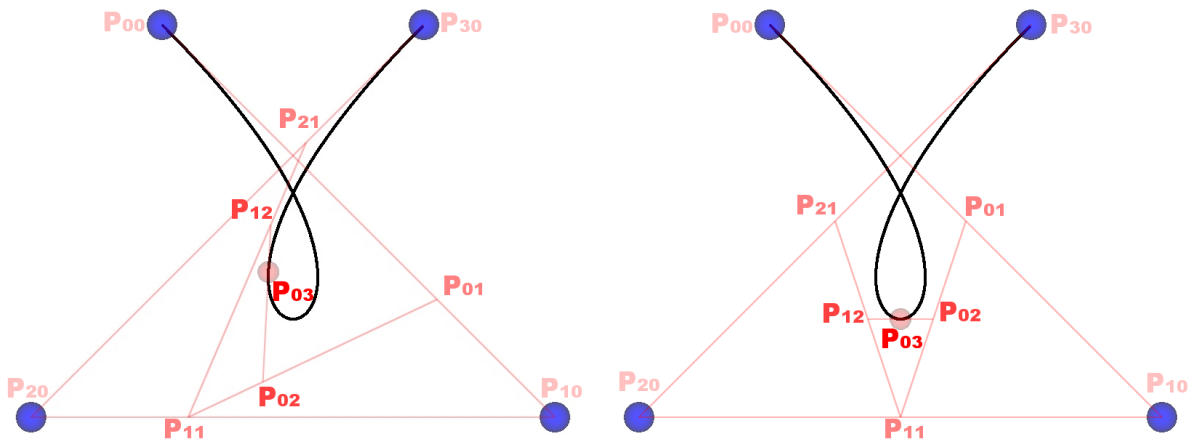
$$\mathbf{P_i^j}(s) = \begin{cases} s\mathbf{P_i^{j-1}}(s) + (1-s)\mathbf{P_{i+1}^{j-1}}(s) & si \quad j > 0 \\ \mathbf{q_i} & sinon \end{cases} \tag{1.20}$$

Cette technique possède l'avantage de ne pas évaluer les puissances qui interviennent dans les polynômes de Bernstein et apporte donc une meilleure stabilité numérique. Géométriquement, cet algorithme récursif calcule des étages de points qui se rapprochent du point recherché comme le montrent les figures (1.16 et 1.17). Dans ces figures, les points calculés sont doublement indicés, le premier indice indique le numéro de point pour l'étape indiquée dans le second indice. Autrement dit, le premier indice correspond à l'indice dans l'équation (1.20) tandis que le second indice correspond à l'exposant dans l'équation (1.20).

Sur la figure (1.16), on peut voir la construction des points intermédiaires de l'algorithme de De-Casteljau pour une spline de Bézier cubique. La relation qui lie les points intermédiaires est souvent notée sous forme pyramidale, comme suit :



Il est possible de raccorder plusieurs splines de Béziers ensemble afin d'obtenir des segments de degré maîtrisé indépendant du nombre total de points de contrôle. Pour cela, il suffit d'imposer aux différentes splines d'avoir un point extrémité en commun. De cette manière, la courbe obtenue est locale d'ordre 2 et de continuité C^0. Il est également possible de poser des conditions sur les points de contrôle pour obtenir une continuité C^1, mais cela supprime des degrés de liberté.



(a) Bézier cubique, évaluation du point d'abscisse paramétrique 0.3.

(b) Bézier cubique, évaluation du point d'abscisse paramétrique 0.5.

FIG. 1.16 – Schéma de calcul d'un point d'une Bézier cubique par l'algorithme de De-Casteljau

1.2.1.6 Courbe de Hermite

Les splines Hermitiennes sont des courbes interpolatrices qui permettent donc de relier un ensemble de points de contrôle. Le degré de la courbe obtenue n'est pas lié au nombre de points de contrôle mais au nombre d'informations que chaque point de contrôle doit fournir. Pour une spline de degré 0, on parlera de spline de Hermite d'ordre 0 et seuls les points de contrôle sont nécessaires puisqu'il s'agit de les relier par des segments de droite. Pour une spline de degré 1, chaque point de contrôle doit être muni de sa tangente, et de la même manière, une spline de degré 2 demande qu'en plus de sa tangente, le point connaisse sa courbure paramétrique.

Pour définir une spline de Hermite d'ordre k, il faut donc simplement spécifier les points de contrôle de la courbe ainsi que leurs k premières dérivées par rapport à l'abscisse paramétrique.

Chaque segment de la spline de Hermite interpole deux points de contrôle consécutifs à l'aide des informations géométriques qu'ils fournissent pour obtenir une continuité C^k aux jonctions. Chaque segment spline fait donc intervenir des polynômes de degré $2*(k+1)$ si k est l'ordre de la spline. La spline de Hermite complète est donc un raccordement de continuité C^k de l'ensemble des segments splines.

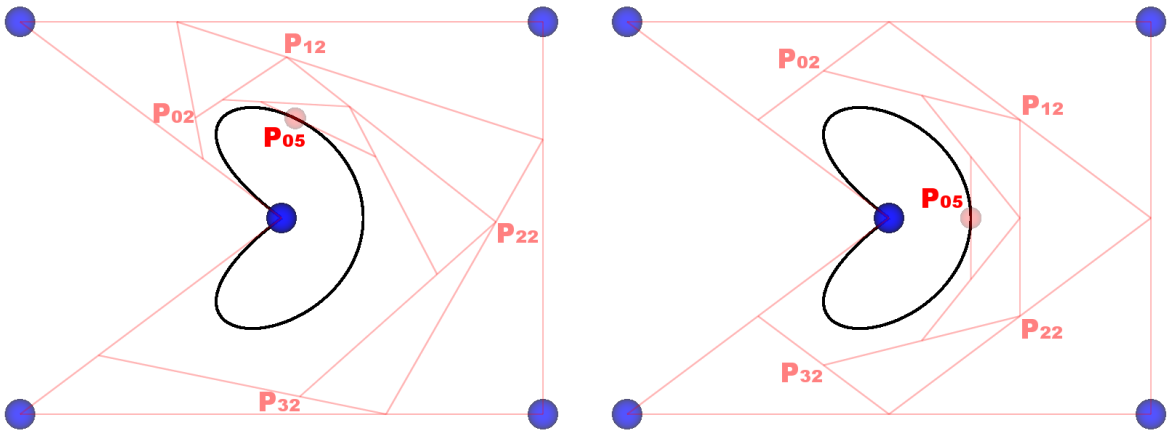
Une courbe de Hermite d'ordre 1 est donc définie par deux points $\mathbf{q_0}$ et $\mathbf{q_1}$ ainsi que par les deux tangentes respectives $\mathbf{t_0}$ et $\mathbf{t_1}$ en ces points. On a donc une spline cubique : $\mathbf{P}(s) = \mathbf{T}.M_3^{Hermite}.\mathbf{G}$ avec $\mathbf{G}$ le vecteur de la géométrie, soit $\mathbf{G} = \begin{pmatrix} \mathbf{q_0} & \mathbf{q_1} & \mathbf{t_0} & \mathbf{t_1} \end{pmatrix}^T$.

Les relations suivantes apparaissent par construction :

$$\begin{aligned}
\mathbf{P}(0) = \mathbf{q_0} &= \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}.M_3^{Hermite}.\mathbf{G} \\
\mathbf{P}(1) = \mathbf{q_1} &= \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}.M_3^{Hermite}.\mathbf{G} \\
\mathbf{P}'(0) = \mathbf{t_0} &= \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}.M_3^{Hermite}.\mathbf{G} \\
\mathbf{P}'(1) = \mathbf{t_0} &= \begin{pmatrix} 3 & 2 & 1 & 0 \end{pmatrix}.M_3^{Hermite}.\mathbf{G}
\end{aligned}$$

d'où la relation

$$\mathbf{G} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix}.M_3^{Hermite}.\mathbf{G}$$



(a) Bézier quintique, évaluation du point d'abscisse paramétrique 0.3.

(b) Bézier quintique, évaluation du point d'abscisse paramétrique 0.5.

FIG. 1.17 – Schéma de calcul d'un point d'une Bézier quintique par l'algorithme de De-Casteljau (seuls les points des étages 2 et 5 de l'algorithme de De-Casteljau sont nommés).

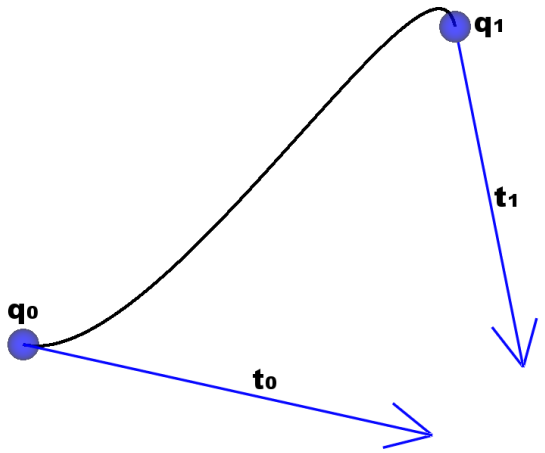


FIG. 1.18 – Spline de Hermite d'ordre 1

et donc

$$M_3^{Hermite} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix}^{-1} = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Il peut être noté que quatre points $\mathbf{q_0}$, $\mathbf{q_1}$, $\mathbf{q_2}$ et $\mathbf{q_3}$, définissent une spline de Bézier et si les vecteurs $\mathbf{t_0} = 3(\mathbf{q_1} - \mathbf{q_0})$ et $\mathbf{t_1} = 3(\mathbf{q_3} - \mathbf{q_2})$ sont définis, ils apportent la relation

$$\begin{pmatrix} \mathbf{q_0} \\ \mathbf{q_1} \\ \mathbf{q_2} \\ \mathbf{q_3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & \frac{1}{3} & 0 \\ 0 & 1 & 0 & \frac{1}{3} \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{q_0} \\ \mathbf{q_3} \\ \mathbf{t_0} \\ \mathbf{t_1} \end{pmatrix} = K. \begin{pmatrix} \mathbf{q_0} \\ \mathbf{q_3} \\ \mathbf{t_0} \\ \mathbf{t_1} \end{pmatrix}$$

Ainsi, la spline de Bézier entre les quatre points $\mathbf{q_i}$ est définie par l'équation (1.19) :

$$\mathbf{P}(s) = \mathbf{T}.M_3^{Bezier}.\begin{pmatrix} \mathbf{q_0} & \mathbf{q_1} & \mathbf{q_2} & \mathbf{q_3} \end{pmatrix}^T = \mathbf{T}.M_3^{Bezier}.K.\begin{pmatrix} \mathbf{q_0} & \mathbf{q_3} & \mathbf{t_0} & \mathbf{t_1} \end{pmatrix}^T$$

La relation $M_3^{Bezier}.K = M_3^{Hermite}$ permet de définir qu'une courbe de Hermite est équivalente à une courbe de Bézier (et réciproquement) avec les constructions adéquates. La courbe de Bézier définit par les quatre points $\mathbf{q_i}$ est équivalente à la courbe de Hermite donnée par les points $\mathbf{q_0}$ et $\mathbf{q_3}$ ainsi que les tangentes respectives $3(\mathbf{q_1} - \mathbf{q_0})$ et $3(\mathbf{q_3} - \mathbf{q_2})$. Inversement, une spline de Hermite définie par les points $\mathbf{q_0}$ et $\mathbf{q_1}$ et les tangentes respectives $\mathbf{t_0}$ et $\mathbf{t_1}$, est équivalente à la courbe de Bézier des points de contrôle $\mathbf{q_0}$, $\mathbf{q_0} + \frac{\mathbf{t_0}}{3}$, $\mathbf{q_3} - \frac{\mathbf{t_1}}{3}$ et $\mathbf{q_1}$.

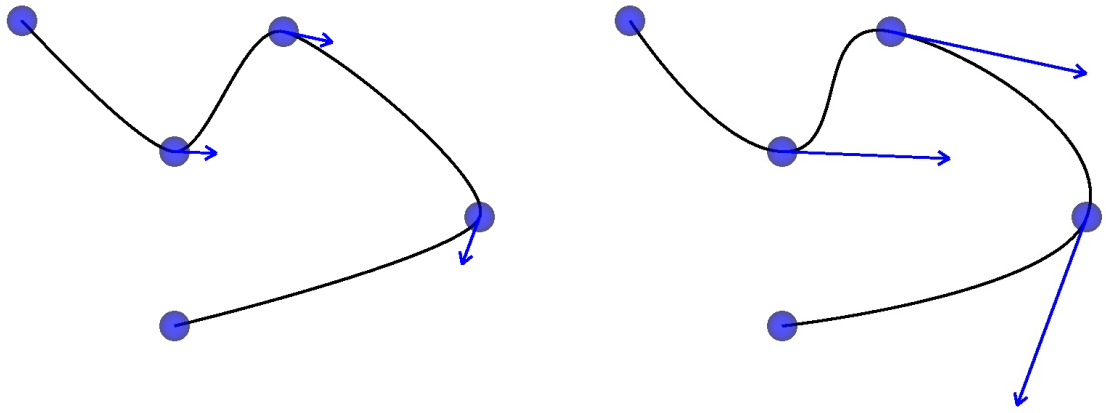
Comme une spline de Hermite d'ordre 1 est équivalente à une spline de Bézier, elle possède les mêmes propriétés intrinsèques, à savoir la normalité et la régularité. Ces propriétés restent vraies pour tous les ordres, de plus les splines de Hermite sont interpolatrices par construction, de localité d'ordre 2 et de continuité C^k.

Cependant, la donnée d'un vecteur tangent en chaque point de contrôle n'est pas un procédé intuitif. Il existe d'autres types de splines d'interpolation plus souple d'utilisation, où les tangentes sont imposées par la configuration des points de contrôle, comme les splines cardinales ou de Catmull-Rom.

1.2.1.7 Spline Cardinale

Une spline cardinale est une spline de Hermite d'ordre 1 où les seules données sont les n points de contrôle $\mathbf{q_i}$. A chaque point de contrôle, une tangente est définie par la relation : $\mathbf{t_i} = \mathbf{q_{i-1}q_{i+1}}$. Dans ce cas, un segment de courbe d'indice i (interpolant les points $\mathbf{q_i}$ et $\mathbf{q_{i+1}}$) dépend, par construction des tangentes, des points $\mathbf{q_{i-1}}$, $\mathbf{q_i}$, $\mathbf{q_{i+1}}$ et $\mathbf{q_{i+2}}$, ce qui augmente la localité de la spline à l'ordre 4. Un scalaire r_i est défini sur la tangente i pour caractériser sa longueur, ce qui offre des degrés de liberté supplémentaires liés aux tangentes qui sont construites automatiquement. Ce paramètre ne fait que moduler la norme de la tangente et ne change pas sa direction, comme indiqué sur la figure (1.19).

Il faut noter que la relation qui permet de construire les tangentes fait apparaître les points $\mathbf{q_{-1}}$ et $\mathbf{q_{n+1}}$. Ces deux points sont appelés les points fantômes de la spline cardinale et peuvent être définis de différentes façons : de manière à prolonger la courbe ($\mathbf{P_{-1}P_0} = \mathbf{P_0P_1}$ et $\mathbf{P_{n-1}P_n} = \mathbf{P_nP_{n+1}}$), pour fermer la courbe ($\mathbf{P_0} = \mathbf{P_n}$, $\mathbf{P_{-1}} = \mathbf{P_{n-1}}$ et $\mathbf{P_{n+1}} = \mathbf{P_1}$), de façon à avoir des tangentes nulles aux extrémités ($\mathbf{P_{-1}} = \mathbf{P_1}$ et $\mathbf{P_{n+1}} = \mathbf{P_{n-1}}$), etc.



(a) r=0.4 (b) r=0.8

FIG. 1.19 – Spline Cardinale (les tangentes sont indiquées par les flèches bleues).

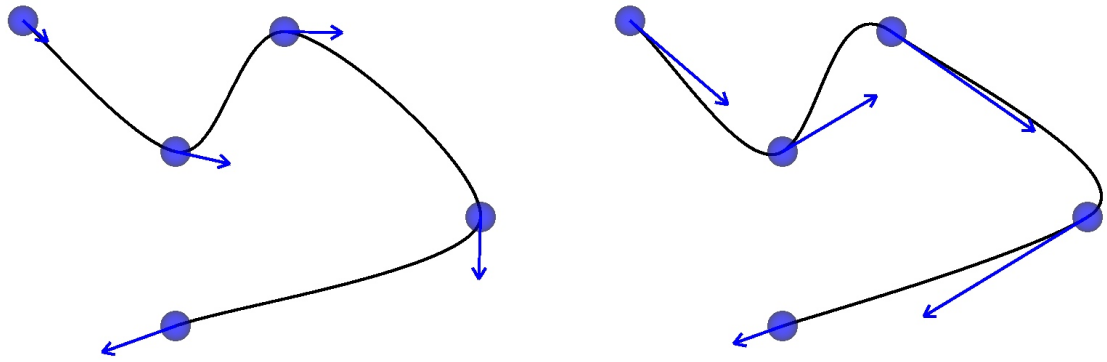
Les splines cardinales ont les mêmes propriétés que les splines de Hermite d'ordre 1 mis à part la localité. Elles sont donc normales, régulières, interpolatrices, de localité d'ordre 4 et de continuité C^1.

1.2.1.8 Courbe de Catmull-Rom

Les splines de Catmull-Rom [CR74] ont la même philosophie que les splines cardinales puisqu'elles se basent également sur les splines de Hermite d'ordre 1. Elles aussi proposent un mécanisme de construction

automatique des tangentes. Ici aussi, un paramètre r permet d'interagir avec la tangente d'un point de contrôle. Cependant, ce paramètre entre en jeu dans la pondération de deux vecteurs et donc induit des directions différentes suivant sa valeur (comme indiqué sur l'exemple figure (1.20)). Voici la construction de la tangente au point de contrôle $\mathbf{q_i}$:

$$\mathbf{t_i} = r_i(\mathbf{q_i q_{i+1}}) + (1 - r_i)(\mathbf{q_{i-1} q_i})$$



(a) r=0.4 (b) r=0.8

FIG. 1.20 – Spline de Catmull-Rom (les tangentes sont indiquées par les flèches bleues).

Un segment de la spline interpolant les points $\mathbf{q_i}$ et $\mathbf{q_{i+1}}$ est donc défini à l'aide des points $\mathbf{q_{i-1}}$, $\mathbf{q_i}$, $\mathbf{q_{i+1}}$ et $\mathbf{q_{i+2}}$. On retrouve donc une localité d'ordre quatre, puisqu'un point de contrôle intervient dans l'évaluation de quatre segments spline.

On retrouve également le fait qu'il existe deux points fantômes aux extrémités de la courbe. Leur construction s'effectue de la même manière que dans le cas des splines cardinales.

Les splines de Catmull-Rom sont donc normales, régulières, interpolatrices, de localité d'ordre quatre et de continuité C^1.

1.2.1.9 B-spline

Les B-splines sont des splines approximatives permettant d'obtenir la continuité d désirée sur tout le modèle, quel que soit le nombre de points de contrôle. Ces splines définissent un ordre k, directement lié au degré d de la courbe par $k = d + 1$, et se basent sur un vecteur de nœuds $\{s_i | i = 0, .., n + k\}$ qui est un ensemble de réels ordonnés de manière croissante. Une B-spline est définie pour toute abscisse paramétrique comprise dans l'intervalle $[s_{k-1}, s_{n+1}[$ (comme précisé dans le paragraphe sur les splines, cet intervalle peut être ramené à l'intervalle $[0, 1]$ par une simple transformation affine) et ses segments sont délimités par l'ensemble des couples de nœuds voisins différents dans cet intervalle. Ces splines définissent leurs fonctions d'influence de manière récursive comme des polynômes par morceaux :

$$N_i^k(s) = \left(\frac{s - s_i}{s_{i+k-1} - s_i}\right) N_i^{k-1}(s) + \left(\frac{s_{i+k} - s}{s_{i+k} - s_{i+1}}\right) N_{i+1}^{k-1}(s) \tag{1.21}$$

et la récursivité prend fin avec les fonctions d'influence d'ordre 1 :

$$N_i^1(s) = \begin{cases} 1 & \text{si} \quad s \in [s_i, s_{i+1}[\\ 0 & \text{sinon} \end{cases} \tag{1.22}$$

Il est à noter qu'il peut arriver que l'un des dénominateurs (ou les deux) intervenant dans l'équation de $N_i^k(s)$ s'annule, dans ce cas, le terme qui le contient prend une valeur nulle. Ce cas arrive lorsque le vecteur de nœuds possède un/des nœud(s) multiple(s).

En 1972, de Boor[dB72] et Cox[Cox72] découvrirent tous les deux en même temps une relation géométrique découlant de la définition analytique des B-splines. Cette relation permet de définir de manière géométrique une B-spline, à la manière de De-Casteljau pour les Bézier :

$$\text{si } s \in [s_l, s_{l+1}[\ , \ \mathbf{P}(s) = \mathbf{P_l^{k-1}}(s)$$

où

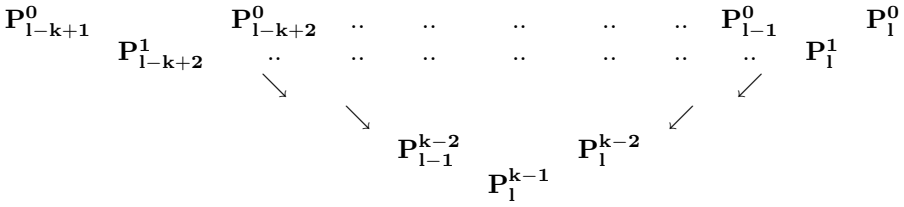
$$\mathbf{P_i^j}(s) = \begin{cases} (1-\tau_i^j)\mathbf{P_{i-1}^{j-1}}(s) + \tau_i^j \mathbf{P_i^{j-1}}(s) & si \quad j > 0 \\ \mathbf{q_i} & si \quad j = 0 \end{cases} \text{, avec } \tau_i^j = \frac{s - s_i}{s_{i+k-j} - s_i}$$

Cette propriété permet de calculer un schéma récursif de calcul de points sans avoir à évaluer les fonctions d'influence, ce qui procure une meilleure stabilité numérique.

De manière plus générale, on peut noter que :

$$\mathbf{P}(s) = \sum_{i=k-1}^{n} \mathbf{P_i^{k-1}} N_i^1(s)$$

De cette manière, on retrouve un algorithme récursif d'évaluation qui, là encore, peut se noter sous forme pyramidale :



Il est à noter qu'il existe une manière de définir une B-spline à l'aide d'un produit de convolution. Cette formule n'est cependant introduite que pour le cas des B-splines uniformes où elle prend une forme plus simple.

De manière générale, les B-splines sont normales, positives, locales d'ordre k et régulières (cf. annexe (C) pour les démonstrations).

Si l'on nomme m_i la multiplicité du nœuds s_i, on peut définir alors qu'une B-spline d'ordre k est de continuité C^{k-2} si tous ses nœuds sont de multiplicité 1. Dès lors qu'un nœud possède une multiplicité plus grande que 1, il diminue d'autant la continuité de la courbe localement. Donc, s'il existe une multiplicité m_i de valeur $k-1$, la courbe est alors localement C^0, ce qui provoque une interpolation du point de contrôle associé à ce nœud de multiplicité $k-1$. Un exemple de B-spline localement interpolatrice est donné en figure (1.21).

Il existe des formes simples de B-spline, appelées B-splines uniformes, qui se basent sur un vecteur de nœuds dit uniforme. Un vecteur de nœuds uniforme est un vecteur de nœuds dont la distance entre tous les couples de nœuds voisins est constante.

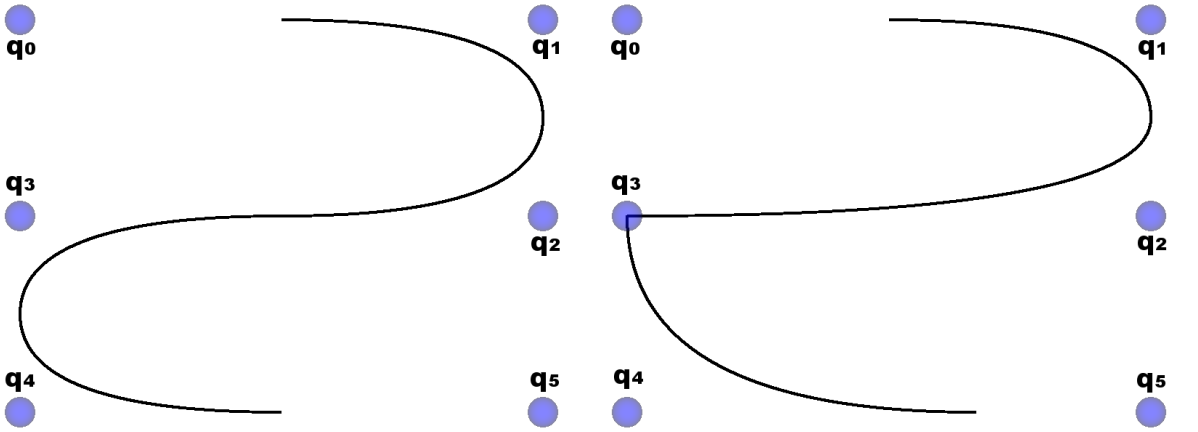
B-spline uniforme :

Les courbes B-splines les plus simples sont les B-splines uniformes. Ces splines se basent simplement sur un vecteur de nœuds uniforme (i.e. $\forall i \in \{1, .., n+k\}$, $s_i - s_{i-1} = 1$), ce qui fixe beaucoup de degrés de liberté du modèle mais permet, en contre-partie, de simplifier l'expression de la spline. Cela offre également la possibilité d'effectuer des précalculs lors de certaine utilisation des splines (par exemple, lors de la simulation physique du modèle géométrique), puisque pour un ordre fixé, les fonctions d'influence sont alors connues. Pour ce type de courbe, aucun point de contrôle n'est interpolé.

Pour ce type particulier de B-splines, on a la relation $N_i^k(s) = N_0^k(s-i)$, ce qui permet de définir les B-splines uniformes par la relation de convolution suivante :

$$N_0^k(s) = (N_0^{k-1} * N_0^1)(s) \tag{1.23}$$

Il est possible de choisir des fonctions d'influence différentes aux segments extrémités pour imposer à la courbe d'interpoler les points extrémités sans diminuer la continuité en ces endroits, comme le montre



(a) Vecteur de nœud $s_i = i$ soit $\{0, 1, 2, 3, 4, 5, 6, 7, 8\}$

(b) Vecteur de nœud $s_i = i$ sauf $t_5 = t_4$ soit $\{0, 1, 2, 3, 4, 4, 6, 7, 8\}$

FIG. 1.21 – Exemple de B-spline d'ordre 3 avec 6 points de contrôle

Blanc[Bla94] en introduisant les B-splines extrémales. Une autre technique consiste à dupliquer les nœuds extrémaux du vecteur de nœud pour réduire la continuité à C^0 aux extrémités et donc imposer à la courbe de passer par les points extrémités (cf. figure (1.22)).

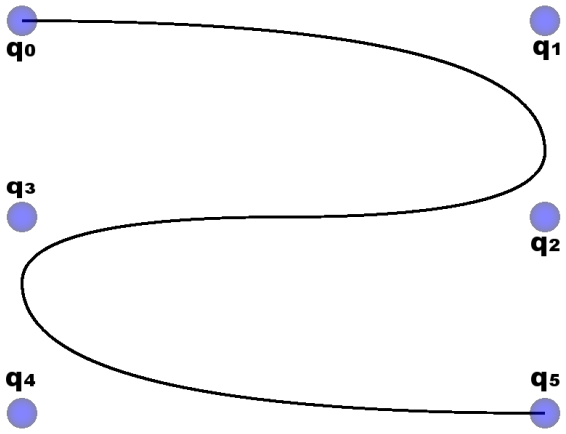


FIG. 1.22 – Exemple d'une B-spline extrémale d'ordre 3 avec 6 points de contrôle. Vecteur de nœud t={0,1,1,3,4,5,7,7,8}

1.2.1.10 B-spline Rationnelle Non-Uniforme

Les B-splines rationnelles non uniformes, ou plus communément appelées NURBS (pour *Non Uniform Rational B-spline*)[PT97], sont une généralisation des B-splines à un espace homogène. En effet, une NURBS est la projection d'une B-spline non uniforme définie dans l'espace homogène $\mathbb{R}^4$. Une NURBS est donc définie par l'expression :

$$\mathbf{P}(s) = \frac{\sum_{i=0}^{n} w_i N_i^k(s) \mathbf{q_i}}{\sum_{i=0}^{n} w_i N_i^k(s)} \tag{1.24}$$

où les $N_i^k(s)$ sont les fonctions d'influence des B-splines et les w_i sont des poids (réels positifs) associés aux points de contrôle $\mathbf{q_i}$. Ainsi donc, la courbe est bien définie dans un espace homogène de $\mathbb{R}^3$ par la donnée des points de contrôle augmentée d'une coordonnée homogène, le poids w_i qui leur est associé.

Les fonctions d'influence des NURBS sont définies, d'après la relation (1.24) par :

$$R_i^k(s) = \frac{w_i N_i^k(s)}{\sum_{j=0}^{n} w_j N_j^k(s)} \tag{1.25}$$

Etant donné la relation qui lie les fonctions d'influence des B-splines à celles des NURBS, les propriétés des fonctions d'influence $R_i^k(s)$ dépendent directement des propriétés des fonctions d'influence $N_i^k(s)$. Comme les poids w_i sont tous positifs, l'équation (1.25) permet de déduire facilement que toutes les propriétés des B-splines sont encore vraies pour les fonctions $R_i^k(s)$. Les NURBS sont donc des splines normales, positives, régulières, locales d'ordre k (le support est $[t_i, t_{i+k}[$).

La continuité de la courbe est définie par l'ordre k de la B-spline et les multiplicités m_i des nœuds t_i du vecteur de nœuds. Sachant que la courbe change de fonctions d'influence en chaque nœud, il en résulte qu'entre deux nœuds, elle est C^∞ puisqu'elle est alors définie par un polynôme. En un nœud t_i de multiplicité m_i, la courbe est de continuité C^{d-m_i} où d est le degré de la courbe, soit $d = k - 1$.

De plus, comme les fonctions d'influence des NURBS sont normales et rationnelles, les courbes NURBS vérifient la propriété d'invariance par projection (parallèle ou perspective) [Bla94, Far92, PT97].

Ce type de courbe est très largement employé dans le domaine de la CAO (*Conception Assisté par Ordinateur*) où la grande richesse du modèle offre aux concepteurs beaucoup de liberté.

1.2.1.11 Les autres types de splines

Des travaux ont permis de définir d'autres types de splines possédant des propriétés supplémentaires.

On trouve par exemple les *Beta-Splines* proposées par Barsky[Bar81] qui introduisent deux nouveaux paramètres β_1 et β_2 intervenant sur l'allure de la courbe (conditions de continuité sur les vecteurs dérivés premier et second).

Blanc[Bla94] propose des splines rationelles de degré 5, les *A-splines*. Ces splines tentent de fournir l'ensemble des propriétés possibles (l'auteur fait remarquer que certaines propriétés sont exclusives et ne peuvent donc pas être réalisées ensemble). Elles procurent une manipulation interactive et une utilisation intuitive, ce qui en fait un outil puissant de modélisation.

Blanc propose également les *X-Splines* [BS95] comme une spline dédiée à l'utilisateur final grâce à ses propriétés interactives et intuitives tirées des *A-Splines*.

Grisoni[GBS99] propose les *HB-Spline* où B-Splines Hermitiennes. Ces splines ont la particularité d'avoir un vecteur de nœuds où chaque nœud possède une multiplicité intrinsèque supérieure à deux. Ainsi, chaque nœud possède un certain nombre de degrés de liberté, ceux-ci étant manipulés à l'aide d'un contrôleur. Ces splines offrent ainsi une grande facilité d'utilisation et de manipulation.

Sederberg et al.[SZBN03] proposent une nouvelle classe de splines nommée *T-Spline*. En 2D, ces splines possèdent la particularité d'avoir une grille de contrôle non forcément quadrillée (au contraire des B-Splines et des NURBS). Ceci permet d'effectuer des raccordements de surfaces T-Splines simplement et avec la continuité souhaitée. Les propriétés des T-Splines procurent également un véritable contrôle local de la surface. Les auteurs proposent également une classe de surfaces raffinables (les T-NURCC) basées sur les T-Splines.

Cette section a permis de détailler quelques modèles géométriques adaptés aux objets 1D. Cette large gamme de modèle géométrique a fait l'objet de diverses études d'animation basée sur la physique, qui ont permis de dégager un certain nombre de modèles mécaniques 1D. Afin de mieux situer notre proposition de modèle 1D, nous proposons d'exposer maintenant quelques uns de ces travaux spécifiques aux modèles mécaniques 1D.

1.2.2 Choix mécaniques pour les modèles 1D

Les possibilités mécaniques d'un objet 1D sont liées au modèle géométrique sous-jacent. Nous proposons donc de classer les modèles mécaniques 1D en fonction de la géométrie employée(cf. section (1.2.1)).

1.2.2.1 Modèles discrets

Cette section n'est pas exclue des généralités de la section (1.1.2.1) détaillée auparavant. Au contraire, elle la complète en présentant des travaux spécifiques au modèle mécanique 1D discret.

Le choix d'un modèle géométrique discret suppose l'utilisation d'un modèle mécanique de type masses-ressorts où les masses sont situées au niveau des points discrets du modèle géométrique. La structure déformable de l'objet est alors réalisée par des ressorts (cf. section (1.1.2.1)).

Casiez [Cas01] propose par exemple de modéliser un fil à l'aide de masses discrètes reliées par des ressorts amortis. Il propose également d'utiliser des ressorts angulaires sur chaque triplet de points consécutifs pour introduire un phénomène de flexion du fil. Casiez souligne que l'animation de textiles peut être réalisée en simulant les fibres à l'aide d'un modèle 1D discret [BHG91, BHG92].

Les modèles 1D discrets sont également utilisés dans l'animation du corps humain. Aubel et Thalmann [AT00] proposent une technique de déformation de la forme du corps humain basée sur une superposition de couches : squelette, muscles, gras, peau. La couche de muscles est réalisée à l'aide d'un habillage polyédrique et de lignes d'action. Ces lignes d'action sont animées par un masses-ressorts amortis 1D et indiquent où doit se placer l'habillage polyédrique des muscles. Dans ce modèle, les masses ponctuelles sont fixées au squelette sous-jacent et, c'est donc le mouvement du squelette qui induit les déformation des muscles, au contraire de la réalité (comme le soulignent les auteurs).

Desbrun et al. [DSB99] proposent une méthode générique pour simuler des objets déformables structurés. Dans cette proposition, le cas des objets 1D est d'abord étudié à l'aide d'un modèle masses-ressorts amortis. Les auteurs réalisent également une étude des méthodes d'intégration les plus adaptées au problème. Dans le cas 1D, les méthodes explicites et implicites sont utilisables. De plus, une extension aux modèles 2D et 3D est proposée où l'on observe des termes non-linéaires dans l'équation des ressorts. Les auteurs proposent alors de décomposer les forces des ressorts en deux parties, une linéaire et l'autre non-linéaire. L'étude de la partie linéaire revient au cas 1D et les auteurs proposent d'intégrer la partie non-linéaire en préservant les moments linéaires et angulaires. Cette proposition a permis de proposer une simulation de tissu interactive [DMB00].

Il est à noter qu'il n'est pas toujours nécessaire de faire appel à la simulation physique pour approcher un comportement réaliste. [Bar97] propose de se baser sur la technique du *key-framing* pour recréer l'animation réaliste d'un modèle 1D sans faire appel à la physique. L'utilisateur positionne simplement l'objet à différents instants de l'animation (ces positionnements forment alors les clés de l'animation). L'animation finale suit alors les positions clés en prenant en compte les déformations de l'objet suivant sa nature. Les déformations sont réalisées par une technique basée sur la cinématique et permettent d'animer un objet précis : corde suspendue, corde déroulée ou enroulée (fil d'un téléphone), ressort. Pour cela, plusieurs types de déformations caractéristiques sont proposés, comme par exemple la propagation d'onde (exemple d'un lasso en mouvement) qui est définie par quatre paramètres (amplitude, fréquence, phase et azimut). Le modèle animé est un modèle discret mais lorsqu'un algorithme a besoin d'une formulation continue, un modèle de type B-spline est automatiquement défini pour interpoler l'ensemble des points discrets. Cependant, le modèle continu n'est pas utilisé dans l'animation, les auteurs donnent simplement un exemple d'utilisation avec le calcul de la longueur du modèle 1D. La B-spline est alors décomposée en un ensemble de petits segments dont les longueurs sont additionnées pour approcher la longueur de la courbe. On peut également imaginer l'utilisation du modèle continu pour une détection robuste des collisions.

Cette technique ne fait qu'approcher la simulation physique et offre donc des résultats moins réalistes mais toutefois suffisants pour certaines applications, notamment pour l'animation d'objets filiformes déformables dans l'industrie du film d'animation (technique utilisée dans le film *Toy Story*).

1.2.2.2 Modèles continus

Le choix d'un modèle géométrique 1D continu ouvre plus de possibilités quant au choix du modèle mécanique. On doit choisir à la fois le formalisme physique, qui permet d'animer la courbe, mais aussi l'énergie de déformation utilisée, qui dicte le comportement du modèle. Nous choisissons ici de détailler quelques travaux classés par choix du formalisme physique. D'abord on trouve les travaux basés sur la mécanique Newtonienne, puis il existe de nombreux travaux réalisés à l'aide du formalisme de Lagrange et enfin, certains travaux explorent de nouvelles possibilités comme l'utilisation de la théorie des Cosserat pour déformer un objet.

Mécanique de Newton :

L'un des travaux pionniers sur la simulation physique en informatique graphique est celui de Terzopoulos et al [TPBF87].

Dans cet article, les auteurs proposent de simuler un objet paramétrique (1D, 2D ou 3D défini par un vecteur $\mathbf{a}$ d'abscisses paramètriques) à l'aide d'une formulation Lagrangienne de l'équation de Newton [TPBF87, GPS01] :

$$\frac{\partial}{\partial t}\left(\mu\frac{\partial \mathbf{r}}{\partial t}\right) = \mathbf{f}(\mathbf{r}, t)$$

où $\mathbf{r}$ est la position de la particule (cette fonction est donc paramétrique et dépendante du temps t : $\mathbf{r}(\mathbf{a}, t)$), $\mu(\mathbf{a})$ est définie localement comme la densité massique de l'objet et $f(\mathbf{a}, t)$ est la contribution des forces extérieures au point d'abscisse(s) $\mathbf{a}$. Le terme de gauche définit donc la variation de quantité de mouvement, principe de base dans la mécanique classique. Comme le terme μ est indépendant du temps, il sort de la dérivation et on obtient le PFD (cf. section (1.1.2.1)) de Newton. Les auteurs proposent également d'ajouter un terme d'amortissement $\gamma\frac{\partial \mathbf{r}}{\partial t}$ où $\gamma(\mathbf{a})$ est la densité de l'amortissement.

Les déformations de l'objet sont définies par une énergie continue $\varepsilon(\mathbf{r})$ qui s'insère dans l'équation dynamique par le biais de sa dérivée variationnelle : $\frac{\delta\varepsilon(\mathbf{r})}{\delta \mathbf{r}}$. Ce terme correspond à l'évaluation discrète de la force conservative locale induite par cette énergie de déformation.

L'énergie de déformation est composée, dans le cas 1D, de trois termes :

– un terme d'élongation calculé à partir du tenseur métrique de l'objet (également appelé *première forme fondamentale*) :

$$G_{ij}(\mathbf{r}(\mathbf{a})) = \frac{\partial \mathbf{r}}{\partial a_i}.\frac{\partial \mathbf{r}}{\partial a_j}$$

Il définit le produit scalaire des tangentes locales à l'objet. Dans le cas 1D, le vecteur $\mathbf{a}$ se réduit à une seule abscisse paramétrique a_1 et donc le tenseur métrique est lui aussi réduit à une valeur scalaire :

$$G_{11}(\mathbf{r}(a_1)) = \left(\frac{\partial \mathbf{r}}{\partial a_1}\right)^2 = s(\mathbf{r}(a_1))$$

qui représente le carré de la norme du vecteur tangente locale et donc informe sur l'élongation locale.

– un terme de flexion, qui est calculé en fonction de la *seconde forme fondamentale* de l'objet :

$$B_{ij}(\mathbf{r}(\mathbf{a})) = \mathbf{n}.\frac{\partial^2 \mathbf{r}}{\partial a_i \partial a_j}$$

où $\mathbf{n}$ est le vecteur unitaire normale à la surface (dans le cas d'une surface). En 1D, ce tenseur se réduit à une valeur scalaire :

$$B_{11}(\mathbf{r}(a_1)) = \mathbf{n}.\frac{\partial^2 \mathbf{r}}{\partial a_1^2} = \kappa(\mathbf{r}(a_1))$$

$\kappa(\mathbf{r}(a_1))$ est donc relatif au vecteur courbure paramétrique locale $\frac{\partial^2 \mathbf{r}}{\partial a_1^2}$. Ce terme informe donc sur la flexion locale de l'objet.

– un terme de torsion dont le calcul n'est pas détaillé par les auteurs. En 1D, ce terme est également un scalaire, il se note $\tau(\mathbf{r}(a_1))$ et il est relatif à la torsion locale de l'objet.

L'énergie de déformation résultante est

$$\varepsilon(\mathbf{r}) = \int_\Omega \alpha(s - s_0)^2 + \beta(\kappa - \kappa_0)^2 + \gamma(\tau - \tau_0)^2 \, da_1$$

où Ω représente le domaine paramétrique de la courbe ; α, β et γ sont les résistances respectives à l'élongation, la flexion et la torsion du matériau. Les termes indicés par zéro sont des termes constants évalués dans la configuration au repos (au temps $t = 0$ par exemple).

Afin d'intégrer cette énergie dans l'équation dynamique et de formuler le système final, les auteurs proposent de découper l'espace paramétrique en une grille discrète (dans le cas 1D, un échantillonnage paramétrique de la courbe) et de définir les équations du mouvement en chaque nœud de cette grille. Ainsi, on peut aisément définir la variation de l'énergie de déformation en chaque nœud à l'aide d'une approximation par différence finie (cf. section (1.1.2.2.1)).

Les équations du mouvement se ré-écrivent alors au niveau des nœuds de la grille par un système d'équations différentielles ordinaires du second ordre :

$$M\frac{\partial^2 \mathbf{r}}{\partial t^2} + C\frac{\partial \mathbf{r}}{\partial t} + K(\mathbf{r})\mathbf{r} = \mathbf{f}$$

où M est la matrice des masses aux nœuds de la grille (matrice diagonale), de même C est la matrice d'amortissement et K est la matrice de rigidité de l'objet (relative aux déformations), $\mathbf{f}$ est le vecteur des forces appliquées aux nœuds de la grille. Il est à noter que $K(\mathbf{r})$ dépend de $\mathbf{r}$ puisque les déformations sont non-linéaires.

Cette proposition part d'une formulation continue du modèle et discrétise l'objet paramétriquement pour en résoudre les équations dynamiques en des points discrets du modèle. Cependant, il est possible de simuler un modèle continu 1D sans le discrétiser spatialement, par exemple en utilisant le formalisme de Lagrange.

Formalisme de Lagrange :

Les bases du formalisme physique de Lagrange ont été développées en section (1.1.2.3.2).

Rémion et al. [RNG99, RNG00] propose de simuler une spline 1D (cf. section (1.2.1)) à l'aide du formalisme de Lagrange. Dans ce formalisme, tout objet doit être défini par un ensemble de coordonnées généralisées qui permettent d'établir entièrement la configuration du modèle. Or, une spline d'un type donné est entièrement définie par ses points de contrôle. Les auteurs proposent donc de définir les coordonnées généralisées comme étant les coordonnées des points de contrôle. Ensuite, ils utilisent l'équation (1.16) de la spline pour déterminer le système d'équations dynamiques du modèle simulé. Ainsi, la simulation porte sur le modèle spline dans son intégralité, donc sur un modèle continu. L'ensemble des propriétés physiques du modèle sont définies de manière continue comme des fonctions paramétriques (masses, frottement visqueux, gravité). Leur contribution au sein du système d'équations est prise en compte sur l'ensemble du modèle spline par une formulation intégrale sur l'abscisse paramétrique. Il est également possible de prendre en considération des forces ponctuelles appliquées à une abscisse paramétrique quelconque. Une extension de ces travaux aux dimensions supérieures a également été proposée [RNN00]. Le modèle de spline dynamique est détaillé dans le chapitre (2).

Les déformations d'un modèle spline 1D animé par les équations de Lagrange peuvent être discrètes (à l'aide de ressorts placés sur le modèle, générant des forces ponctuelles) ou continues [NR01]. L'énergie continue de déformation proposée par Nocent et Rémion [NR01] se base sur le tenseur de Green-Lagrange pour mesurer les déformations non-linéaires et celui de Piola-kircchoff pour imposer les contraintes (cf. annexe (A)). Ainsi, le modèle résultant est un modèle déformable capable de prendre en considération de grands déplacements et ainsi couvrir une plus large gamme de matériau (comme les fils utilisés dans l'élaboration de vêtements tricotés). Nourrit [Nou99] a étudié l'utilisation d'un modèle dynamique de spline de Catmull-Rom (cf. section (1.2.1)) pour définir un

textile à base de mailles.

De même, Terzopoulos et Qin [QT96] proposent d'utiliser le formalisme de Lagrange pour animer une courbe de type NURBS (cf. section (1.2.1)). Le choix d'un modèle NURBS procure plus de degrés de liberté que les autres types de splines puisque les NURBS sont définies dans un espace homogène. En 3D, une NURBS est donc définie par des points de contrôle 3D agrémentés par des poids. On obtient donc une simulation physique basée sur quatre coordonnées généralisées par point de contrôle. Comme il n'est pas toujours nécessaire de posséder autant de degrés de liberté sur une courbe, les auteurs proposent d'utiliser une méthode de contraintes (les multiplicateurs de Lagrange avec un schéma de Baumgarte : cf. section (1.3.4)) pour fixer des degrés de liberté comme certains poids. Cependant, les auteurs soulignent le fait qu'un poids négatif peut poser des problèmes dans la formulation des NURBS et donc faire diverger la matrice du système d'équations dynamiques. Pour éviter ce problème, les auteurs proposent d'assurer la positivité des poids par une méthode projective (cf. section (1.3.2)) (ils étudient également l'utilisation d'une méthode à pénalité (cf. section (1.3.1)), mais ils concluent que la méthode projective fonctionne mieux). Ainsi, après chaque itération de simulation, le système vérifie alors le signe de chaque poids et les signes négatifs sont projetés sur le demi-espace positif. Ils deviennent donc nuls (étant donné l'aspect dynamique des poids, leur vitesse peut également être corrigée pour assurer une stabilité).
La proposition de Terzopoulos et Qin est réalisée dans le but d'offrir un outil puissant de modélisation et de sculpure. En effet, les propriétés géométriques des NURBS en font l'outil de modélisation par excellence. L'aspect dynamique du modèle rend l'utilisation de cet outil intuitive et efficace.

Modèle de Cosserat :

Pai[Pai02] propose de simuler un modèle 1D à l'aide d'une théorie introduite en 1909 par Eugène Cosserat.
Dans la théorie de l'élasticité classique (cf. annexe (A)), l'énergie se base sur les déplacements des points du matériau pour en déterminer une force. Dans celle de Cosserat (plus de détails sont donnés dans [Lak95]), chaque point du matériau possède un déplacement mais aussi une rotation. Ainsi, l'énergie de Cosserat est déterminée à partir d'un couple force/moment local. Lakes[Lak95] fait remarquer que la théorie de l'élasticité classique définit les déformations d'un matériau isotrope à l'aide de deux coefficients (cf. annexe (A)), alors que la théorie de Cosserat s'appuie sur six coefficients caractéristiques. L'utilisation de la théorie de Cosserat permet d'introduire l'élasticité d'un modèle, dans son ensemble. Ainsi, elle définit intrinsèquement l'élongation, la flexion et la torsion de l'objet physique.
[Pai02] propose donc de simuler de manière cinématique une courbe évoluant dans l'espace $\mathbb{R}^3$ en utilisant un repère local pour établir les déplacements et rotations de chaque point du modèle (comme indiqué sur la figure (1.23)). Ainsi, en utilisant ce repère, l'auteur est en mesure d'utiliser l'élasticité de Cosserat. Il fait remarquer que le repère local n'est pas le repère local de Frenet (cf annexe (D)), celui-là possède des informations supplémentaires sur la torsion locale du modèle.
Le modèle de Cosserat permet de déterminer un couple force/moment (respectivement $\mathbf{n}/\mathbf{m}$) en fonction des déformations. Cela correspond au tenseur des déformations dans la théorie de l'élasticité classique. Il faut donc utiliser une loi de constitution pour définir l'énergie de déformation résultante. Pai propose d'utiliser la loi de constitution suivante :

$$\begin{cases} \mathbf{m} &= K(\mathbf{u} - \hat{\mathbf{u}}) \\ \mathbf{n} &= L(\mathbf{v} - \hat{\mathbf{v}}) \end{cases}$$

où K et L sont deux valeurs caractéristiques de l'élasticité du matériau. $\mathbf{u}$ est le vecteur de Darboux (vecteur relatif à la vitesse angulaire), $\mathbf{v}$ est un vecteur relatif à la vitesse d'un point à l'origine. Les vecteurs $\hat{\mathbf{u}}$ et $\hat{\mathbf{v}}$ étant les vecteurs de référence (considérés dans la configuration au repos). (De plus amples informations sont données dans [Pai02]).
Comme la théorie de Cosserat prend en considération les rotations intrinsèques de l'objet, elle permet de considérer de larges déformations globales tout en ayant de petits déplacements en chaque point du matériau. Donc les modifications importantes de la forme de l'objet induisent des effets non-linéaires pris en compte par le modèle d'élasticité de Cosserat.

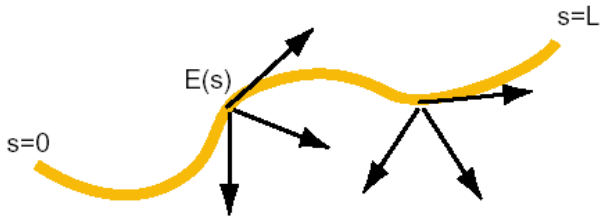


FIG. 1.23 – Repère locaux (E) pour l'utilisation du modèle énergétique de Cosserat. (Extrait de [Pai02]) s représente l'abscisse curviligne. L la longueur de la courbe.

Ce type d'énergie permet notamment de simuler un fil de chirurgie comme l'illustre la figure (1.24).

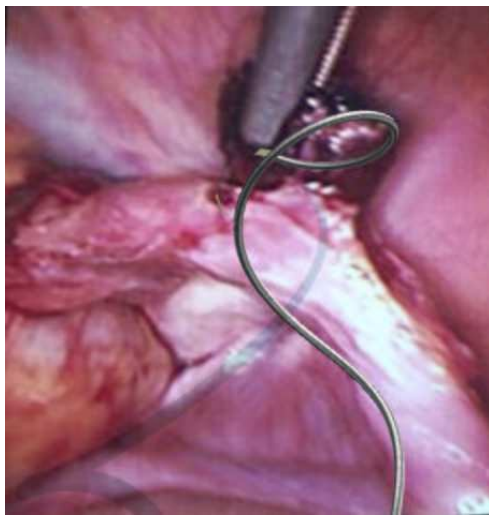


FIG. 1.24 – Utilisation du modèle de Pai dans la simulation d'un fil de chirurgie (Extrait de [Pai02]).

1.3 Contraintes

Les contraintes permettent, de manière générale, d'enrichir un modèle en lui imposant des configurations à respecter. Ces configurations peuvent être de natures différentes : géométriques, physiques ou encore purement mathématiques. Un exemple de contrainte géométrique classique est l'exemple du point fixe : un point donné d'un objet est contraint à rester fixé dans l'espace ou relativement à un autre point. Un exemple de contrainte physique est de fixer les conditions aux limites dans les problèmes de simulation physique basée sur la méthode des éléments finis. Un exemple de contrainte purement mathématique peut s'obtenir en formulant une équation à l'aide des degrés de liberté de l'objet et de leur dérivée première par rapport au temps.

Cependant, un classement des contraintes peut être effectué suivant le type d'équations employées : on distingue les équations ne faisant intervenir que les degrés de liberté de l'objet (contraintes *holonomes*), de celles faisant intervenir à la fois ces degrés de liberté et leur dérivée première par rapport au temps (contraintes *cinématiques*). De plus, on distinguera les contraintes *bilatérales* (données par des équations de contraintes) des contraintes *unilatérales* (données par des inéquations de contraintes).

De manière générale, un système dynamique contraint peut s'écrire sous la forme :

$$\left\{ \begin{array}{ll} \text{minimiser} & f(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) \\ \text{sujette à} & \left\{ \begin{array}{ll} g_i(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) \leq 0 & \forall i \in \{0..m\} \\ h_j(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) = 0 & \forall j \in \{0..l\} \end{array} \right. \end{array} \right. \tag{1.26}$$

avec $\mathbf{q}(t)$ les degrés de liberté du système dynamique, $\dot{\mathbf{q}}(t)$ leur dérivée première par rapport au temps (donc leur vélocité) et t le temps. f est alors appelée la fonction objective, c'est la fonction principale qui doit être minimisée. Cette minimisation est sujette à la vérification des m équations g_i (contraintes unilatérales) et l inéquations h_j (contraintes bilatérales).

Dans tout système mécanique, des objets cohabitent dans un même environnement et sont donc soumis à leur collisions respectives. Ces collisions peuvent être vues également comme des contraintes typiquement unilatérales et holonomes[MW88]. Soit une collision entre deux objets, pour assurer le contact et ainsi apporter une réponse à la collision, on peut introduire la contrainte :

$$g(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) \leq 0$$

avec g représentant l'interpénétration relative des deux objets. En général,la fonction g se base sur la distance d'interpénétration et donc dépend uniquement des coordonnées géométriques $\mathbf{q}$ (et pas des données cinématiques $\dot{\mathbf{q}}$). En d'autres termes, cette contrainte g est de type holonome.

Il existe diverses méthodes permettant d'intégrer des contraintes dans un système dynamique. Nous proposons de détailler quelques une d'entre-elles, pour ensuite les confronter.

1.3.1 Pénalité

Une manière simple d'intégrer des contraintes au sein d'un modèle dynamique est de s'appuyer sur une méthode à pénalité[CA99]. Ces méthodes transforment simplement la minimisation sous contraintes (cf. équation (1.26)) en une minimisation sans contrainte. Pour cela, de nouveaux termes viennent s'agglutiner à la fonction objective pour compenser les équations de contraintes. Ces termes vont donc tendre à faire respecter les contraintes unilatérales et bilatérales :

$$\text{minimiser } f(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) + k.c(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) \tag{1.27}$$

l'ajout du terme c est entièrement et uniquement lié aux contraintes et le facteur k vient juste pondérer les effets dus aux contraintes. La fonction c permet de décliner la méthode en deux catégories (cf la proposition de taxinomie du SRL de gatech [9]) :

méthodes séquentielles Ces méthodes proposent de gérer les contraintes de manière séquentielle au cours des itérations sans jamais assurer leur réalisation. Elle est également connue sous le nom de SUMT (*Sequential Unconstrained Minimization Techniques*) qui est une technique basée sur les travaux de Fiacco et McCormick de 1968 [FM90].

Cette méthode peut se décomposer en deux grandes sous-catégories :

fonctions de pénalité Cette sous-catégorie regroupe l'ensemble des cas où la solution est déterminée en passant par des états où des contraintes ont pu être violées. Exemple :

$$c(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) = \sum_{i=0}^{m} max(0, g_i)^2 + \sum_{j=0}^{l} h_j^2$$

Platt et Barr[PB88b, PB88a] illustrent cette méthode dans le cas de contraintes bilatérales.

fonctions de barrière Cette sous-catégorie regroupe à contrario l'ensemble des cas où la solution est déterminée sans jamais violer la moindre contrainte. Pour cela, les termes ajoutés à la fonction objective sont d'autant plus forts que la solution se rapproche de la limite d'une contrainte. Par exemple, on utilisera les termes suivants :

$$-\sum_{j=0}^{l} h_j^{-1} \text{ ou } -\sum_{i=0}^{m} ln(-g_i)$$

Ces méthodes assurent qu'à chaque instant, les contraintes ne sont pas violées, mais ne sont pas convergentes. L'assurance de la non violation des contraintes passe par une technique d'anticipation.

[9] http ://www.srl.gatech.edu/education/ME6172/Penalty-Barrier.ppt

méthode exacte Cette méthode consiste à construire la fonction c qui permettra de réaliser l'ensemble des contraintes. Malheureusement la plupart des fonctions de pénalité exacte sont non différentiable, ce qui impose l'utilisation de méthodes particulières pour résoudre le problème[10].

Les méthodes à pénalité traitent donc les contraintes en injectant, auprès de la fonction objective, des termes qui vont tendre à faire respecter les contraintes. Généralement on trouve un terme par contrainte. De cette manière, la contrainte n'est pas garantie mais le système tend toujours à la faire respecter. De plus, cette technique traite le cas des contraintes multiples de manière intrinsèque puisqu'alors la somme des contributions entre en jeu, favorisant la réalisation de l'ensemble des contraintes (même si celles-ci sont impossibles à réaliser).

Par abus de langage, le terme *méthodes à pénalité* est très souvent employé pour désigner les *méthodes séquentielles* à *fonctions de pénalité*. Dans cette sous-catégorie, on remarque que les termes dus aux contraintes sont les carrés de leurs équations. Ils sont donc l'équivalent d'une énergie potentielle. Par exemple, l'équation $\mathbf{x}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) = \mathbf{x_0}$ qui impose que le point $\mathbf{x}$ d'un objet soit toujours égal au point immobile $\mathbf{x_0}$ de l'espace considéré. Ainsi, la contrainte est de type holonome bilatérale et s'écrit sous forme vectorielle $\mathbf{h}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) = \mathbf{x}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) - \mathbf{x_0}$. De cette équation découle le terme ajouté à la fonction objective :

$$k.c(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) = k.\mathbf{h}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t)^2 = k.(\mathbf{x}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) - \mathbf{x_0})^2 \tag{1.28}$$

Ce terme fait apparaître l'énergie d'un ressort entre les deux points $\mathbf{x}$ et $\mathbf{x_0}$ de longueur au repos 0 et de raideur $2k$. Bien entendu, on peut moduler cette énergie en insérant une longueur au repos non nulle ou encore monter le ressort en parallèle avec un amortisseur.

Cependant, un tel modèle possède des défauts. En effet plus la raideur k du ressort (ou le coefficient k de manière générale) est grande, moins la contrainte est violée. Mais, en contre-partie, la force grandit et risque d'injecter trop d'énergie dans le système à minimiser et donc d'apporter un problème raide dit *stiff problem*. A partir de là, les problèmes raides sont extrêmement difficiles à intégrer numériquement : soit l'intégrateur ne réussi pas à déterminer la solution du problème, soit il met un temps assez long pour résoudre le problème de manière itérative. De plus, une telle énergie de contrainte va tendre à faire osciller la solution autour de l'équilibre d'où l'utilisation d'un amortisseur en parallèle au ressort. Cet amortisseur permet d'atténuer cet effet oscillatoire.

1.3.2 Projection

Une autre façon de réaliser des contraintes au cours d'une simulation dynamique est d'utiliser une projection. Cette méthode consiste à simplement réaliser la minimisation de la fonction objective sans prendre en compte les contraintes. Puis, une fois la solution obtenue, projeter cette solution de manière à ce que les diverses contraintes soient vérifiées. Une contrainte définit, à l'aide de son équation, un espace vectoriel qui sera utilisé lors de la projection. Ainsi, la solution est projetée sur les espaces vectoriels des contraintes une à une.

Par exemple, Gascuel[GCG92] propose une technique projective pour simuler des contraintes de corps articulés. Pour cela, les corps de l'articulation établissent leur bilan de forces sans prendre en compte les contraintes dues aux articulations. Puis l'intégration numérique calcule le nouvel état de chacun des corps. Les contraintes sont généralement alors violées et sont donc vérifiées après l'intégration par correction des positions mais aussi des vitesses (pour assurer une dynamique cohérente). Cette correction s'effectue par une projection de la solution, trouvée après intégration numérique, sur l'espace des contraintes. Ainsi, cette technique assure la réalisation des contraintes, dans la mesure où celles-ci sont réalisables toutes ensembles. Si tel n'est pas le cas, l'algorithme itératif détecte qu'une convergence n'est pas possible et s'arrête.

De même, Yu et Chen[YC00] proposent une technique directe pour corriger les violations dans un système multi-objets contraints. Pour cela, après chaque intégration numérique, ils évaluent l'erreur commise sur l'équation de contrainte et corrigent, si besoin est, les coordonnées généralisées à l'aide d'une méthode

[10] http ://roso.epfl.ch/cours/optimisation_B/2002-2003/cours/cours06.ppt

itérative de type Newton-Raphson. Ensuite, ils proposent d'une manière similaire de corriger les vitesses des coordonnées généralisées en considérant l'équation dérivée pour les contraintes holonomes.

Promayon et al.[PBP96] proposent d'utiliser une technique projective pour assurer les déformations d'un objet contraint en déplacement et en volume. Pour cela, ils corrigent les positions des particules (à l'aide d'un vecteur déplacement issu d'un calcul projectif) après une étape d'intégration numérique et ré-itèrent la boucle d'intégration numérique jusqu'à ce que la solution trouvée soit assez proche de l'ancienne solution (critère de cohérence temporelle). Enfin, les vitesses des particules sont ajustées. Les auteurs appliquent le même procédé pour assurer la conservation du volume d'un objet.

1.3.3 Réduction cinématique

Une façon de résoudre le problème des contraintes est de supprimer la possibilité qu'a un objet de violer les contraintes. Pour cela, un objet simulé dynamiquement est défini mécaniquement à l'aide d'un ensemble fini de degrés de liberté $q_i(t)$ dépendant du temps t. Afin de lui ôter la possibilité de violer une contrainte, on supprime les degrés qui pourraient entraîner la violation d'une ou plusieurs contraintes. Il est alors possible de réadapter le modèle à la nouvelle configuration en changeant les degrés de liberté.

Un exemple simple est le point fixe d'un corps rigide : un corps rigide A est défini à l'aide de son centre de masse $\mathbf{G}$ et de son orientation $\mathbf{\Omega}$. Si la contrainte de point fixe porte sur $\mathbf{G}$, le corps ne peut plus se translater mais simplement pivoter dans toutes les directions autour de son centre de masse. Ainsi, les degrés de liberté définissant l'objet A contraint sont simplement les trois rotations définissant l'orientation de A. On a réduit les six degrés de liberté à trois. Le principe de la réduction cinématique est donc de supprimer les degrés de liberté gênant la réalisation des contraintes.

Cette technique est toutefois bien difficile à appliquer dans la plupart des cas, les contraintes imposent le choix de départ des degrés de liberté qui ne sont alors pas évidents à déterminer.

Cette technique assure que les contraintes soient vérifiées puisque les corps n'ont pas les degrés de liberté nécessaires pour les violer. Elle a donc la bonne propriété de converger mathématiquement vers la solution sans jamais dériver (mis à part les problèmes d'arrondi numérique).

1.3.4 Multiplicateurs de Lagrange

Une méthode également très connue pour traiter des systèmes contraints est la méthode des multiplicateurs de Lagrange. Cette méthode consiste à traiter les contraintes non pas en réduisant le nombre de degrés de liberté comme la méthode de la réduction cinématique mais au contraire en ajoutant des degrés de liberté au système. Chacun de ces nouveaux degrés de liberté aura pour tâche de faire réaliser une contrainte [Bar96]. Cette méthode se base sur le principe des variations, elle est donc faible par rapport au problème traité (recherche d'un minimum local). Le cas des contraintes unilatérales est traité après, une fois celui des contraintes bilatérales exposées. On part donc d'un problème contraint par une seule équation de contrainte bilatérale :

$$\begin{cases} \text{minimiser} & f(\mathbf{q}(t), \dot{\mathbf{q}}(t)) \\ \text{sujette à} & g(\mathbf{q}(t), \dot{\mathbf{q}}(t)) \quad = 0 \end{cases}$$

pour simplifier les équations, on suppose que tous les paramètres $\mathbf{q}(t)$ et $\dot{\mathbf{q}}(t)$ sont indépendants et on les regroupe dans un seul vecteur de paramètres $\mathbf{x}$ (supposé de dimension n). Le système se ré-écrit alors en :

$$\begin{cases} \text{minimiser} & f(\mathbf{x}) & \text{avec } \mathbf{x} \in \mathbb{R}^n \\ \text{sujette à} & g(\mathbf{x}) \quad = 0 \end{cases} \tag{1.29}$$

La recherche d'un extremum de la fonction f passe par la détermination d'une valeur nulle de ses variations :

$$\delta f = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \delta x_i = 0 \tag{1.30}$$

Comme les n paramètres δx_i sont linéairement indépendants, cette équation est équivalente au système d'équations :

$$\frac{\partial f}{\partial x_i} = 0 \quad \forall i \in \{1..n\}$$

Si maintenant on considère le système contraint (1.29), on obtient un système de $n+1$ équations à n inconnues. Ainsi, on a plus l'indépendance linéaire des paramètres. Mais, en plus de l'équation (1.30) on a la relation :

$$\sum_{i=1}^{n} \frac{\partial g}{\partial x_i} \delta x_i = 0$$

On peut utiliser cette équation pour substituer les variations d'un paramètre dépendant des autres dans l'équation (1.30) (on suppose ici que ce paramètre est x_n) :

$$\delta x_n = -\frac{1}{\frac{\partial g}{\partial x_n}} \sum_{i=1}^{n-1} \frac{\partial g}{\partial x_i} \delta x_i$$

ainsi, les variations de f sont données par :

$$\delta f = \sum_{i=1}^{n-1} \frac{\partial f}{\partial x_i} \delta x_i - \sum_{i=1}^{n-1} \frac{\partial g}{\partial x_i} \delta x_i \frac{\frac{\partial f}{\partial x_n}}{\frac{\partial g}{\partial x_n}} = 0$$

En posant $\lambda = \dfrac{\partial f}{\partial x_n} / \dfrac{\partial g}{\partial x_n}$, l'équation se réécrit en :

$$\delta f = \sum_{i=1}^{n-1} \frac{\partial f}{\partial x_i} \delta x_i - \sum_{i=1}^{n-1} \frac{\partial g}{\partial x_i} \delta x_i \lambda = \sum_{i=1}^{n-1} \left[\frac{\partial f}{\partial x_i} - \lambda \frac{\partial g}{\partial x_i} \right] \delta x_i = 0$$

avec $n-1$ paramètres δx_i linéairement indépendants, donc l'équation est équivalente au système d'équations suivant :

$$\frac{\partial f}{\partial x_i} - \lambda \frac{\partial g}{\partial x_i} = 0 \quad \forall i \in \{1..n-1\}$$

et, il reste les équations $\lambda = \dfrac{\partial f}{\partial x_n} / \dfrac{\partial g}{\partial x_n}$ et $g(x) = 0$ qui complètent le système d'équations :

$$\left\{ \begin{array}{rcl} \dfrac{\partial f}{\partial x_i} - \lambda \dfrac{\partial g}{\partial x_i} & = & 0 \quad \forall i \in \{1..n\} \\ g(\mathbf{x}) & = & 0 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{rcl} \boldsymbol{\nabla} f - \lambda \boldsymbol{\nabla} g & = & \mathbf{0} \\ g(\mathbf{x}) & = & 0 \end{array} \right.$$

on obtient ainsi un système de $n+1$ équations à $n+1$ inconnues, la nouvelle inconnue λ étant nommée *multiplicateur de Lagrange*. Ce résultat signifie simplement que la solution du système contraint est réalisée sur la fonction g lorsque les deux fonctions f et g possèdent des tangentes parallèles (comme indiqué sur la figure (1.25)).

Ce principe se généralise pour un ensemble de c équations de contraintes g_j indépendantes les unes des autres (i.e. une contrainte n'est présente qu'une et une seule fois) :

$$\left\{ \begin{array}{rcl} \dfrac{\partial f}{\partial x_i} - \displaystyle\sum_{j=1}^{c} \lambda_j \frac{\partial g_j}{\partial x_i} & = & 0 \quad \forall i \in \{1..n\} \\ g_j(\mathbf{x}) & = & 0 \quad \forall j \in \{1..c\} \end{array} \right.$$

La généralisation de ce principe aux contraintes unilatérales a été mise en place par Kuhn et Tucker[11]. Les contraintes unilatérales sont intégrées dans le système d'équations par un processus itératif qui vérifie,

[11] Eric W. Weisstein. "Kuhn-Tucker Theorem." From MathWorld–A Wolfram Web Resource. http ://mathworld.wolfram.com/Kuhn-TuckerTheorem.html

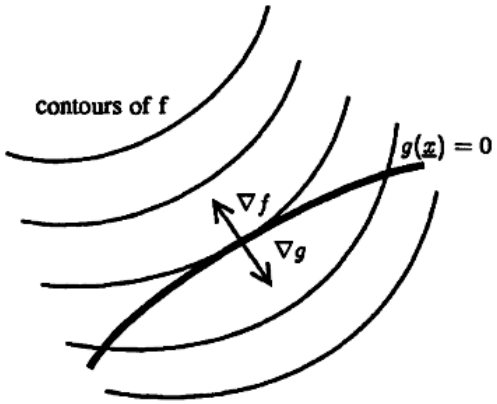


FIG. 1.25 – Schéma explicatif des multiplicateurs de Lagrange (extrait de [PB88a]).

après chaque résolution, l'intégrité des contraintes. On commence la résolution sans aucune contrainte unilatérale, si après une résolution, certaines d'entre elles sont violées, il suffit d'insérer leur équation de contrainte dans le système et de ré-itérer. Le processus s'arrête lorsque toutes les contraintes sont vérifiées.

Olivier Nocent[Noc99] expose la méthode des multiplicateurs de Lagrange appliquée aux équations physiques de Lagrange.

Platt et Barr[PB88b, PB88a] proposent une méthode itérative dérivée des multiplicateurs de Lagrange. Ils démontrent qu'une descente de gradient ne fonctionne pas avec la méthode des multiplicateurs de Lagrange et proposent donc un algorithme légèrement différent (appelé *Basic Differential Multiplier Method* ou BDMM). En posant, $\varepsilon = f(\mathbf{x}) - \lambda g(\mathbf{x})$, il suffit de résoudre le système différentiel suivant :

$$\begin{cases} \dot{x_i} &= \dfrac{\partial \varepsilon}{\partial x_i} &= \dfrac{\partial f}{\partial x_i} - \lambda \dfrac{\partial g}{\partial x_i} \quad \forall i \in \{1..n\} \\ \dot{\lambda} &= \dfrac{\partial \varepsilon}{\partial \lambda} &= -g(\mathbf{x}) \end{cases} \tag{1.31}$$

Les auteurs proposent également d'introduire des contraintes unilatérales en transformant l'inéquation $h(\mathbf{x}) \geq 0$ en équation $g(\mathbf{x}) = h(\mathbf{x}) - \mathbf{x}^2$. Comme $\mathbf{x}^2$ est toujours positif, $h(\mathbf{x})$ doit forcément l'être aussi.

Les auteurs soulignent que leur méthode est totalement compatible avec l'utilisation de méthode à pénalité et proposent pour cela une méthode étendue (appelée *Modified Differential Method of Multipliers* ou MDMM) intégrant des termes de pénalité.

L'application de la méthode des multiplicateurs de Lagrange à un système physique est différente suivant que le système est statique ou dynamique.

Si le système physique est résolu de manière statique, la solution du système est l'ensemble des positions (ou déplacements). L'équation (ou inéquation) d'une contrainte doit donc s'exprimer par rapport à ses positions. Il ne peut donc exister que des contraintes holonomes dans un système statique. De plus, l'équation de contrainte est directement injectée dans le système. De cette manière, la contrainte a l'assurance d'être vérifiée. Un exemple d'utilisation de la méthode des multiplicateurs de Lagrange dans une simulation statique est donné par Cotin et al.[CDA99].

Si le système physique est résolu de manière dynamique, le problème rencontré est tout simplement que les équations de contraintes sont holonomes ou cinématiques donc dépendent des degrés de liberté et de leur dérivé première par rapport au temps. Or, dans un système dynamique, les inconnues sont les accélérations de ces degrés de liberté. Donc, pour former correctement le système matériel, les équations de contraintes vont subir des modifications. Là, plusieurs choix sont possibles :

Différentiation Le choix le plus évident est de simplement différentier l'équation de contrainte par rapport au temps autant de fois que nécessaire pour faire apparaître les accélérations des degrés de

liberté. Si la contrainte est cinématique, une seule différentiation suffit, alors que pour une contrainte de type holonome, il en faudra deux successives [Bar96] :

$$\frac{d^2 c_i^h}{dt^2} \quad \text{et} \quad \frac{dc_j^c}{dt} \tag{1.32}$$

avec c_i^h les i équations de contraintes holonomes et c_j^c les j équations de contraintes cinématiques.
Le fait de différentier les équations de contraintes va introduire une dérive numérique liée à la fois au problème d'arrondi et au principe mathématique que les équations de contraintes portent sur les accélérations des degrés de liberté plutôt que les degrés de liberté eux-même ou leur vitesse (cas cinématique). Donc, toute erreur numérique sur les positions et/ou vitesses sera propagée et jamais corrigée au cours de la simulation puisqu'il n'y a aucune contrainte ni correction sur les positions et les vitesses. Ainsi, cette technique par différentiation temporelle n'assure pas la convergence de la solution. On peut remarquer également que le problème résolu n'est pas le problème posé puisque l'équation de contrainte résolue est différente et non équivalente à l'équation de contrainte posée.

Schéma de Baumgarte Baumgarte [Bau72] propose une technique, pour les contraintes holonomes, qui consiste à mixer les résultats des différentes différentiations en une équation différentielle de contrainte :

$$\frac{d^2 c_i^h}{dt^2} + \alpha_i \frac{dc_i^h}{dt} + \beta_i c_i^h = 0 \tag{1.33}$$

Suivant les choix faits pour les coefficients α_i et β_i, l'équation différentielle va converger ou diverger et donner plus ou moins de stabilité à la réalisation de la contrainte (phénomènes oscillatoires). Ce choix des paramètres est en général assez difficile. De plus, la réalisation exacte de la contrainte n'est pas assurée dans l'immédiat. Si les paramètres α_i et β_i sont correctement choisis, l'état du système mécanique pour un temps tendant vers l'infini est un état stable avec les contraintes réalisées. Cette technique corrige donc la dérive numérique d'une simple différentiation mais possède toujours le même défaut, à savoir que le problème résolu n'est pas le problème posé.
La réalisation de la contrainte est alors soumise à la résolution de l'équation différentielle de second ordre (1.33). Pour cela, on étudie le polynôme caractéristique de l'équation différentielle :

$$x^2 + \alpha_i x + \beta_i = 0 \quad \text{de discriminant} \quad \delta = \alpha_i^2 - 4\beta_i$$

Dans [Asc97], Ascher rappelle que la méthode de Baumgarte impose que les racines de l'équation caractéristique doivent être négatives. C'est pourquoi il souligne que les coefficients de l'équation peuvent être définis par $\alpha_i = 2\gamma$ et $\beta_i = \gamma^2$ avec $\gamma > 0$.
Si δ est négatif, la contrainte possède un régime pseudo-périodique, elle oscille donc autour de la position d'équilibre avec une amplitude de plus en plus faible jusqu'à réaliser la contrainte asymptotiquement.
Si δ est positif, la contrainte possède un régime apériodique qui tend à vérifier la contrainte sans jamais osciller mais en s'en rapprochant de plus en plus.
Enfin, si δ est nul, le régime est alors critique. Ce qui a pour effet de réaliser la contrainte avec un amortissement minimal sans phénomènes oscillatoires.
Le régime procurant la réalisation de la contrainte la plus rapide est par définition même le régime d'amortissement critique.
Le phénomène amorti de la réalisation de la contrainte est dû au terme $\alpha_i x$ avec la condition que $\alpha_i > 0$. Le coefficient α_i est appelé l'amortissement et β_i la pulsation propre du système (ce coefficient est relatif à une fréquence).

Par exemple, le schéma de Baumgarte est utilisé par Barzel et Barr[BB88] pour définir les équations de contrainte dans un système de modélisation basé sur des contraintes dynamiques. Les auteurs proposent d'utiliser l'équation suivante :

$$\frac{d^2 c_i^h}{dt^2} + \frac{2}{\delta_t}\frac{dc_i^h}{dt} + \frac{1}{\delta_t^2} c_i^h = 0$$

où δ_t est la constante de temps définissant la durée de chaque itération de simulation. De cette manière, le régime de la contrainte est amorti critique et permet d'avoir un retour exponentiel à la contrainte contrôlée dans le temps.

1.3.5 Méthodes post-stabilisation

Ces méthodes tentent de stabiliser la contrainte après chaque étape d'intégration numérique. Elles sont distinctes des méthodes projectives par le fait qu'elles n'assurent pas la réalisation de la contrainte mais elles permettent de ne pas introduire d'instabilité dans la dynamique de l'objet (au contraire des méthodes projectives). Ces méthodes trouvent une justification dans le fait que la différentiation des équations de contraintes transforme le problème initial. Le problème initial est sous forme d'une DAE (*Differential Algebraic Equation*) alors qu'une fois transformée, on obtient une ODE (*Ordinary Differential Equation*). Afin de conserver le même problème, il faut introduire des invariants dans le système (voir la thèse de Chin[Chi95] pour plus de détails). Les méthodes post-stabilisation tentent de satisfaire au mieux ces invariants.

Ascher et al.[ACR94] soulignent que le défaut de la méthode de Baumgarte réside dans la difficulté à choisir les coefficients de l'équation. Ils proposent donc d'établir un schéma correctif post-stabilisation. Pour cela, Cline et Pai[Cli02][CP03] décrivent simplement une application possible de la méthode d'Ascher comme suit :
Soit $\mathbf{p}$ un point de l'objet après intégration numérique, soit $\mathbf{g}$ les équations de contraintes à vérifier. Généralement, $\mathbf{g}(\mathbf{p}) \neq \mathbf{0}$. Soit la matrice $G = \frac{\partial \mathbf{g}}{\partial \mathbf{p}}$, la méthode consiste à déterminer un déplacement $\mathbf{dp}$ tel que $\mathbf{g}(\mathbf{p} + \mathbf{dp}) = \mathbf{0}$. En supposant que le déplacement $\mathbf{dp}$ est assez petit, l'équation se linéarise en

$$\mathbf{g}(\mathbf{p} + \mathbf{dp}) \simeq \mathbf{g}(\mathbf{p}) + G(\mathbf{p})\mathbf{dp} \tag{1.34}$$

Donc, le déplacement doit vérifier $G\mathbf{dp} = -\mathbf{g}(\mathbf{p})$. Autrement dit, le terme $G\mathbf{dp}$ doit corriger l'erreur commise sur la contrainte : $\mathbf{g}(\mathbf{p})$. De manière généralle, la matrice G n'est pas carrée, on utilise donc une pseudo inverse (ici à droite) :

$$\mathbf{dp} = -(G^T(GG^T)^{-1})\mathbf{g}(\mathbf{p})$$

sous réserve que GG^T soit inversible.

Cette technique fonctionne bien tant que les déplacements sont relativement petits, dès lors qu'ils deviennent trop importants, la linéarisation (cf. équation 1.34) introduit trop d'erreurs dans le système physique et le système finit par diverger [CP03].

Faure [Fau98] propose un mécanisme de post-stabilisation basé également sur une linéarisation des contraintes géométriques. Il détermine ainsi une correction sur les accélérations et une post-stabilisation pour corriger les vitesses et positions.

Faure propose ensuite un algorithme permettant de contrôler la violation de la contrainte en appliquant des corrections sur les positions. Pour cela, l'algorithme itère jusqu'à ce que l'ensemble des contraintes soit vérifié. Une itération de correction n'est pas forcément suffisante puisque l'algorithme se base sur une forme linéarisée de l'équation de contrainte.

L'auteur utilise une méthode d'intégration numérique permettant d'effectuer des corrections uniquement sur les positions, ce qui permet d'économiser l'étape de post-stabilisation des vitesses.

1.3.6 Discussion

Les diverses méthodes de gestion de contrainte proposées ont toutes leurs spécificités. Il est donc important de bien cerner chacune d'elle pour faire un choix éclairé lorsque l'on doit avoir recours à une méthode de contrainte.

Les méthodes à pénalité sont capables d'intégrer un grand nombre de contraintes dans un même système mécanique sans aucune gestion particulière. En effet, chacun des contraintes vient ajouter son propre terme à la fonction objective. De plus, la méthode à pénalité transforme un système contraint

en un système non contraint ainsi, quelque soit le nombre de contrainte, la résolution reste la même. Cependant, cette classe de méthode ne permet pas d'assurer la réalisation de la contrainte et n'est absolument pas robuste. En pratique, cette méthode est très utilisée pour la gestion des collisions parce qu'elle a les avantages d'être rapide, de n'apporter aucun surcoût à la résolution du système et de gérer automatiquement les systèmes multi-contraint. Cependant elle n'est pas suffisamment robuste dans le cas général d'une contrainte posée explicitement par l'utilisateur.

La méthode à réduction cinématique permet de réaliser exactement les contraintes puisque le modèle n'a plus les degrés de liberté qui permettaient de les violer. Cependant, cette méthode de contrainte est très souvent inutilisable puisqu'elle demande généralement une redéfinition des degrés de liberté d'un modèle. De plus, cette redéfinition est d'autant plus difficile qu'il y a de contraintes. En pratique cette méthode est mise en place sur des systèmes mécaniques très peu contraint et dont la re-définition des degrés de liberté reste anodine (simple suppression d'un degré de liberté...).

Les méthodes de projection ont l'avantage de réaliser exactement les contraintes demandées. Mais elles perturbent généralement le système dynamique, et notament l'intégration numérique, ce qui en fait une méthode instable. Il est cependant possible d'apporter des corrections sur les vitesses pour atténuer ces instabilités. L'un des inconvénients de ces méthodes est la difficulté de prendre en considération de multiple contraintes. En effet, chaque contrainte fait l'objet de modifications sur les degrés de liberté, ainsi des contraintes non indépendantes vis-à-vis des degrés de liberté se perturbent entre elles. Ainsi, la méthode doit opérer des itérations pour ajuster les corrections de chacune des contraintes.

Les multiplicateurs de Lagrange permettent de réaliser exactement un ensemble de contraintes sans aucune itération et sans perturber l'intégration des équations, puisque la prise en compte des contraintes se fait dès l'établissement des équations. Cette méthode de contrainte est donc assez largement utilisée pour assurer des contraintes dans les systèmes statiques mais aussi dynamiques. Le seul problème est que les contraintes exprimées sur un système dynamique doivent être également dynamique. Or, la plupart des contraintes sont géométriques ou cinématiques et les diverses méthodes proposées pour insérer ce type de contraintes ne permettent pas d'assurer la réalisation à chaque instant. Par exemple, en utilisant un schéma de Baumgarte, une contrainte géométrique est transformée en contrainte dynamique permettant ainsi un retour exponentiel à la réalisation de la contrainte. Cependant, l'utilisation d'un tel schéma de Baumgarte permet d'utiliser les multiplicateurs de Lagrange pour gérer au mieux plusieurs contraintes contradictoires. Pour un tel ensemble de contraintes, les méthodes à pénalité permettent de prendre en considération l'ensemble des contraintes, les multiplicateurs de Lagrange munis d'un schéma de Baumgarte également, alors que la réduction cinématique ou les méthodes par projection ne permettent pas de les gérer. La réduction cinématique aboutit à une incohérence du système et les méthodes par projection itèrent à l'infini sans jamais trouver d'état stable où toutes les contraintes sont réalisées puisque cela est impossible par définition.

Enfin, les méthodes post-stabilisation permettent de gérer exactement les contraintes posées, qu'elles soient géométriques ou cinématiques. Cependant, l'utilisation de ces méthodes demandent un investissement intellectuel important, et la plupart des travaux proposés supposent certaines approximations (linéarisation des déplacements) qui rendent la méthode hasardeuse dans le cas de grands déplacements. Ces méthodes sont de plus en plus étudiées dans la littérature et deviennent de plus en plus intéressantes.

1.4 Conclusion

Ce chapitre nous a permis de faire un tour d'horizon des propositions existantes concernant la simulation physique d'objets et plus particulièrement la simulation physique d'objets 1D. Dans ce contexte, il est intéressant de noter qu'il n'y a pas vraiment eu de proposition de modèle 1D en élément fini. La plupart des proposition de modèle 1D sont des modèles discrets à part quelques rares exceptions. Le domaine des modèles déformables 1D continues est encore assez méconnu, c'est pourquoi il semble important d'explorer les possibilités de tels modèles.

Ce chapitre nous a également permis d'exposer divers travaux existants sur les méthodes de gestion de contraintes dans les systèmes dynamiques. De cet exposé, s'en suit une discussion permettant de mieux appréhender les avantages et défauts de chacune d'elle. Il peut être conclu que les méthodes à pénalités sont rapides et permettent de gérer naturellement de multiple contraintes, ainsi cette méthode est souvent

utilisée dans la gestion des collisions. Les méthodes par projections assurent exactement les contraintes, mais elles sont itératives, intolérantes aux contradictions et perturbent l'intégration numérique des équations dynamiques. Les méthodes post-stabilisations sont coûteuses, encore approximatives mais sont très étudiées et devraient proposer une bonne solution au problème des contraintes assez rapidement. Enfin, la méthode des multiplicateurs de Lagrange offre pour le moment le meilleur compromis robustesse, stabilité, tolérance aux incohérences (par l'utilisation d'un schéma de Baumgarte).

Après avoir détaillé divers travaux se rapportant d'une manière ou d'une autre à notre problème, nous décrivons maintenant notre proposition de modèle déformable 1D. Ainsi, nous pouvons nous appuyer sur les travaux existants pour faire des choix éclairés dans l'établissement de notre modèle.

Chapitre 2

Splines dynamiques temps réel

Sommaire

Ce chapitre a pour but de proposer un modèle de spline dynamique temps réel. L'une des conclusions du chapitre précédent est le besoin de modèle 1D continu, aussi nous souhaitons établir un modèle possédant une certaine continuité maîtrisée. Une possibilité pour avoir des propriétés de continuité intéressantes sur le modèle est de commencer par choisir un modèle géométrique possédant les propriétés voulues, puis de se servir de ce modèle comme support de l'animation physique.

Or, les modèles géométriques 1D ont déjà été définis en section (1.2.1).Parmi les choix possibles, les splines ont la propriété de continuité désirée, et de plus elles sont définies par l'intermédiaire de points de contrôle (les degrés de liberté du modèle) qui permettent des modifications simples, efficaces et intuitives. Ces propriétés rendent les splines particulièrement attractives pour le problème qui nous intéresse.

Les deux premières sections de ce chapitre, qui traitent des choix géométriques et physiques faits pour simuler le modèle 1D, se basent sur les travaux de Yannick Rémion et son équipe du LERI de Reims[RNG99, RNG00, RNN00, NR01, NNR01] et de nombreuses formules sont tirées de ces travaux. Notre proposition s'inscrit simplement dans une mise en place temps réel [LMGC02, LMC02] (notamment dans la seconde section qui traite de la simulation physique du modèle). Ainsi, le modèle physique s'appuie sur la géométrie et peut donc tirer partie de l'ensemble des propriétés inhérentes au modèle géométrique choisi. Ensuite, comme notre modèle 1D est déformable, il faut le munir d'énergies de déformation qui permettent de maîtriser son comportement, c'est-à-dire le rendre spécifique à un matériau donné. Il s'en suit une section dont le but est de déployer une large gamme de méthodes permettant de visualiser un modèle 1D. Ensuite, quelques exemples d'applications de cette première version du modèle sont exposés. Enfin, Rémion[RNN00] propose d'étendre la simulation d'une spline 1D aux dimensions supérieures. Nous présentons donc, en dernière section, la mise en œuvre de cette extension du modèle aux dimensions supérieures, en discutant de l'utilisation en temps réel de ces modèles.

2.1 Choix du modèle géométrique

Le choix du modèle géométrique est fortement lié aux types d'objets que l'on souhaite simuler. Or nous souhaitons utiliser notre modèle dans des simulations aussi variées que celles d'un fil de chirurgie, d'un organe filiforme, d'une corde ou encore d'un lacet. Tous ces objets ont la propriété d'être déformable en tout point et doivent potentiellement glisser de manière fluide au travers d'un objet (cas typique d'une suture, d'un lacet...). Ces comportements requiert un modèle de continuité intéressante et maîtrisée. Pour cela, le modèle géométrique qui va soutenir le modèle physique doit, lui aussi, être continu. Parmi les modèles géométriques 1D décrits dans l'état de l'art (en page 30), les modèles géométriques continus les plus riches en propriétés mathématiques sont les courbes de type spline. On y trouve, par exemple, une propriété de localité qui procure, à un niveau physique, des déformations locales pour des interactions locales. Dans la famille des splines, les courbes de Bézier sont écartées à cause de l'absence de localité, on choisira plutôt les splines de Catmull-Rom, mais aussi les B-splines uniformes et non uniformes. Au delà, on trouve les B-splines non uniformes rationnelles (*NURBS*) qui ont des propriétés supplémentaires intéressantes en géométrie projective. Mais, l'utilisation de NURBS se traduit aussi par des manipulations dans un espace tri-dimensionnel homogène donc 4D. Au niveau physique, cela implique à priori que le modèle possède plus de degrés de liberté (un poids est associé à chaque point de contrôle de l'espace tri-dimensionel), et est donc plus lourd à manipuler. Afin de maîtriser certains degrés de liberté (comme les poids des points de contrôle), il faut utiliser une méthode de gestion de contrainte [TQ94], ce qui alourdit le modèle physique. Nous avons donc écarté les NURBS.

Même si nos exemples sont illustrés sur des splines appartenant à l'ensemble $geom = \{$Catmull-Rom, B-spline uniforme cubique, B-spline non uniforme cubique$\}$ qui regroupe des modèles splines très utilisés, les apports du chapitre sont valides pour tous les types de spline possédant une propriété de localité.

Ainsi, quel que soit le type de splines choisi, la définition d'un point de la courbe peut toujours se ramener à la formulation uniformisée suivante :

$$\mathbf{P}(s) = \sum_{i=1}^{n} \mathbf{q_i} b_i(s) \tag{2.1}$$

avec s l'abscisse paramétrique normalisée du point désiré (par normalisation $s \in [0,1]$), $\mathbf{q_i}$ les points de contrôle de la spline, $b_i(s)$ les fonctions de base de la spline et n le nombre de points de contrôle (les

points de contrôle sont des paramètres du modèle mais, pour des raisons de lisibilité, ils ne figurent pas dans la liste des paramétres. Il aurait fallu noter $\mathbf{P}(s)$ comme $\mathbf{P}(s, \mathbf{q})$).

De plus, on a toujours une propriété de localité, plus ou moins élevée suivant le choix mené sur le type de spline. Pour une spline de Catmull-Rom, on trouve une localité de 4. Pour une B-spline, si on note d le degré de la courbe, sa localité est alors de $d + 1$, etc.

Par la suite, on ne spécifie jamais le type de spline utilisé, excepté dans la section des résultats où tous les choix et les paramètres choisis pour chaque test sont donnés. La section qui suit traite de l'animation du modèle géométrique à l'aide d'une simulation dynamique.

2.2 Animation dynamique du modèle géométrique

Afin d'animer physiquement le modèle géométrique préalablement choisi, il nous faut déterminer quel est le formalisme physique le plus adapté à ce modèle géométrique et à notre souhait d'un modèle physique continu (pour cela, nous nous référençons à l'état de l'art sur les modèles déformables vus en section (1.1.2)) :

- Les modèles particulaires de type masses-ressorts permettent d'animer un réseau discret de points massiques en les reliant par des ressorts. L'utilisation de ce type de modèle sur la spline suppose que les masses sont réparties de manière discrète au niveau des points de contrôle et que toute les forces s'appliquent sur ces points. Dans ce cas, le modèle spline n'est absolument pas pris en compte lors de la simulation physique, il n'est qu'un habillage du modèle et n'apporte rien à la simulation physique. Or, le choix du modèle géométrique s'est porté sur une classe d'objets offrant des propriétés intéressantes, dont la continuité précisément pour en tirer parti lors de l'animation. Le choix d'un modèle mécanique particulaire n'est donc pas judicieux puisqu'il nous ferait perdre, au niveau mécanique, tout l'intérêt d'avoir choisi un modèle géométrique continu.
- Les modèles éléments finis (FEM) commencent par énoncer la loi de constitution d'un matériau puis, à l'aide d'une discrétisation spatiale du modèle continu, donnent les équations physiques (statiques ou dynamiques) discrétisées qui permettent de le simuler. Le choix des FEM (ou de toutes méthodes dérivées comme les BEM, VDM...) suppose donc une discrétisation spatiale du modèle continu. Cependant, une spline est définie intrinsèquement par ses points de contrôle, ce qui implique que sa configuration courante soit entièrement définie par la donnée de ces points de contrôle. Donc, si la discrétisation est trop grossière, la spline simulée ne peut pas prendre toute les configurations possibles du modèle géométrique. Et dans le cas d'une discrétisation trop précise, l'objet simulé peut prendre au contraire des configurations extrêmes que le modèle géométrique n'est pas capable d'adopter. Pour avoir une totale correspondance entre le modèle géométrique et le modèle simulé, il faut choisir la discrétisation qui apporte le bon nombre de degrés de liberté. Si l'on veut simuler physiquement le modèle géométrique, le choix d'un modèle éléments finis est assez difficile à mettre en place car il faut déterminer la meilleure discrétisation spatiale possible.
  De plus, un modèle FEM s'appuie sur des fonctions d'interpolations permettant d'évaluer des champs dans/sur tout l'objet (ici 1D). Donc, la méthode des éléments finis n'est pas prévue pour des fonctions d'approximations, ce qui exclu potentiellement l'utilisation de splines d'approximation. Cela laisse supposer que l'utilisation des FEM avec un modèle géométrique 1D de type spline se limiterait aux splines d'interpolation.
- Le formalisme physique de Lagrange permet lui aussi de simuler des objets continus en passant par une discrétisation. Cependant, la discrétisation n'est plus forcément spatiale, elle doit simplement permettre de définir entièrement le modèle. Cette définition procure plus de liberté pour définir un objet et elle permet notamment de définir des fonctions non plus d'interpolation mais d'approximation. Ainsi, les éléments approximés, qui sont les degrés de liberté du modèle, ne sont plus sur l'objet. De ce fait, le modèle continu est discrétisé, mais l'utilisation des fonctions d'interpolation permet de conserver l'aspect continu du modèle lors de la simulation comme pour les modèles éléments finis. L'avantage est ici que la discrétisation est beaucoup plus libre puisqu'elle n'est plus forcément spatiale. Le choix du formalisme physique se porte sur celui de Lagrange pour sa plus grande souplesse dans la discrétisation du modèle continu.

Le formalisme de Lagrange a déjà été développé dans l'état de l'art (1.1.2.3.2). On rappelle ici les définitions de base et les notations employées.

Le formalisme de Lagrange se base sur un ensemble de degrés de liberté notés $\mathbf{q}$, appelés coordonnées généralisées, qui doivent être indépendants les uns des autres et qui définissent entièrement le modèle physique. Comme ces coordonnées généralisées permettent de définir la dynamique du modèle, elles varient au cours du temps, on les note : $\mathbf{q}(t)$.

Un modèle défini par un ensemble de n coordonnées généralisées $\{q_i(t)\}$ peut être animé par le formalisme de Lagrange s'il est capable de définir l'ensemble de ses énergies X par rapport à ses coordonnées généralisées, leur dérivée première par rapport au temps et le temps lui même : $X(\mathbf{q}(t), \dot{\mathbf{q}}(t), t)$ (donc, lorsque l'on parle d'une énergie E, implicitement il s'agit de : $E(\mathbf{q}(t), \dot{\mathbf{q}}(t), t)$). Le modèle est alors animé par les équations de Lagrange (1.14) qui peuvent également prendre en compte des forces non conservatives et alors se mettre sous la forme :

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}_i}\right) + \frac{\partial K}{\partial q_i} = Q_i - \frac{\partial E}{\partial q_i} \quad , \quad \forall i \in \{1..n\} \tag{2.2}$$

avec K l'énergie cinétique du modèle, Q_i la force généralisée, associée à la coordonnée généralisée q_i, des forces ne dérivant pas d'un potentiel et E l'énergie des forces dérivant d'un potentiel (ou énergie potentielle).

Afin d'appliquer le formalisme de Lagrange sur notre modèle géométrique (la courbe spline), il faut d'abord définir les coordonnées généralisées de notre système mécanique. Or, le modèle géométrique est entièrement défini par un ensemble de points de contrôle. Cette définition est unique (i.e. une configuration spatiale donnée est définie, pour un type de spline donné, par un unique ensemble de points de contrôle). Ceci nous permet donc de choisir les coordonnées des points de contrôle comme coordonnées généralisées puisqu'elles définissent entièrement la configuration de la spline. Ces coordonnées généralisées seront notées q_i^α avec $i \in \{1..n\}$ l'indice du point de contrôle et $\alpha \in \{x, y, z\}$ la coordonnée x, y ou z de ce point de contrôle. Comme les coordonnées généralisées sont la base de tout modèle dynamique animé par Lagrange, elles dépendent obligatoirement du temps, ce qui étend la définition du modèle spline de l'équation (2.1) à :

$$\mathbf{P}(s,t) = \sum_{i=1}^{n} \mathbf{q_i}(t) b_i(s) \tag{2.3}$$

en rappelant que les points de contrôle sont également des paramètres de l'équation mais n'y figurent pas pour des raisons de lisibilité (il aurait fallu noter $\mathbf{P}(s,t)$ comme $\mathbf{P}(s, \mathbf{q}(t), t)$).

Ainsi défini, tout point de la courbe géométrique possède maintenant une dynamique et donc une vitesse, donnée par l'équation :

$$\dot{\mathbf{P}}(s,t) = \sum_{i=1}^{n} \dot{\mathbf{q}}_\mathbf{i}(t) b_i(s) \tag{2.4}$$

avec $\dot{\mathbf{q}}_\mathbf{i}$ la dérivée première par rapport au temps du point de contrôle $\mathbf{q_i}$ (i.e. son vecteur vitesse) et en notant que les vitesses des points de contrôle sont également des paramètres de l'équation mais n'y figurent pas pour des raisons de lisibilité (il aurait fallu noter $\dot{\mathbf{P}}(s,t)$ comme $\dot{\mathbf{P}}(s, \mathbf{q}(t), \dot{\mathbf{q}}(t), t)$). Le modèle géométrique ainsi étendu peut être animé par les équations de Lagrange suivantes :

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}_i^\alpha}\right) + \frac{\partial K}{\partial q_i^\alpha} = Q_i^\alpha - \frac{\partial E}{\partial q_i^\alpha} \quad , \quad \forall i \in \{1..n\} \ \texttt{et} \ \forall \alpha \in \{x, y, z\} \tag{2.5}$$

avec K l'énergie cinétique du système, Q_i^α la force généralisée, associée à la coordonnée généralisée q_i^α, des forces ne dérivant pas d'un potentiel et E l'énergie des forces dérivant d'un potentiel.

Nous allons maintenant étudier les équations de Lagrange appliquées à notre modèle (cf. équations (2.5)) en séparant la partie gauche de la partie droite. Puis, nous regrouperons les deux parties au sein d'un système d'équations pour ensuite le résoudre.

2.2.1 Partie gauche des équations de Lagrange

La partie gauche des équations de Lagrange comporte un terme principal qui est K, l'énergie cinétique du modèle. Nous commençons donc par définir cette énergie dans le cas de notre modèle géométrique (cf équations (2.3) et (2.4)). On peut définir l'énergie cinétique en un point de la spline comme étant :

$$K(s, \mathbf{q}(t), \dot{\mathbf{q}}(t), t) = \frac{1}{2}\mu(s,t)\dot{\mathbf{P}}(s,t)^2 \tag{2.6}$$

avec $\dot{\mathbf{P}}(s,t)$ la vitesse du point considéré (cf. équation 2.4) et $\mu(s,t)$ la distribution linéique de masse sur la courbe. Ce paramètre μ peut dépendre, de manière générale, de l'abscisse paramétrique (pour une distribution paramétrique hétérogène de masse) et du temps (pour une distribution dynamique avec des phénomènes de transport de masse). L'énergie cinétique de toute la courbe est alors déterminée par l'intègration de l'énergie cinétique ponctuelle le long de la courbe, qui est définie paramétriquement entre 0 et 1 (cf. équation (2.1)) :

$$K(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) = \int_0^1 \frac{1}{2}\mu(s,t)\dot{\mathbf{P}}(s,t)^2\, ds$$

Or, la masse d'un objet est liée à sa structure. De plus, la structure de la spline est décrite par sa configuration paramétrique (son vecteur de nœuds pour les B-splines). De plus, la structure d'un objet spline ne change pas, ce sont les points de contrôle qui subissent les déformations. Autrement dit, une élongation modifie la longueur curviligne de la courbe mais pas sa longueur paramétrique (ni la structure du vecteur de nœuds). Donc, comme la structure de l'objet ne varie pas au cours du temps, la distribution de masse est elle aussi constante au cours du temps ($\mu(s,t)$ s'écrit $\mu(s)$). Il est à noter qu'ainsi défini, le modèle physique a l'assurance de conserver sa masse quelles que soient les déformations subies.

Cependant, selon la configuration de référence, cette distribution de masse peut varier le long de la structure. Si le modèle de référence possède une transformation linéaire permettant de passer de l'abscisse paramétrique à l'abscisse curviligne, alors la distribution de masse est paramétriquement constante, égale à m. Sinon, la fonction continue $\mu(s)$ doit être déterminée comme le propose [NR01] à l'aide d'un précalcul : soit la densité linéique de masse au repos $\rho_0(s)$ et la configuration de référence indicée par un zéro, on obtient alors la distribution paramétrique de masse :

$$\mu(s) = \rho_0(s)|\mathbf{T_0}(s)| = \rho_0(s)|\frac{d\mathbf{P_0}}{ds}(s)|$$

où $\mathbf{T_0}(s)$ est le vecteur localement tangent (en s) à la courbe de référence.

Sur la figure (2.1), le premier segment paramétrique est deux fois plus lourd que le second.

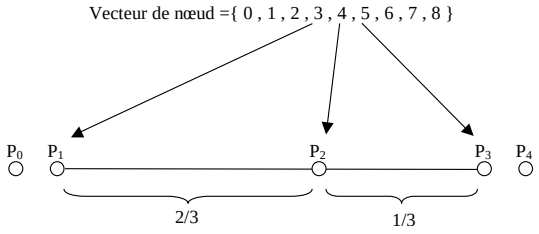


FIG. 2.1 – Schéma de distribution de la masse sur une B-spline uniforme cubique de référence à 5 points de contrôle.

Dans la suite, on suppose que la configuration de référence est toujours rectiligne, homogène (paramétriquement et curvilignement) et donc que la distribution de masse $\mu(s,t)$ est constante égale à m.

En prenant en considération cette remarque, l'énergie cinétique sur le modèle spline devient :

$$K(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) = \frac{1}{2}m\int_0^1 \dot{\mathbf{P}}(s,t)^2\, ds$$

En remplaçant le terme $\dot{\mathbf{P}}(s,t)$ par son expression dans l'équation (2.4), on aboutit à :

$$K(\mathbf{q}(t),\dot{\mathbf{q}}(t),t) = \frac{1}{2}m\int_0^1 \left(\sum_{i=1}^{n} \dot{\mathbf{q}}_\mathbf{i}(t)b_i(s)\right)^2 ds \tag{2.7}$$

Les équations de Lagrange (2.5) font intervenir les termes $\frac{\partial K}{\partial \dot{q}_i^\alpha}$ et $\frac{\partial K}{\partial q_i^\alpha}$. Or l'énergie cinétique ne dépendant que des vitesses des coordonnées généralisées, (cf. équation (2.7)) entraine donc :

$$\frac{\partial K}{\partial q_i^\alpha} = 0 \tag{2.8}$$

et (2.9)

$$\begin{aligned}
\frac{\partial K}{\partial \dot{q}_i^\alpha} &= \frac{1}{2}m\int_0^1 \frac{\partial}{\partial \dot{q}_i^\alpha}\left(\left(\sum_{j=1}^{n} \dot{\mathbf{q}}_\mathbf{j}(t)b_j(s)\right)^2\right) ds \\
&= \frac{1}{2}m\int_0^1 2\frac{\partial}{\partial \dot{q}_i^\alpha}\left(\sum_{j=1}^{n} \dot{\mathbf{q}}_\mathbf{j}(t)b_j(s)\right)\sum_{j=1}^{n} \dot{\mathbf{q}}_\mathbf{j}(t)b_j(s)\, ds \\
&= m\int_0^1 \left(\sum_{j=1}^{n} \frac{\partial \dot{\mathbf{q}}_\mathbf{j}}{\partial \dot{q}_i^\alpha}(t)b_j(s)\right)\left(\sum_{j=1}^{n} \dot{\mathbf{q}}_\mathbf{j}(t)b_j(s)\right) ds \\
&= m\int_0^1 \left(\begin{pmatrix}\delta_{x\alpha}\\ \delta_{y\alpha}\\ \delta_{z\alpha}\end{pmatrix} b_i(s)\right)\left(\sum_{j=1}^{n} \dot{\mathbf{q}}_\mathbf{j}(t)b_j(s)\right) ds \\
&= m\int_0^1 \sum_{j=1}^{n} b_i(s)b_j(s)\dot{q}_j^\alpha(t)\, ds \\
&= m\sum_{j=1}^{n}\int_0^1 b_i(s)b_j(s)\, ds\dot{q}_j^\alpha(t)
\end{aligned} \tag{2.10}$$

avec δ_{ij} le symbole de kronecker :

$$\delta_{ij} = \begin{cases} 1 & \texttt{si i=j} \\ 0 & \texttt{sinon} \end{cases}$$

Ainsi, la partie gauche de l'équation de Lagrange (2.5) s'écrit dans le cas d'une spline 1D dynamique à l'aide des équations (2.8) et (2.10) :

$$\begin{aligned}
\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}_i^\alpha}\right) + \frac{\partial K}{\partial q_i^\alpha} &= \frac{d}{dt}\left(m\sum_{j=1}^{n}\int_0^1 b_i(s)b_j(s)\, ds\dot{q}_j^\alpha(t)\right) \\
&= m\sum_{j=1}^{n}\int_0^1 b_i(s)b_j(s)\, ds\ddot{q}_j^\alpha(t)
\end{aligned} \tag{2.11}$$

Cette équation peut se mettre sous la forme d'un produit matrice-vecteur $\mathcal{M}.\mathbf{A}$ où $\mathcal{M}_{(3n\times 3n)}$ est appelé matrice des masses généralisées et $\mathbf{A}_{(\mathbf{3n})}$ le vecteur des accélérations des coordonnées généralisées.

Propriétés de $\mathcal{M}$ et A :

- On remarque que dans l'équation (2.11), la variation d'énergie cinétique par rapport à un axe α ne fait intervenir que des données relatives à cet axe (i.e. $\ddot{q}_j^\alpha$). On a donc une décorrélation des axes dans les équations, qui peuvent donc être scindées en trois groupes d'équations, chaque groupe ne

faisant intervenir qu'un seul axe. Ceci engendre sur la matrice $\mathcal{M}$ une structure par blocs diagonaux de même dimension $(n \times n)$:

$$\mathcal{M} = \begin{pmatrix} M^x_{(n\times n)} & 0 & 0 \\ 0 & M^y_{(n\times n)} & 0 \\ 0 & 0 & M^z_{(n\times n)} \end{pmatrix}$$

et sur le vecteur $\mathbf{A}$ un arrangement des accélérations des coordonnées généralisées par axe également :

$$\mathbf{A} = \begin{pmatrix} \mathbf{A^x_{(n)}} \\ \mathbf{A^y_{(n)}} \\ \mathbf{A^z_{(n)}} \end{pmatrix}$$

– De plus, on remarque que les coefficients des accélérations ne dépendent pas des axes, ce qui implique qu'ils sont égaux pour les trois axes. Ainsi, les trois matrices M^x, M^y et M^z sont identiques à une matrice de dimension $(n \times n)$ que l'on nomme M :

$$\mathcal{M} = \begin{pmatrix} M_{(n\times n)} & 0 & 0 \\ 0 & M_{(n\times n)} & 0 \\ 0 & 0 & M_{(n\times n)} \end{pmatrix} \tag{2.12}$$

M est définie par les coefficients des accélérations des coordonnées généralisées, données par l'équation (2.11) :

$$M_{ij} = m \int_0^1 b_i(s) b_j(s)\, ds \tag{2.13}$$

– Il est alors à noter que les coefficients de la matrice M font intervenir des produits de fonctions de base de splines. Or, le modèle géométrique choisi offre une propriété de localité (cf. page 34), ce qui signifie qu'une fonction de base de spline n'influence qu'une zone localisée de la courbe. Pour une localité d'ordre l, les fonctions splines s'étendent sur l segments paramétriques de la spline et donc, un point de contrôle n'influence que l segments paramétriques. On obtient donc une matrice bande M de largeur $2(l-1)+1 = 2l-1$. Par exemple, avec une localité d'ordre 4 (exemple d'une B-spline uniforme cubique), on obtient une matrice M avec la structure suivante :

$$M = \begin{pmatrix} X & X & X & X & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .. & 0 \\ . & & & & & & & & & & & & . \\ 0 & .. & 0 & X & X & X & X & X & X & X & 0 & .. & 0 \\ . & & & & & & & & & & & & . \\ 0 & .. & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & X & X & X \end{pmatrix}$$

où X marque simplement l'emplacement d'une donnée non nulle.

– Ensuite, on a la relation suivante :

$$\begin{aligned} M_{ij} &= m \int_0^1 b_i(s) b_j(s)\, ds \\ &= m \int_0^1 b_j(s) b_i(s)\, ds \\ &= M_{ji} \end{aligned}$$

ce qui implique que M est symétrique.

– Enfin, les coefficients qui interviennent dans le calcul de M ne dépendent pas du temps, ce qui a pour effet que la matrice M est constante au cours du temps.

Donc, on peut conclure que la matrice $\mathcal{M}$ est diagonale par blocs identiques égaux à M, avec M symétrique, constante au cours du temps et bande de largeur $2l-1$ (où l est l'ordre de la localité de la spline).

2.2.2 Partie droite des équations de Lagrange

Dans la partie droite des équations de Lagrange (2.5), on trouve toutes les autres énergies du modèle, en dehors de l'énergie cinétique. L'ensemble de ces énergies est alors distribué sur les coordonnées généralisées à l'aide d'un vecteur $\mathbf{B} = \left(\mathbf{B^x_{(n)}}\mathbf{B^y_{(n)}}\mathbf{B^z_{(n)}}\right)^T$ de dimension $3n$.

Nous détaillons dans une première sous-section le cas général où l'on souhaite simplement insérer l'énergie d'une force. Puis, dans une seconde sous-section, nous illustrons ce principe sur des exemples.

2.2.2.1 Cas général

On distingue les énergies qui proviennent des forces dérivant d'un potentiel (on parle alors d'énergies potentielles et de forces conservatives) des autres énergies. De ce fait, l'ensemble des contributions énergétiques se résume à :

$$B_i^\alpha = \sum_{\text{Forces non conservatives}} Q_i^\alpha + \sum_{\text{Forces conservatives}} Q_i^\alpha$$

Force dérivant d'un potentiel Si la force $\mathbf{F}$, appliquée au point $\mathbf{P}(s,t)$ de la courbe, dérive d'un potentiel E, on a alors la relation [12]

$$\mathbf{F} = -\boldsymbol{\nabla} E$$

Le potentiel E est alors appelé énergie potentielle induite par la force $\mathbf{F}$. Une telle force porte alors le qualificatif de conservative. Dans ce cas, l'énergie potentielle E est prise en compte par le formalisme de Lagrange au travers de ses variations par rapport à chaque degré de liberté :

$$Q_i^\alpha = -\frac{\partial E}{\partial q_i^\alpha} \tag{2.14}$$

La contribution, relative à une coordonnée généralisée, d'une force conservative est donc déterminée par l'opposé de sa variation par rapport à cette coordonnée généralisée. Ce terme $-\frac{\partial E}{\partial q_i^\alpha}$ est celui que l'on trouve dans l'équation de Lagrange (2.5).

Cette contribution peut être généralisée pour permettre d'intégrer des forces non conservatives dans le système d'équations. En effet, l'énergie potentielle E peut être exprimée par rapport aux coordonnées généralisées et ainsi s'écrire : $E = E(x(q^x), y(q^y), z(q^z))$ sachant que $x(q^x) = P^x(s,t)$, $y(q_i^y) = P^y(s,t)$ et $z(q_i^z) = P^z(s,t)$ où s correspond à l'abscisse paramétrique du point d'application de la force. On obtient alors :

$$\begin{aligned} Q_i^\alpha &= -\frac{\partial E}{\partial q_i^\alpha} \\ &= -(\frac{\partial E}{\partial x}\frac{\partial x}{\partial q_i^\alpha} + \frac{\partial E}{\partial y}\frac{\partial y}{\partial q_i^\alpha} + \frac{\partial E}{\partial z}\frac{\partial z}{\partial q_i^\alpha}) \\ &= -\boldsymbol{\nabla} E.\frac{\partial \mathbf{P}}{\partial q_i^\alpha}(s,t) \\ &= \mathbf{F}.\frac{\partial \mathbf{P}}{\partial q_i^\alpha}(s,t) \end{aligned}$$

Cette équation est l'expression de la force généralisée associée au degré de liberté q_i^α.

Force ne dérivant pas d'un potentiel Si la force $\mathbf{F}$ appliquée au point $\mathbf{P}(s,t)$ de la courbe ne dérive pas d'un potentiel, sa contribution vient s'ajouter aux équations de Lagrange par le biais de sa force généralisée associée à chaque coordonnée généralisée q_i^α, grâce à la formule montrée ci-dessus :

$$Q_i^\alpha = \mathbf{F}.\frac{\partial \mathbf{P}}{\partial q_i^\alpha}(s,t)$$

[12] $\boldsymbol{\nabla} E = \left(\frac{\partial E}{\partial x} \quad , \quad \frac{\partial E}{\partial y} \quad , \quad \frac{\partial E}{\partial z} \right)^T$ en coordonnées cartésiennes.

En remplaçant $\mathbf{P}(s,t)$ par son expression donnée dans l'équation (2.3), on obtient :

$$\begin{aligned} Q_i^\alpha &= \mathbf{F}.\frac{\partial}{\partial q_i^\alpha}\left(\sum_{j=1}^{n}\mathbf{q_j}(t)b_j(s)\right) \\ &= \mathbf{F}.\left(\sum_{j=1}^{n}\frac{\partial \mathbf{q_j}(t)}{\partial q_i^\alpha}b_j(s)\right) \\ &= \mathbf{F}.\left(\begin{pmatrix}\delta_{x\alpha}\\ \delta_{y\alpha}\\ \delta_{z\alpha}\end{pmatrix}b_i(s)\right) \\ &= F^\alpha b_i(s) \end{aligned} \tag{2.15}$$

Ce terme Q_i^α est celui que l'on trouve dans l'équation de Lagrange (2.5).

Nous allons maintenant détailler quelques exemples de forces usuelles appliquées à notre modèle. La plus évidente de toutes est la force de gravité, ensuite vient s'ajouter la force de frottement due au milieu ambient et enfin, l'ensemble des forces extérieures au modèle dues, cette fois ci, aux interactions du modèle avec son environnement extérieur et à sa cohabitation potentielle avec d'autres modèles.

2.2.2.2 Exemples

Cette sous-section est consacrée à divers exemples d'énergies prise en compte par le modèle. Ces énergies regroupent l'énergie de gravitation, les énergies externes au modèle, et les énergies dues aux frottements visqueux.

La gravité

La première partie de ce paragraphe a pour but de déterminer l'apport de la gravité dans les équations de Lagrange pour un point ponctuel $\mathbf{P}(s,t)$ de la courbe. Ensuite, nous généralisons le principe pour l'appliquer à l'ensemble de notre modèle de manière continue.

Si l'on considère maintenant que la force de gravité est dirigée de manière aléatoire, on peut décomposer la gravité $\vec{g}$ [13] par $\vec{g} = g(-\vec{u})$ où g représente l'intensité de la gravité et $\vec{u}$ un vecteur unitaire représentant la direction opposée de la gravité. De même, la force de gravité appliquée à une particule $P(s,t)$ de masse m est alors donnée par l'expression :

$$\vec{F} = m\vec{g} = mg(-\vec{u}) = -mg\begin{pmatrix}u^x\\ u^y\\ u^z\end{pmatrix} = -\boldsymbol{\nabla}(mg(u^x x + u^y y + u^z z))$$

où x, y et z représentent les coordonnées du point d'application $\mathbf{P}(s,t)$, on obtient alors l'énergie potentielle de gravité :

$$\begin{aligned} E &= mg\left(u^x\sum_{j=1}^{n}q_j^x(t)b_j(s) + u^y\sum_{j=1}^{n}q_j^y(t)b_j(s) + u^z\sum_{j=1}^{n}q_j^z(t)b_j(s)\right) \\ &= mg\sum_{j=1}^{n}b_j(s)\left[u^x q_j^x(t) + u^y q_j^y(t) + u^z q_j^z(t)\right] \end{aligned}$$

Ainsi, la contribution de cette énergie pour une coordonnée généralisée donnée q_i^α est donnée par :

$$\begin{aligned} -\frac{\partial E}{\partial q_i^\alpha} &= -mg\sum_{j=1}^{n}b_j(s)\left[u^x\frac{\partial q_j^x}{\partial q_i^\alpha}(t) + u^y\frac{\partial q_j^y}{\partial q_i^\alpha}(t) + u^z\frac{\partial q_j^z}{\partial q_i^\alpha}(t)\right] \\ &= -mgb_i(s)(u^x\delta_{x\alpha} + u^y\delta_{y\alpha} + u^z\delta_{z\alpha}) \\ &= mg^\alpha b_i(s) \end{aligned}$$

[13] La notation fléchée pour les vecteurs n'est pas employée dans ce document, sauf ici, où elle apporte une certaine lisibilité.

Ceci nous permet d'obtenir l'apport de la force de gravité pour un point ponctuel d'abscisse s. Pour avoir l'apport de la gravité de la courbe toute entière, il suffit d'intégrer ce résultat le long de la courbe :

$$-\frac{\partial E}{\partial q_i^\alpha} = mg^\alpha \int_0^1 b_i(s)\, ds \tag{2.16}$$

Cette intégrale de fonction de base est réalisée en pratique de façon formelle, ce qui permet d'éviter les problèmes d'arrondis et d'erreurs numériques. De plus, on considère que le vecteur gravité $\vec{g}$ peut varier en cours de simulation, les termes $m\int_0^1 b_i(s)\, ds$ de l'équation 2.16 sont donc précalculés.

Il peut être noté que si la gravité est toujours dirigée vers le bas, les termes (2.16) pour $\alpha = x$ et $\alpha = z$ sont tous nuls. Ainsi, seul n termes sur $3n$ sont effectivement calculés.

Frottements visqueux

Les frottements introduisent, dans un système dynamique, des forces qui vont à l'encontre du mouvement et freinent celui-ci. Ainsi, on définit généralement un frottement visqueux par une force proportionnelle à la vitesse, de sens opposé[RNN00] :

$$\mathbf{F} = -C\dot{\mathbf{P}}$$

avec $\dot{\mathbf{P}}$ la vitesse de la particule sur laquelle la force s'exerce et C le coefficient de frottement. Si on applique ce principe à notre modèle 1D, on obtient alors la force suivante : $\mathbf{F} = -C\dot{\mathbf{P}}(s,t)$. Afin de déterminer l'apport de cette force non conservative au sein des équations de Lagrange, on calcule la force généralisée relative à une coordonnée généralisée q_i^α (cf. équation 2.15) :

$$\begin{aligned} Q_i^\alpha(s, \mathbf{q}(t), \dot{\mathbf{q}}(t)) &= F^\alpha b_i(s) \\ &= -C\dot{P}^\alpha(s,t) b_i(s) \end{aligned}$$

En remplaçant la vitesse d'un point de la courbe par son expression (2.4), on obtient :

$$\begin{aligned} Q_i^\alpha(s, \mathbf{q}(t), \dot{\mathbf{q}}(t)) &= -C\left(\sum_{j=1}^{n} \dot{q}_j^\alpha b_j(s)\right) b_i(s) \\ &= -C\sum_{j=1}^{n} b_j(s) b_i(s) \dot{q}_j^\alpha \end{aligned}$$

Pour généraliser le principe à l'ensemble du modèle, on intègre cette force généralisée le long de la courbe :

$$\begin{aligned} Q_i^\alpha(\mathbf{q}(t), \dot{\mathbf{q}}(t)) &= -C\int_0^1 \sum_{j=1}^{n} b_j(s) b_i(s) \dot{q}_j^\alpha\, ds \\ &= -C\sum_{j=1}^{n} \int_0^1 b_j(s) b_i(s)\, ds \dot{q}_j^\alpha \end{aligned} \tag{2.17}$$

On retrouve dans l'équation (2.17) des termes que l'on a déjà rencontrés dans le calcul de la matrice M (cf. équation (2.13)) à un facteur près : $Q_i^\alpha(q(t), \dot{q}(t)) = -\frac{C}{m} M_{ij} \dot{q}_i^\alpha(t)$. On peut donc appliquer le frottement visqueux à toutes les coordonnées généralisées en même temps, en utilisant la matrice $\mathcal{M}$, on obtient alors l'expression : $-\frac{C}{m}\mathcal{M}.\mathbf{V}$ où $\mathbf{V}$ est le vecteur vitesse des coordonnées généralisées.

Les forces extérieures

Notre modèle 1D peut être potentiellement en collision avec d'autres objets présents dans la scène. On suppose qu'un algorithme de détection de collisions existe et qu'il est en mesure de nous apporter une réponse à la collision au travers d'une force et de son point d'application (i.e. l'abscisse paramétrique s

du point de la spline impliqué dans la collision). A ce stade, il nous faut simplement intégrer cette force ponctuelle au sein du système mécanique. Comme nous n'avons aucune information sur la nature de cette force, on utilise la force généralisée associée à chaque coordonnée généralisée via l'équation (2.15). Donc, la contribution de ces forces externes apparaît dans le vecteur $\mathbf{B}$ sous la forme de forces généralisées. De même, les forces dues aux collisions du modèle sur lui-même peuvent être détectées par la plate-forme de simulation de la même manière qu'une collision externe.

2.2.3 Résolution du système d'équations linéaires

Nous avons pu déterminer que les équations de Lagrange appliquées à notre modèle 1D apportent le système d'équations linéaire suivant :

$$\mathcal{M}.\mathbf{A} = \mathbf{B} - \frac{C}{m}\mathcal{M}.\mathbf{V} \tag{2.18}$$

Le fait que le frottement visqueux dû au milieu ambiant fasse intervenir la matrice des masses généralisées nous permet de le prendre en compte après la résolution du système :

$$\begin{aligned} \mathcal{M}.\mathbf{A} &= \mathbf{B} - \frac{C}{m}\mathcal{M}.\mathbf{V} \\ \Leftrightarrow \mathbf{A} &= \mathcal{M}^{-1}.\mathbf{B} - \frac{C}{m}\mathbf{V} \end{aligned}$$

Il suffit donc de résoudre le système linéaire sans prendre en considération le frottement visqueux. Puis de soustraire aux accélérations des coordonnées généralisées ainsi obtenues le vecteur $\frac{C}{m}\mathbf{V}$. Le système linéaire à résoudre est donc réduit à :

$$\mathcal{M}.\mathbf{A} = \mathbf{B}$$

Afin de résoudre au mieux ce système linéaire d'équations, voici un récapitulatif des propriétés qui ont pu être dégagées sur ce système dans l'étude de la partie gauche des équations de Lagrange :

- Décorrélation des axes dans les équations :

$$\mathcal{M}.\mathbf{A} = \begin{pmatrix} M^x & 0 & 0 \\ 0 & M^y & 0 \\ 0 & 0 & M^z \end{pmatrix} . \begin{pmatrix} \mathbf{A^x} \\ \mathbf{A^y} \\ \mathbf{A^z} \end{pmatrix}$$

  Cette propriété nous permet de scinder le système d'équations linéaire de taille $3n$ en trois sous-systèmes d'équations linéaires de taille n. Chacun de ces sous-systèmes s'occupant d'un axe en particulier :

$$\begin{cases} M^x.\mathbf{A^x} = \mathbf{B^x} \\ M^y.\mathbf{A^y} = \mathbf{B^y} \\ M^z.\mathbf{A^z} = \mathbf{B^z} \end{cases}$$

- Facteur des accélérations indépendant des axes :

$$M^x = M^y = M^z = M$$

- Structure bande de la matrice M grâce à la propriété de localité des splines. Pour une spline de localité d'ordre l, la bande de M est de taille $2l-1$.
- La matrice M est symétrique (donc $\mathcal{M}$ également).
- La matrice M est constante au cours du temps.

Comme la matrice M est constante au cours du temps, nous pouvons effectuer des précalculs qui permettent d'accélérer la résolution du système. De plus, la matrice M est bande, or la décomposition LU conserve la propriété bande d'une matrice. Donc une décomposition LU de M donne deux nouvelles matrices L et U, la première triangulaire inférieure, la seconde triangulaire supérieure mais toutes deux sont bandes de largeur $2l-1$ dont l termes nuls à cause de leur propriété triangulaire. Donc la bande

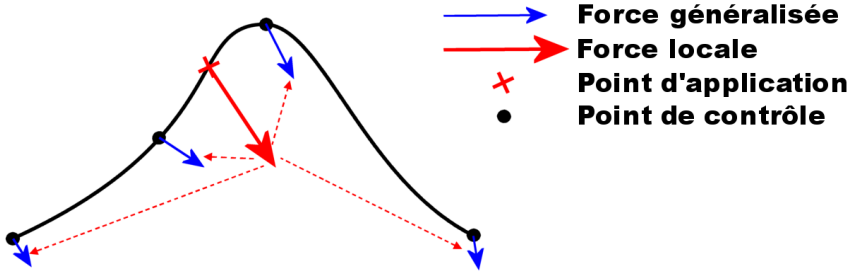
se restreint d'un seul côté de la diagonale et se réduit donc à une largeur de l. Ainsi, la résolution des sous-systèmes s'effectue de la manière suivante :

$$M.\mathbf{A}^{\alpha} = L.U.\mathbf{A}^{\alpha} = \mathbf{B}^{\alpha}$$

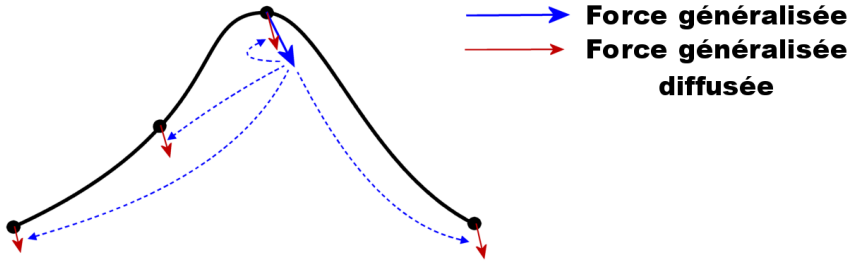
en posant $\mathbf{X}^{\alpha} = U.\mathbf{A}^{\alpha}$, on résoud le système $L.\mathbf{X}^{\alpha} = \mathbf{B}^{\alpha}$. Or, L est triangulaire inférieure, bande de largeur l, cette résolution est donc en $\mathcal{O}(n)$. De même, la seconde partie de la résolution consiste à résoudre : $U.\mathbf{A}^{\alpha} = \mathbf{X}^{\alpha}$. Mais, là encore, U est triangulaire supérieure, bande de largeur l, cette résolution est donc également en $\mathcal{O}(n)$. En ajoutant la complexité de la mise à jour des accélérations après résolution pour le frottement visqueux (mise à jour en temps linéaire également), on obtient une complexité en temps de calcul de la résolution globale en $\mathcal{O}(n)$. La résolution du système d'équations linéaire engendré par le formalisme de Lagrange sur notre modèle 1D est donc linéaire en temps.

Il existe une méthode appelé le *mass lumping* [CDA00], qui consiste à concentrer toute la masse de l'objet sur la diagonale de la matrice des masses. Ainsi, on obtient une matrice M diagonale, ce qui permet de découpler les équations du mouvement en un ensemble indépendant d'équations linéaires. On peut alors assimiler la résolution du système à la résolution d'un simple masses-ressorts. L'application de ce principe au modèle introduit dans ce chapitre apporterait une résolution du système d'équations linéaire en temps, donc la complexité algorithmique ne changerait pas, mais le facteur de complexité serait moindre. Cependant, le fait de concentrer la masse sur les éléments diagonaux de M, n'est pas sans conséquence sur la cohérence physique du résultat.

En effet, la masse est répartie de manière discrète sur les points de contrôle. Et en ce qui concerne la contribution des forces, elle est dans un premier temps distribuée sur les degrés de liberté par le biais des forces généralisées (comme indiqué sur le schéma ((a)) de la figure (2.2)) puis, dans un second temps l'action de chacune de ces contributions est diffusée (lors de la résolution du système) sur les points de contrôle voisins (comme schématisé sur la figure (2.2(b))) à l'aide de l'inverse de la matrice des masses généralisées. L'application du *mass lumping* sur le modèle de spline dynamique consiste donc à ne pas effectuer la diffusion des actions des forces généralisées, et donc à discrétiser les actions.



(a) Distribution d'une force en forces généralisées (dans le calcul des forces généralisées)



(b) Diffusion de l'action d'une force généralisée (lors de la résolution du système d'équations)

FIG. 2.2 – Distribution des forces sur le modèle spline dynamique

Si on simule notre modèle en l'état, les points de contrôle de la spline sont animés indépendamment les uns des autres, leurs mouvements relatifs ne sont donc pas corrélés et peuvent amener des configura-

tions qu'un matériau donné ne pourrait pas emprunter. Afin de fournir au modèle 1D un comportement plausible, il faut le munir d'énergies de déformation qui intégrent, au sein du système, les comportements des matériaux désirés.

2.3 Les énergies de déformation

Notre modèle 1D est un modèle déformable qui peut prendre une infinité de configurations différentes. Le but ici est de lui définir une forme de référence et de quantifier à chaque instant les écarts relatifs à cette forme de référence pour empêcher le modèle de prendre n'importe quelle forme. Afin d'y parvenir, il nous faut définir une énergie de déformation élastique qui a pour tâche de mesurer cet écart et ainsi contrôler la forme. Cette énergie élastique permet donc de lui donner un comportement bien déterminé suivant le matériau désiré. La manière la plus simple d'intégrer des énergies de déformation dans un modèle physique est d'utiliser des énergies discrètes (i.e. contrôler la position d'un nombre fini de points). Mais, pour des modèles physiques continus, des énergies de déformation continues peuvent également être utilisées.

Les énergies de déformation d'un modèle 1D peuvent être classées selon les déformations traitées vis-à-vis d'une configuration de référence (correspondant à la configuration au repos) (définie pour l'exemple par le schéma (a) de la figure 2.4) :

Elongation/compression : Une énergie d'élongation/compression contrôle les déformations longitudinales du modèle. Ainsi, dès que le modèle est étiré ou contracté (comme le montre le schéma (b) de la figure (2.4)), cette énergie va tendre à maintenir l'élongation du modèle proche de celle de la configuration de référence. Sachant que l'élongation locale est déterminée par la norme du vecteur tangent local, on en déduit que l'énergie de déformation en élongation/compression est relative au vecteur localement tangent à la courbe.

Flexion (ou courbure) : Une énergie de flexion tend à faire respecter un invariant de courbure sur un modèle. Pour un modèle 1D, il s'agit de maintenir la courbure courante proche de celle au repos. Si le modèle est continu, cette courbure existe en tout point du modèle, elle est donnée par la norme du vecteur courbure locale comme indiqué sur la figure (2.3). Ce vecteur est continu sur la courbe si et seulement si le modèle géométrique est de continuité au moins C^2. Un exemple de flexion est donné par le schéma (c) de la figure (2.4).

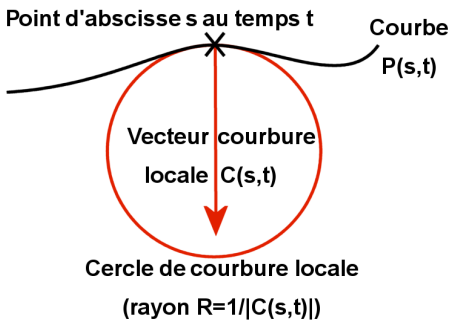
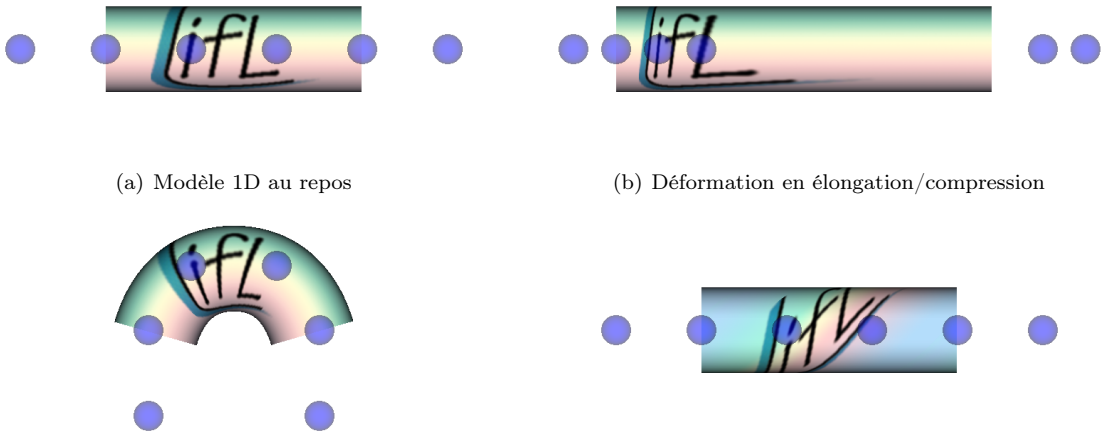


FIG. 2.3 – Schéma du vecteur courbure locale d'un modèle 1D

Torsion : Une énergie de torsion tend à faire respecter un invariant de torsion sur le modèle, autrement dit, la capacité du modèle à se tordre sur lui-même (comme indiqué sur le schéma (d) de la figure (2.4)). On distingue ici deux façons distinctes de définir une torsion, la torsion géométrique et la torsion volumique. La première est purement géométrique[TPBF87] et correspond à la variation du vecteur bi-normal du repère de Frenet (cf. annexe D). Cependant, le vecteur bi-normal dépend, par construction, du vecteur normal qui n'est pas toujours défini (cas rectiligne). Même dans l'hypothèse où le vecteur normal est défini par continuité, cela implique que la torsion est dépendante de la

courbure. Le problème de la cette méthode est qu'elle est basée sur la géométrie du modèle qui est de dimension une. Or dans la réalité, la torsion d'un matériau se mesure par rapport à ses déformations dans le volume. Ainsi, une torsion physique peut être définie à l'aide d'un paramètre dédié décorrélé de l'élongation et de la flexion. Dans ce cas, un mécanisme doit être mis en place pour établir la relation entre ce paramètre physique et les déformations géométriques du modèle. Ce nouveau paramètre vient donc augmenter la dimension du modèle géométrique 1D en lui procurant une information d'épaisseur.

Il est à noter qu'ainsi définies, les énergies sont toutes indépendantes les unes des autres. Une élongation ne doit pas provoquer de flexion ni de torsion, de même qu'une flexion n'apporte aucune énergie d'élongation ou de torsion. Enfin, une torsion du modèle peut changer sa configuration spatiale mais n'influe pas sur son élongation ni sur sa flexion.



(a) Modèle 1D au repos

(b) Déformation en élongation/compression

(c) Déformation en flexion

(d) Déformation en torsion (non issue de la 2D)

FIG. 2.4 – Exemples de déformations d'une B-spline uniforme cubique (les sphères bleues sont les points de contrôle). Le modèle est pourvu d'une épaisseur afin de mieux rendre compte des effets des différentes déformations sur les schémas (a), (b) et (c). Sur le schéma (d), l'épaisseur est nécessaire pour définir la torsion physique, la torsion géométrique étant nulle partout.

Nous allons décrire à présent l'ensemble des énergies étudiées sur notre modèle, à commencer par les énergies discrètes puis nous enchaînerons sur les énergies continues de déformation en élongation et en flexion. A noter que nous n'avons pas traité le cas de la torsion car cette déformation est due à la structure volumique de l'objet or notre modèle est purement 1D. Il faudrait donc étendre notre modèle pour permettre la gestion d'une telle déformation, par exemple à l'aide de repère locaux placés sur la courbe permettant de quantifier les éventuelles vrilles du modèle. Cette extension fait partie des perspectives de ce travail de thèse.

2.3.1 Energies discrètes

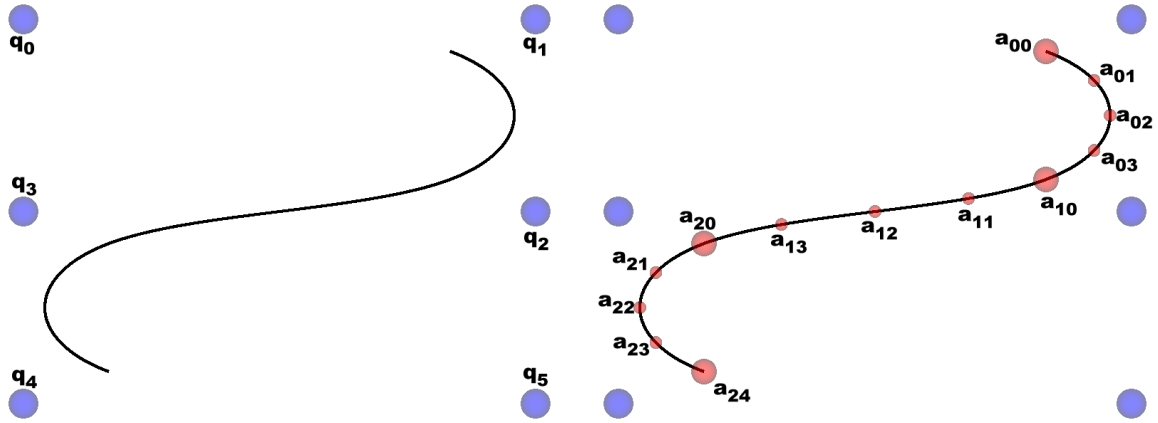
Les énergies de déformation discrètes sont basées sur des ressorts (cf. section 1.1.2.1) positionnés sur des points ponctuels du modèle. Les ressorts ont une configuration au repos et induisent des forces sur les points de support dès lors que la configuration courante s'éloigne de celle au repos.

Il est tout à fait imaginable de placer des ressorts entre les points de contrôle d'une spline. Cependant, pour les splines d'approximation, les points de contrôle ne sont pas situés sur le modèle continu (exemple de la B-spline sur la figure (2.5)). Cela n'induit donc pas de forces relatives aux déformations du modèle continu, mais définit les déformations d'un modèle masses-ressorts alors que la masse n'est pas distribuée de manière ponctuelle. Autrement dit, cela introduit un décalage entre le modèle simulé et le modèle déformé donc, une certaine instabilité.

En effet, dans notre modèle, les déformations sont dictées par la configuration courante de la courbe, autrement dit, par le modèle continu qu'est la spline. Donc, toutes les forces susceptibles de définir une déformation du modèle doivent être définies sur un point de ce modèle continu. Ainsi, les ressorts de déformations doivent impérativement être placés sur des points de la spline et non directement sur ses points de contrôle.

Donc, afin de placer correctement les ressorts de déformation, il est nécessaire d'effectuer un échantillonnage du modèle continu. Le nombre de ressorts n'est pas mathématiquement lié au nombre de degrés de liberté du modèle (i.e. sa résolution), mais physiquement l'élasticité du matériau doit être homogène, ce qui impose une certaine corrélation entre ces deux valeurs. Il est tout à fait imaginable d'avoir trop de ressorts par rapport au nombre de degrés de liberté, cela ne ferait que rendre plus précis les déformations, au détriment des temps de calcul. Par contre, ne pas avoir un nombre suffisant de ressorts (par rapport au nombre de degrés de liberté) ne permettrait pas de rendre tout les effets élastiques du matériau. Il faut donc un nombre minimum de ressorts pour chaque degré de liberté. On propose donc de définir un certain nombre de ressorts par segment spline (généralement trois). On commence donc par discrétiser spatialement la spline par rapport à l'abscisse paramétrique : on considère chaque segment paramétrique de la courbe spline, et pour chacun d'eux, on définit les points extrémités ainsi qu'un échantillonnage paramétriquement homogène de points sur le segment. On a donc un nombre de sous-points par segment paramétrique. Ce principe est illustré sur la figure (2.5), où les points construits sur la courbe sont doublement indicés, le premier indice correspond au numéro de segment et le second au numéro du point sur ce segment.

Comme la construction des points est ordonnée, on peut définir la notion de distance entre un point $\mathbf{A}$ et un point $\mathbf{B}$ comme le nombre $d(\mathbf{A}, \mathbf{B})$ de points intermédiaires qui sépare $\mathbf{A}$ de $\mathbf{B}$ plus un. Ainsi, deux points voisins sont distants de 1, de la même manière deux points distincts ayant un voisin en commun sont distants de 2.



(a) Modèle spline de base

(b) Calcul des points intermédiaires

Fig. 2.5 – Détermination des points de support des ressorts. Exemple sur une B-spline uniforme cubique. Les sphères bleues sont les points de contrôle, les rouges sont les points intermédiaires (les plus grosses sphères représentent les extrémités des segments paramétriques)

Ces points issus de la discrétisation spatiale du modèle sont les points de support des ressorts suivants :

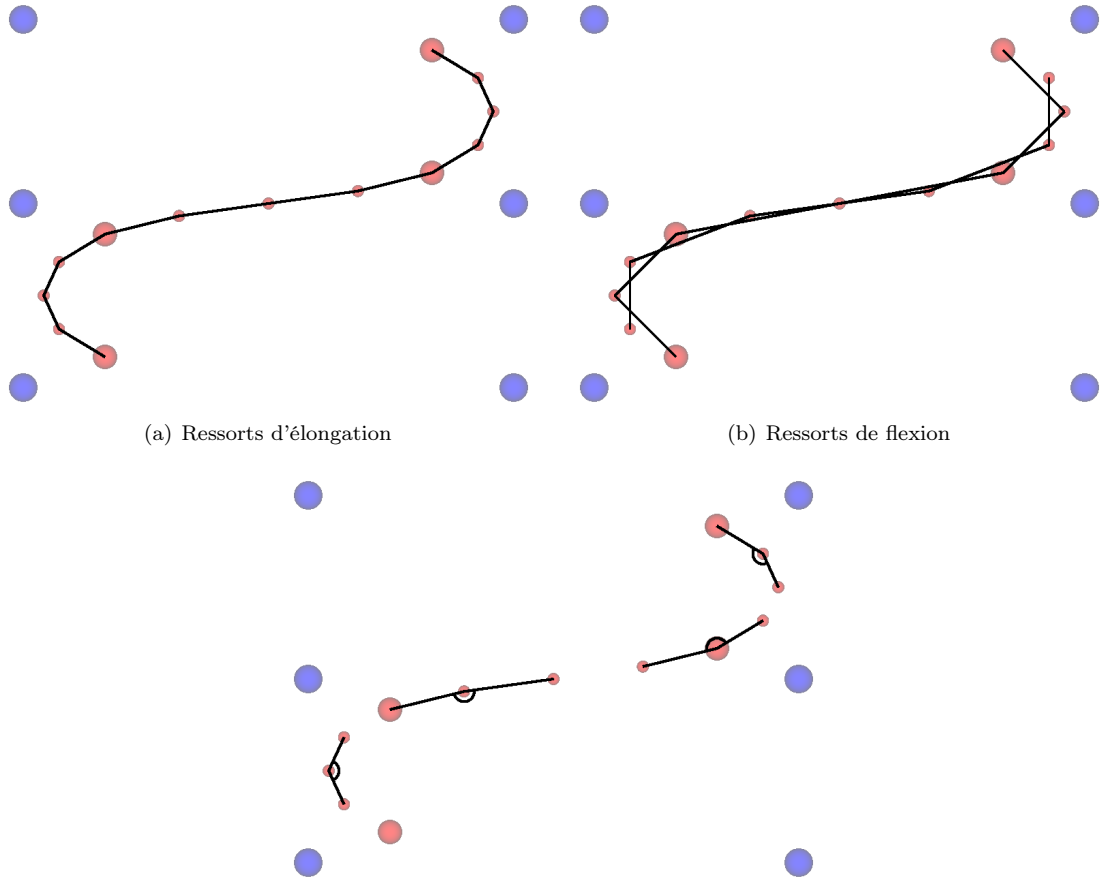
– Ressorts d'élongation/compression amortis :
  Ce type de ressort est placé sur chaque couple de points de support distant de 1 (comme indiqué sur le schéma (a) de la figure (2.6)) pour définir la force due à l'élongation locale. Ainsi, on obtient une énergie discrète d'élongation.

  On place également ce type de ressorts entre chaque couple de points de support distants de 2 comme le montre le schéma (b) de la figure (2.6). Sachant qu'un point et un voisin distants de 2 ont un voisin en commun, ils sont donc séparés par deux ressorts d'élongation. Si ces deux ressorts

ne produisent pas de force, c'est que la configuration locale n'est pas étirée. Donc, si le ressort placé entre les deux points distants de 2 produit une force, alors qu'il n'y a pas d'élongation, c'est que la configuration est courbée et donc l'énergie produite par cette force n'est pas une énergie d'élongation mais une énergie de flexion.
Il peut être noté que la flexion est généralement basée sur la courbure locale d'une courbe. Or cette courbure locale n'est rien d'autre que l'angle formé par deux vecteurs tangents locaux infinitésimalement proches. Cette approche basée sur un angle est également utilisé par un autre type de ressort, les ressorts angulaires.

– Ressorts angulaires :
Ce type de ressort a déjà été exposé dans l'état de l'art en section (1.1.2.1).
Ces ressorts angulaires sont placés sur chaque triplet consécutif de points de support comme indiqué sur le schéma (c) de la figure (2.6).

Les forces ponctuelles générées par ces ressorts sont intégrées dans les équations de Lagrange via l'expression de leurs forces généralisées Q_i^α associées à chaque coordonnée généralisée q_i^α comme indiqué par l'équation (2.15). On remarque que le nombre de ressorts présents dans le modèle déformable est dépendant de la discrétisation qui est indépendante du nombre de degrés de liberté. Donc, à la différence d'un modèle masses-ressorts, le nombre de ressorts est ici indépendant du nombre de degrés de liberté.



(a) Ressorts d'élongation

(b) Ressorts de flexion

(c) Ressorts angulaire - flexion (Dans un souci de lisibilité, un ressort sur trois est affiché)

FIG. 2.6 – Placement des ressorts sur les points de support

Cependant, le modèle physique 1D proposé est un modèle continu, il peut donc, de manière naturelle, intégrer des énergies de déformation elles aussi continues. C'est ce que nous allons étudier maintenant au travers d'énergies continues d'élongation et de flexion.

2.3.2 Energie continue d'élongation

L'énergie continue d'élongation a été étudiée dans le cas d'une spline 1D par Olivier Nocent [NR01]. Dans cette étude, Nocent se base sur une énergie induite par des déformations calculées avec le tenseur de Green-Lagrange et une distribution d'énergie associée au tenseur des déformations qui est celui de Piola Kircchoff. Cette distribution d'énergie induit une énergie d'élongation non linéaire.

Dans un souci de lisibilité, on note $\mathbf{T}(s,t)$ la dérivé paramétrique première au point $\mathbf{P}(s,t)$, ce vecteur est colinéaire à au vecteur tangent, on a donc :

$$\mathbf{T}(s,t) = \sum_{i=1}^{n} \mathbf{q_i}(t) b_i'(s)$$

et, on suppose que la configuration au repos de la courbe est donnée à l'instant $t = 0$.

Une fois appliquée au modèle spline 1D, l'énergie de déformation vaut :

$$E(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) = \frac{Y A_{ire}}{8} \int_0^1 \left(\gamma(s,t)^2 - 1\right)^2 |\mathbf{T}(s,0)|\, ds \tag{2.19}$$

avec Y le module de Young qui est caractéristique de l'élasticité du matériau, A_{ire} la surface de la section du modèle (aspect volumique de la spline) considérée comme constante et $\gamma(s,t) = \dfrac{|\mathbf{T}(s,t)|}{|\mathbf{T}(s,0)|}$ représente le pourcentage d'élongation locale ou autrement dit, le facteur relatif de dilatation locale.

Cette énergie potentielle de déformation s'intègre dans les équations de Lagrange par le biais de ses variations par rapport aux coordonnées généralisées (cf. équation (2.14)) :

$$-\frac{\partial E}{\partial q_i^\alpha} = \frac{Y A_{ire}}{2} \sum_{m=1}^{n} q_m^\alpha(t) \left[B_{im} - \sum_{p,q=1}^{n} B_{impq} \mathbf{q_p}(t).\mathbf{q_q}(t) \right] \tag{2.20}$$

avec $B_{im} = \displaystyle\int_0^1 \frac{b_i'(s) b_m'(s)}{|\mathbf{T}(s,0)|}\, ds$ et $B_{impq} = \displaystyle\int_0^1 \frac{b_i'(s) b_m'(s) b_p'(s) b_q'(s)}{|\mathbf{T}(s,0)|^3}\, ds$.

Or, Nocent fait remarquer que la localité des splines permet de réduire les intervalles de sommation au strict minimum : l'intersection entre l'intervalle discret original $\{1..n\}$ et l'intervalle discret donné par la propriété de localité $\{i-l..i+l\}$. De ce fait, l'équation (2.20) devient :

$$-\frac{\partial E}{\partial q_i^\alpha} = \frac{Y A_{ire}}{2} \sum_{m=max(1,i-l)}^{min(n,i+l)} q_m^\alpha(t) \left[B_{im} - \sum_{p,q=max(1,i-l)}^{min(n,i+l)} B_{impq} \mathbf{q_p}(t).\mathbf{q_q}(t) \right] \tag{2.21}$$

En remarquant que les termes B_{im} et B_{impq} ne dépendent pas du temps, des précalculs peuvent être menés. Les termes B_{im} sont au nombre de n^2 et les termes B_{impq} sont au nombre de n^4, ce qui peut s'avérer être un désavantage en complexité spatiale. Or, ces termes sont extrêmement redondant : en effet, on a une propriété de commutativité sur les dérivées des fonctions de base et donc sur les quatre indices des termes B_{impq}. Prenant cela en considération, on peut stocker et calculer uniquement les termes vérifiant la propriété $i \geq m \geq p \geq q$. Ensuite, la plupart de ces termes sont nuls grâce à la propriété de localité des fonctions splines et de la conservation du support entre une fonction polynômiale et ses dérivées. Ainsi, on peut calculer et stocker le sous-ensemble non nul de termes, mais cela suppose des tests supplémentaires lors de l'accès à la donnée et des indirections pour déterminer l'emplacement d'une donnée recherchée.

Ainsi, l'apport de cette énergie d'élongation par rapport à une coordonnée généralisée a une complexité en $\mathcal{O}((2l+1)^3)$. Or l est considéré comme une constante du modèle géométrique, donc la complexité est en $\mathcal{O}(1)$. L'intégration de cette énergie continue d'élongation dans les équations de Lagrange est donc en $\mathcal{O}(n)$.

2.3.3 Energie continue de flexion

Nous allons à présent proposer une énergie continue capable de contrôller les mouvements en flexion du modèle spline dynamique.

La flexion d'un modèle est liée à sa courbure et plus encore à son vecteur courbure. Or, le vecteur courbure d'une courbe géométrique peut être défini par la théorie de Frenet (cf. annexe D). De manière expérimentale, si l'on considère un modèle 1D rectiligne au repos, lorsqu'un vecteur courbure apparaît (dû à une action extérieure ou à celle de la gravité), il devrait induire une énergie locale de flexion. La force de flexion engendrée tend alors à ramener le modèle dans sa configuration au repos. Or, comme le vecteur courbure est toujours orienté à l'intérieur du cercle de courbure locale (cf. figure (2.3)), la force doit être orientée dans le même sens pour tendre à réduire cette courbure. Le vecteur courbure locale $\mathbf{C}$ est défini dans l'annexe (D). Mais on peut définir une approximation du vecteur courbure locale par simple calcul des variations du vecteur tangent :

$$\mathbf{C_a}(s,t) = \frac{d\mathbf{T}}{ds}(s,t) = \frac{d^2\mathbf{P}}{ds^2}(s,t) = \sum_{i=1}^{n} \mathbf{q_i}(t) b_i''(s) \tag{2.22}$$

Ainsi, on peut définir une configuration de flexion au repos (considérée ici comme la configuration au temps $t = 0$) et une force locale de flexion à l'instant t par

$$\mathbf{F}(s,t) = k_f \left(\mathbf{C_a}(s,t) - \mathbf{C_a}(s,0)\right) = k_f \sum_{i=1}^{n} b_i''(s) \left(\mathbf{q_i}(t) - \mathbf{q_i}(0)\right) \tag{2.23}$$

où k_f est le coefficient de flexion continue. On peut remarquer que cette force est l'expression d'un ressort d'élongation $\mathbf{C_a}(s,t) - \mathbf{C_a}(s,0)$ et de raideur k_f. L'énergie potentielle d'un tel ressort est donnée par l'expression :

$$E(s,t) = -\frac{1}{2} k_f \left(\mathbf{C_a}(s,t) - \mathbf{C_a}(s,0)\right)^2$$

Ainsi la force $\mathbf{F}$ dérive du potentiel $\mathbf{E}$ et vérifie la relation $\mathbf{F} = -\boldsymbol{\nabla} E$. Donc sa contribution aux équations de Lagrange passe par l'opposé des variations de cette énergie par rapport à chaque coordonnée généralisée (cf. équation (2.14)) :

$$\begin{aligned}
-\frac{\partial E}{\partial q_i^\alpha}(s,t) &= \frac{1}{2} k_f \frac{\partial \left(\mathbf{C_a}(s,t) - \mathbf{C_a}(s,0)\right)^2}{\partial q_i^\alpha} \\
&= \frac{1}{2} k_f 2 \frac{\partial \left(\mathbf{C_a}(s,t) - \mathbf{C_a}(s,0)\right)}{\partial q_i^\alpha} \left(\mathbf{C_a}(s,t) - \mathbf{C_a}(s,0)\right) \\
&= k_f \left(\sum_{j=1}^{n} \frac{\partial \mathbf{q_j}}{\partial q_i^\alpha}(t) b_j''(s) \right) \left(\sum_{j=1}^{n} \left(\mathbf{q_j}(t) - \mathbf{q_j}(0)\right) b_j''(s) \right) \\
&= k_f \left(\begin{pmatrix} \delta_{x\alpha} \\ \delta_{y\alpha} \\ \delta_{z\alpha} \end{pmatrix} b_i''(s) \right) \left(\sum_{j=1}^{n} \left(\mathbf{q_j}(t) - \mathbf{q_j}(0)\right) b_j''(s) \right) \\
&= k_f \sum_{j=1}^{n} \left(q_j^\alpha(t) - q_j^\alpha(0)\right) b_j''(s) b_i''(s)
\end{aligned}$$

Cette contribution ponctuelle peut être généralisée à l'ensemble du modèle spline en sommant de manière infinitésimale toutes les contributions locales :

$$\begin{aligned}
-\frac{\partial E}{\partial q_i^\alpha}(t) &= \int_0^1 k_f \sum_{j=1}^{n} \left(q_j^\alpha(t) - q_j^\alpha(0)\right) b_j''(s) b_i''(s)\, ds \\
&= k_f \sum_{j=1}^{n} \int_0^1 b_j''(s) b_i''(s)\, ds \left(q_j^\alpha(t) - q_j^\alpha(0)\right)
\end{aligned} \tag{2.24}$$

Les contributions sont simplement ajoutées au vecteur $\mathbf{B}$ dans le système d'équations linéaires du système mécanique.

Les termes calculés dépendent d'intégrales de produits de fonctions indépendantes du temps, elles sont donc précalculées et utilisées à la volée à chaque itération mécanique. Leur évaluation est donc constante et permet d'obtenir une complexité en temps en $\mathcal{O}(1)$ pour chaque terme 2.24. Ceci permet donc d'intégrer une flexion continue avec une complexité en temps en $\mathcal{O}(n)$. Le calcul des intégrales de produit de fonctions $b_i''(s)$ possède exactement les mêmes propriétés que les intégrales intervenant dans le calcul de la matrice des masses généralisées puisque le support des fonctions $b_i(s)$ et $b_i''(s)$ est le même.

Remarque :

La force de flexion est basée sur le vecteur $\mathbf{C_a}(s,t) = \frac{d^2\mathbf{P}}{ds^2}(s,t)$, or le véritable vecteur courbure locale est définie par la relation

$$\mathbf{C}(s,t) = \frac{d}{ds}\left(\frac{\mathbf{T}(s,t)}{|\mathbf{T}(s,t)|}\right)$$

avec $\mathbf{T}(s,t) = \frac{d\mathbf{P}}{ds}(s,t)$ la tangente locale à la spline. La norme de $\mathbf{T}$ indique l'élongation locale et donc la variation de l'abscisse curviligne c par rapport à l'abscisse paramétrique s est donnée par : $|\mathbf{T}(s,t)| = \frac{\partial c}{\partial s}$. Par la suite, on notera $\mathbf{T_n}(s,t) = \frac{\mathbf{T}(s,t)}{|\mathbf{T}(s,t)|}$. Le repère local de Frenet permet de trouver une relation entre les deux vecteurs courbures $\mathbf{C_a}(s,t)$ et $\mathbf{C}(s,t)$ (voir l'annexe D) :

$$\mathbf{C_a}(s,t) = \mathbf{C_{a_t}}(s,t) + \mathbf{C_{a_n}}(s,t)$$

$$\text{avec} \quad \mathbf{C_{a_t}}(s,t) = \frac{\partial^2 c}{\partial s^2}\mathbf{T_n}(s,t) \quad \text{et} \quad \mathbf{C_{a_n}}(s,t) = \frac{1}{R}\left(\frac{\partial c}{\partial s}\right)^2 \mathbf{C}(s,t)$$

où R est le rayon de courbure local de la courbe (cf. figures (2.3)).

L'énergie de flexion obtenue est liée à l'élongation à double titre. Une première composante $\mathbf{C_{a_t}}$ est directement une énergie d'élongation, qui n'apparaît que lorsque la spline se courbe. Le second terme $\mathbf{C_{a_n}}$ est une force de courbure. L'intensité de cette force dépend de l'élongation locale de la courbe.

On se place dans le cas particulier d'une loi de constitution définissant une flexion par, à la fois, une élongation d'une partie de la matière et une contraction d'une autre partie (cf. figure (2.7)). De ce fait, chaque flexion est forcément accompagnée d'une élongation. Donc, chaque flexion du modèle induit une énergie d'élongation (au travers du terme $\mathbf{C_{a_t}}$) qui va tendre à modifier l'élongation locale du modèle. Cette élongation locale induit un terme d'énergie en flexion $\mathbf{C_{a_n}}$, qui a son tour produit une flexion. L'énergie proposée permet donc de gérer un système déformable 1D à la fois en élongation et flexion. Dans cette proposition, les deux termes d'énergies sont corrélés, ce qui a pour effet de limiter les comportements possible du modèle spline dynamique.

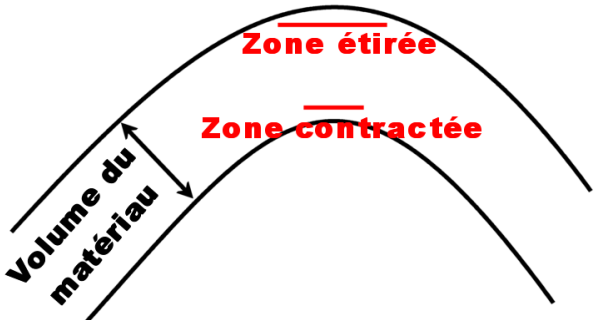


FIG. 2.7 – Déformation du volume du matériau lors d'une flexion

Il est important de noter également que l'énergie de flexion est dépendante de la paramétrisation de la courbe. Cette paramétrisation étant invariante pendant la simulation, l'énergie est elle aussi paramétriquement invariante pour une spline donnée. Si l'on défini deux courbes de même longueur et de même coefficient de flexion mais avec un nombre de points de contrôle différent, le comportement en flexion sera alors différent.

2.4 La visualisation du modèle 1D

Le modèle physique présenté permet de simuler tout objet tubulaire dont la dynamique peut se restreindre au squelette. Ainsi, on peut simuler des objets tels qu'un fil, une corde ou un lacet. Mais on peut également utiliser ce modèle pour simuler un organe filiforme comme un intestin grêle ou un être vivant comme un lombric.

De manière générale, la nature de l'objet simulé impacte grandement l'aspect visuel et la texture de ce dernier. Par exemple, une corde est d'aspect relativement rigide avec une texture de fibres entremélées, alors que la visualisation d'une simulation d'intestin grêle doit mettre en valeur son aspect plutôt organique et vivant avec une texture plutôt vascularisée. Il faut donc mettre en place un large panel de techniques de modélisation d'objets tubulaires, dont la dynamique est réduite au squelette, pour permettre de visualiser le modèle avec des caractéristiques qui lui sont propres.

Un modèle 1D peut être affiché à l'aide de différentes méthodes suivant les propriétés visuelles recherchées. Quelle que soit la technique employée, le modèle géométrique continu est discrétisé en un ensemble de m points ponctuels $\mathbf{a_i}$ constituant la base du modèle visuel : $\{\mathbf{a_i}|i \in \{1..m\}\}$. Cette discrétisation est propre à la visualisation.En se basant sur cette discrétisation de la spline, voici une liste non exhaustive des méthodes de visualisation d'un modèle filiforme :

Ligne brisée Le simple tracé de segments entre des points consécutifs situés sur l'objet permet de visualiser la configuration spatiale du modèle 1D. Il suffit donc de tracer des segments rectilignes pour chaque couple de points présents dans l'ensemble $\{(\mathbf{a_i}, \mathbf{a_{i+1}})|i \in \{1..m-1\}\}$.

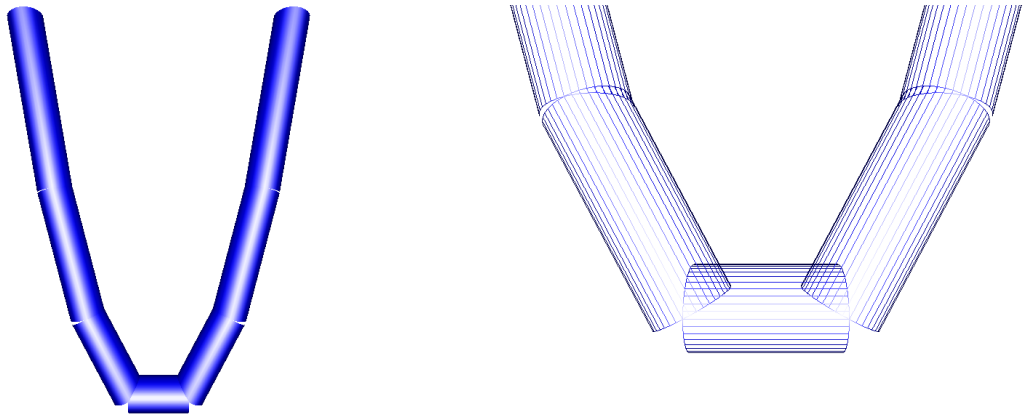
Un tel rendu est cependant de continuité C^0 par construction. De plus, le modèle est perçu à l'aide d'une simple ligne au sens mathématique du terme (i.e. sans la moindre épaisseur) et pose alors le problème de l'évaluation de la configuration dans un espace à trois dimensions. En effet, la notion d'épaisseur d'un objet permet à l'oeil de distinguer sa profondeur dans une scène ou encore sa configuration si celle-ci est complexe. Ce type d'affichage est adapté aux objets particulièrement fin dont l'épaisseur réelle est négligeable vis-à-vis de sa longueur, comme un fil de nylon...

Il peut être demandé aux cartes graphiques de tracer les lignes avec une épaisseur donnée (fonction *glLineWidth()* en OpenGL) mais, cette épaisseur est définie en unité écran (pixels) et non en distance du système physique utilisé (mètres). Elle ne résiste donc pas au changement d'échelle et, de plus, elle est tributaire de la résolution écran. Il apparaît donc clairement qu'un modèle volumique englobant le modèle 1D doit être tracé pour donner des repères visuels dans l'espace tri-dimensionnel.

Cylindres consécutifs La manière la plus simple de définir un volume englobant d'une ligne brisée est de "gonfler" les segments constituant la ligne brisée en les dotant d'une épaisseur. Ainsi, en définissant une épaisseur e pour notre modèle 1D, il suffit de remplacer les segments précédemment définis par des cylindres de rayon e. On obtient alors une succession de cylindres modélisant l'objet 1D agrémenté de son épaisseur.

Un tel rendu pose des problèmes au niveau des jonctions des cylindres consécutifs puisque, dans la majeure partie des cas, les jonctions des cylindres contigus ne sont pas continues (comme indiqué sur la figure (2.8)). On obtient donc une géométrie volumique discontinue. Ce problème restreint considérablement la classe d'objets pouvant utiliser ce mode d'affichage aux objets filiformes de faible courbure, comme les cables à hautes tensions, les tiges métalliques légèrement flexibles... Ce problème peut être résolu en observant ce qui se passe lorsque l'on dispose des cylindres sur des segments de plus en plus petits. A l'infini, les cylindres sont réduits à une simple section représentée par un cercle. Cette technique est celle du cylindre généralisé.

Cylindre généralisé [Bin87, Blo90] Le principe d'un cylindre généralisé, est de construire une surface à l'aide d'une section plane se déplacant et s'orientant le long d'un chemin. Dans notre cas, le chemin est simplement défini par le modèle géométrique spline et la section plane est un cercle. De cette manière, l'extrusion de la section sur une ligne droite donne un cylindre. Si le chemin est de continuité C^0, la section se déplace sur le chemin de manière continue mais son orientation n'est pas continue. Si le chemin est au moins de continuité C^1, alors l'orientation de la section est également continue lors de son déplacement le long de la courbe, la surface ainsi créée est continue. Un exemple de cylindre généralisé est donné sur la figure (2.9).



(a) Vue éloignée. (b) Zoom en vue filaire.

FIG. 2.8 – Affichage à l'aide de cylindres et mise en évidence des problèmes de continuité aux jonctions.

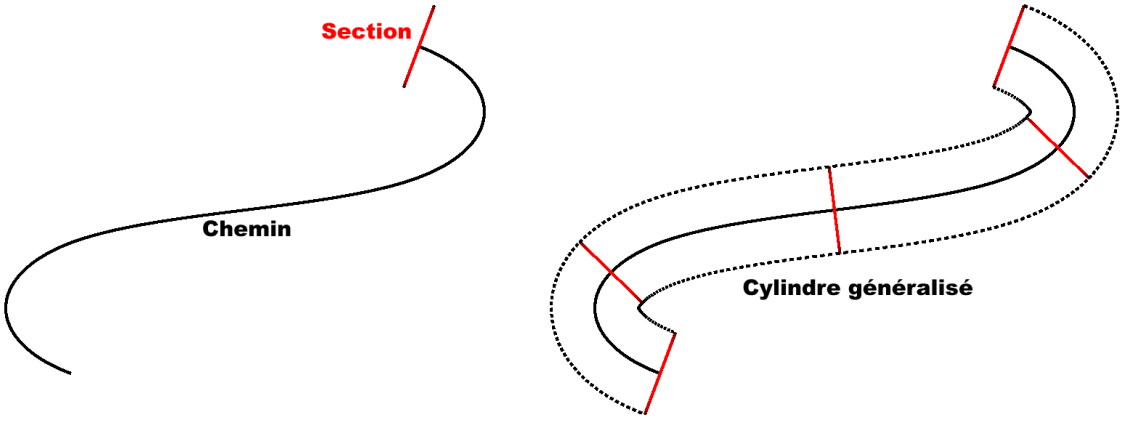


FIG. 2.9 – Schéma de construction d'un cylindre généralisé

Orienter la section dans un espace à trois dimensions n'est pas toujours évident puisque le repère local (dit trièdre de Frenet) n'est pas toujours défini. Ce repère local (relatif à une abscisse paramétrique s) est formé des vecteurs tangent $\mathbf{T}(s)$ et normal $\mathbf{N}(s)$ à la courbe ainsi que d'un vecteur nommé bi-normale $\mathbf{B}(s)$ construit à partir des deux premiers vecteurs. Pour une courbe C^2, le vecteur tangent est toujours défini, mais le vecteur courbure est nul sur les portions rectilignes. Ce manque d'information peut entraîner une torsion locale du modèle dû au fait que la section pivote sur son axe de définition (la tangente locale). Pour éviter ses torsions intempestives, on utilise une technique légèrement différente basée sur la cohérence paramétrique du repère d'une manière similaire à Christopher Twigg [14] : deux repères locaux voisins possèdent des normales et bi-normales proches. Ainsi, si à l'abscisse paramétrique s_i on a la base $(\mathbf{T}(s_i), \mathbf{N}(s_i), \mathbf{B}(s_i))$, à l'abscisse paramétrique s_{i+1}, on aura la base $(\mathbf{T}(s_{i+1}), \mathbf{N}(s_{i+1}), \mathbf{B}(s_{i+1}))$ où

$$\begin{aligned} \mathbf{B}(s_{i+1}) &= \mathbf{T}(s_{i+1}) \wedge \mathbf{N}(s_i) \text{ et} \\ \mathbf{N}(s_{i+1}) &= \mathbf{B}(s_{i+1}) \wedge \mathbf{T}(s_{i+1}) \end{aligned}$$

De plus, des problèmes peuvent apparaître dans les zones de grandes courbures où des sections voisines peuvent s'interpénétrer et ainsi définir des zones angulaires à l'intérieur de la courbure

[14] http ://www-2.cs.cmu.edu/ fp/courses/graphics/asst5/cameraMovement.pdf

(comme schématisé sur la figure (2.10)). Ce problème de discontinuité visuelle apparaît lorsque le rayon de courbure local devient plus petit que le rayon du cylindre.

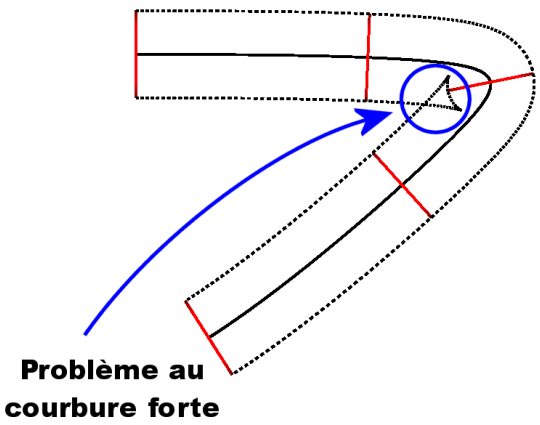


FIG. 2.10 – Mise en évidence du problème des fortes courbures pour les cylindres généralisés.

En pratique, la section se déplace sur le modèle discrétisé par les points $\mathbf{a_i}$, ce qui définit une suite de sections positionnées et orientées dans l'espace. Ensuite, une surface vient relier les sections voisines. Le cylindre généralisé procure alors une géométrie de continuité C^0, puisqu'au niveau de chaque section, les surfaces de droite et gauche possèdent cette section extrémité en commun, mais les directions ne sont pas identiques.

Cependant, une continuité visuelle plus élevée peut être obtenue à l'aide d'une technique appelée *mélange géométrique* (technique du *Vertex Blendling*)[Nvi, Blo02, GM03]. Cette technique permet de "tordre" dynamiquement et à moindre coût une géométrie fixe. Pour cela, il suffit de définir un poids w_i pour chaque sommet $\mathbf{v_i}$ de la géométrie et deux transformations M_1 et M_2 qui positionnent et orientent la géométrie fixe dans les états de début et de fin souhaités (comme indiqué sur la figure (2.11)). De cette manière, chaque point de la géométrie définit sa propre transformation comme étant le mélange des transformations de début et de fin pondéré par le poids qui leur est associé :

$$\mathbf{v_i^{final}} = w_i M_1 \mathbf{v_i} + (1 - w_i) M_2 \mathbf{v_i}$$

Ainsi, en partant d'une géométrie cylindrique fixe, avec des poids correctement choisis, et en définissant les états de début et de fin comme étant les positions et orientations de deux sections voisines, on obtient un cylindre plié et on peut assurer une continuité G^1 aux jonctions.

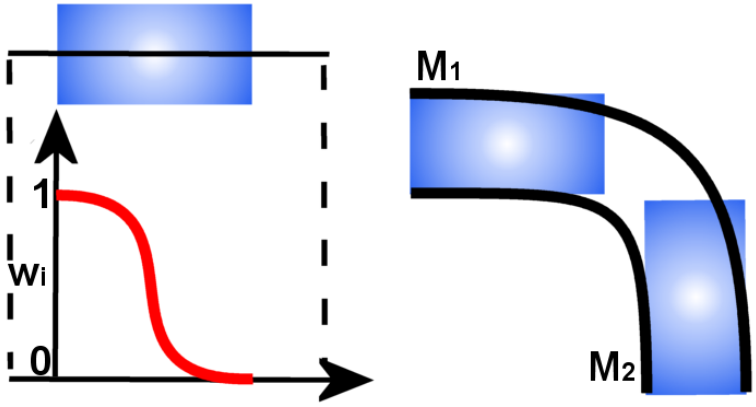


FIG. 2.11 – Technique du vertex blending

A l'aide de cette technique, on est donc capable de visualiser un modèle 1D d'une épaisseur donnée avec une surface de continuité G^1. Cette méthode d'affichage est donc adaptée à tous les objets filiformes d'aspect continu, comme les fils, les cordes, les cables, les lacets, les spaghetti... Malheureusement, le résultat d'un cylindre généralisé procure au modèle un aspect lisse et très régulier qui ne correspond pas à tous les types d'objets. En effet, cette technique d'affichage possède comme invariant la conservation de la section, ce qui procure un aplatissement aux jonctions des éléments cylindriques (comme le montre la figure (2.10)). Or, pour des objets de type organique, la visualisation doit plutôt procurer une propriété de conservation du volume. C'est pourquoi des méthodes spécifiques telles que les surfaces implicites peuvent être utilisées.

Surface implicite à squelette ponctuel [Bli82, WMW86] Une surface implicite S est définie par la donnée d'une iso-valeur e et d'une somme de fonctions potentielles S_i qui, elles-mêmes définissent des champs de potentiels dans l'espace considéré. Les potentiels S_i sont donc des fonctions définies dans cet espace et sont à valeurs réelles. Dans l'espace tri-dimensionel, une surface implicite est alors définie par :

$$S = \{\mathbf{M}(x,y,z) | \sum_{i=1}^{n} S_i(x,y,z) = e\}$$

Le fait de sommer des champs de potentiel permet de cumuler leurs influences et d'obtenir des effets de mélange comme montré sur la figure (2.12). C'est exactement cette propriété qui donne un aspect organique à la surface puisqu'elle n'est plus lisse mais dépend des champs de potentiels. Si ces derniers sont mobiles, la surface sera alors dynamique et l'objet visualisé aura l'aspect organique désiré.

(a) Influence très légère des fonctions potentielles (b) Mélange par influence des fonctions potentielles

FIG. 2.12 – Exemple d'une surface implicite à trois potentiels ponctuels

On peut alors définir les élements caractéristiques d'une fonction potentielle par la surface définie pour l'iso-valeur 0, ces élements sont appelés le squelette de la surface implicite. Par exemple, la fonction potentielle $S_i(x,y,z) = x^2 + y^2 + z^2$ définit un seul élément caractéristique qui est le point de coordonnées $(0,0,0)$. Ces fonctions potentielles basées sur des éléments caractéristiques ponctuels définissent les surfaces implicites à squelette ponctuel.

La visualisation d'une surface implicite peut être réalisée en prenant en considération son équation mathématique au travers d'un lancer de rayon. Andrei Sherstyuk propose [She98, She99b] une technique améliorée qui permet d'obtenir des résultats moins coûteux en temps de calcul. Cependant, cette technique reste inutilisable pour une application interactive. Avec cette condition, l'affichage d'une surface implicite passe obligatoirement par la construction géométrique de la surface. Frédéric Triquet [TMC01] a étudié l'utilisation des techniques dans un cadre de simulation temps réel avec des surfaces implicites à squelette ponctuel et a mis en place une technique de *marching cubes* temps

réel. Cette technique consiste à découper l'espace tri-dimensionnel en une grille régulière définissant des cubes homogènes. Ensuite, pour chaque sommet de tous les cubes, la fonction implicite est évaluée attribuant au sommet une valeur réelle. Pour chaque cube, l'ensemble des valeurs situées aux sommets définit une configuration particulière qui permet de déterminer de quelle manière la surface implicite occupe le cube. En définissant l'ensemble de toutes les configurations possibles de cube, la surface se construit par aggrégation des différentes configurations rencontrées dans chacun des cubes. Donc, en traçant les morceaux de surface définis par chaque cube, on obtient alors la surface implicite dans l'espace considéré (défini par la grille du *marching cubes*) et cette construction est continue de continuité C^0.

Antoine Leclercq [Lec04] s'appuie sur les travaux de Frédéric Triquet[TMC01] pour proposer des améliorations comme l'élimination des faces arrières, dès la génération des triangles, ou encore l'analyse par tranche de la grille pour créer progressivement l'iso-surface.

Cette technique donne de bons résultats et traduit correctement l'aspect organique et vivant de l'objet grâce au mélange des champs de potentiel. Cependant, si l'on imagine un intestin grêle posé dans le fond de la cavité abdominale, il va prendre une configuration tassée. Dans cette configuration, le mélange des champs de potentiel risque de définir une surface à la périphérie du tas et non à des endroits stratégiques (comme les zones de forte courbure). Ce mélange intrinsèque à la méthode du *marching cubes* n'est donc pas toujours souhaité et doit être maîtrisé.

Cette maîtrise peut être effective par le biais d'un graphe de mélange[DC95, TGMC03] qui recense les couples de potentiels qui sont autorisés à se mélanger. Malheureusement, cette maîtrise du mélange demande des temps de calcul plus importants que la technique de base.

L'application d'une surface implicite à potentiels ponctuels sur notre modèle passe par la discrétisation de la courbe en un ensemble de points ponctuels. Si cette discrétisation se fait de manière paramétrique, les potentiels vont suivre l'élongation du modèle et donc vont être mobiles les uns par rapport aux autres. Ceci se traduit immédiatement par le fait que les mélanges des potentiels sont variables au cours du temps dans une zone précise du modèle 1D, et donc le modèle visuel est dynamique comme peuvent l'être les objets organiques, tels que l'intestin grêle...

Surface à convolution [BS91, She99a] (sous-classe des surfaces implicites à squelette continue)

Les surfaces à convolution sont basées sur un outil mathématique appelé produit de convolution de deux fonctions. Ce produit est défini comme suit :

$$\forall f, g : \mathbb{R}^3 \longrightarrow \mathbb{R}$$
$$(f * g)(\mathbf{p}) = \int_{\mathbb{R}^3} f(\mathbf{p} - \mathbf{q}) g(\mathbf{q}) \mathbf{dq}$$

Les surfaces à convolution sont des surfaces implicites construites à partir d'éléments caractéristiques continus [FAM+02] (le squelette) et d'un *kernel* K [15] défini comme une fonction de l'espace tridimensionel, à valeur réelle : $K : \mathbb{R}^3 \longrightarrow \mathbb{R}$. Une étude comparative a été menée par Andrei Sherstyuk [She99c] sur sept *kernels* utilisables en pratique, il en résulte que le choix du kernel dépend grandement du squelette employé. Ce squelette peut être défini par :

$$S(\mathbf{p}) = \begin{cases} 1 & \texttt{si } \mathbf{p} \text{ appartient au squelette} \\ 0 & \texttt{sinon} \end{cases}$$

Ainsi, la surface de convolution d'un squelette S et d'un *kernel* K est définie en un point $\mathbf{p}$ de l'espace par :

$$(K * S)(\mathbf{p}) = \int_{\mathbb{R}^3} K(\mathbf{p} - \mathbf{q}) S(\mathbf{q}) \mathbf{dq}$$

Soit Q l'ensemble des points de l'espace appartenant au squelette, $Q = \{\mathbf{p} | S(\mathbf{p}) = 1\}$. L'équation de la surface à convolution se simplifie alors en :

$$(K * S)(\mathbf{p}) = \int_{Q} K(\mathbf{p} - \mathbf{q}) \mathbf{dq}$$

[15] *kernel*=noyau

La mise en application d'une surface à convolution passe donc par la définition d'un *kernel* et d'un squelette. Dans notre cas, le squelette est défini par une discrétisation du modèle géométrique spline en un ensemble de segments rectilignes et contigus. Le *kernel* choisi est défini par $K(\mathbf{p}) = \frac{1}{r(\mathbf{p})^3}$ avec $r(\mathbf{p})$ la distance du point $\mathbf{p}$ au squelette [CH01, FAM+02]. Le choix de ce kernel tient au fait qu'il est l'un des plus faciles et rapides à évaluer, ce qui est non négligeable pour le temps réel. Ainsi, pour un squelette continu décrit par un segment $[ab]$, on obtient la surface de convolution définie en un point $\mathbf{p}$ par[FAM+02, CH01] (cf. figre (2.13)) :

$$(K * S)(\mathbf{p}) = \frac{sin(\alpha_1) - sin(\alpha_2)}{|\mathbf{ph}|^2}$$

où $\mathbf{h}$ est le projeté orthogonal de $\mathbf{p}$ sur le segment $[ab]$, α_1 est l'angle formé par les vecteurs $([\mathbf{ph}], [\mathbf{pa}])$ et α_2 est l'angle formé par les vecteurs $([\mathbf{ph}], [\mathbf{pb}])$.

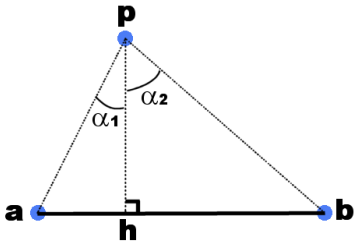


FIG. 2.13 – Schéma de calcul d'un potentiel d'une surface à convolution de squelette rectiligne

La visualisation d'une surface à convolution peut s'effectuer par un lancer de rayon comme le propose McCormack et Sherstyuk [MS98]. Cette technique donne de très bons résultats visuels mais n'est absolument pas adaptée à une application interactive. Une surface de convolution est un cas particulier de surface implicite, elle peut donc être également visualisée par la méthode du *marching cubes*. Cependant, la structure continue du squelette offre un point d'accès pour un rendu basé sur le chemin, comme le tracé d'un cylindre généralisé. Il suffit de construire à intervalles réguliers des sections épousant localement la surface de convolution. Ainsi, en rejoignant les sections voisines, on reconstruit de manière discrète la surface de convolution.

Le squelette est formé de segments rectilignes qui proviennent du modèle géométrique spline. On peut donc choisir, lors de la discrétisation du modèle géométrique, la résolution des segments utilisés pour la visualisation ainsi que leur nombre. Cependant, si le squelette ne provient pas d'un modèle continu, il est possible de subdiviser les segments discrets pour obtenir une résolution plus fine lors de la visualisation [CH01].

Les surfaces à convolution tirent parti de l'aspect continu du modèle même si celui-ci est discrétisé. La convolution permet d'éviter les renflements visuels créés d'habitude par l'influence de plusieurs fonctions potentielles discrètes. Ici, la convolution se base sur un squelette continu, ce qui permet, pour tout point de l'espace, de déterminer les zones du squelette qui influencent ce point. Ainsi, on a bien un cumul des potentiels de différents segments aux endroits courbés ce qui provoque le mélange des potentiels et donc une surface plus importante dans ces zones. De plus la convolution permet de prendre en considération les jonctions des segments sans créer de renflements ni de trous. Alors qu'une surface à distance par exemple est définie, pour une surface S, par une fonction potentielle évaluée en un point $\mathbf{p}$ par la formule : $f(S, \mathbf{p}) = max(e^{\frac{-|\mathbf{ps}|^2}{2}})$ pour $\mathbf{s} \in S$. Ainsi, le potentiel en $\mathbf{p}$ est défini à l'aide du point du squelette le plus proche de $\mathbf{p}$. Les jonctions provoquent donc des renflements [BS91]. Il est à noter que Jules Bloomenthal [Blo95] propose une technique intermédiaire permettant de maîtriser ces renflements.

A noter que les surfaces à convolution mélange des potentiels qui, comme dans le cas des surfaces implicites à squelette ponctuel, ne sont pas toujours désirable. Dans ce contexte particulier de contrôle du mélange, Hornus et al.[HAC03] proposent une technique assurant la continuité du modèle visuel en utilisant des courbes à subdivision comme squelette du modèle implicite.

2.5 Applications

La version de base du modèle permet de mettre en place quelques applications autour de la manipulation d'un objet 1D tel qu'un fil de chirurgie, une corde ou encore des organes humains filiformes comme l'intestin grêle...

Cependant, avant de présenter les tests d'applications qui ont pu être menés sur cette version du modèle, il est important de noter que le modèle évolue dans un environnement de simulation où d'autres objets peuvent cohabiter. Ceci pose le problème des collisions. Donc, nous détaillons dans un premier temps le principe des collisions dans l'environnement de simulation.

2.5.1 Le modèle de collision

La simulation prend place dans un cadre logiciel bien établi qu'est la bibliothèque de simulation SPORE développé par l'équipe GRAPHIX du LIFL. Cette bibliothèque est réalisée de manière à offrir le plus de liberté possible pour le développement de modèle. Par exemple, elle propose une large gamme de méthodes d'intégration numérique comme la méthode d'Euler explicite, Runge Kutta 4 ou encore Euler implicite par la méthode du Broyden proposée par Hilde[HMC01]. Les modèles dynamiques n'ont pas à ce soucier de la résolution de leurs équations différentielles ordinaires du second ordre, puisqu'une fois la méthode d'intégration numérique désignée, le moteur SPORE prend en charge cette phase. La seule contrainte imposée par le simulateur concerne les collisions. En effet, le moteur propose une gestion centralisée des collisions et des auto-collisions (les collisions d'un modèle sur lui même). De ce fait, un modèle unifié de collision est établi par le moteur pour tous les modèles simulés. Ce modèle est basé sur une description en sphères des objets. Ainsi, chaque modèle doit être capable de spécifier un ensemble de sphères supposées approximer l'objet simulé. En retour, le moteur de collision donne pour chacune des sphères la réponse à d'éventuelles collisions sous forme d'une force. Ensuite, chaque modèle traite cette force comme il le souhaite.

Le traitement d'une collision peut se décomposer en deux étapes : la détection et, une fois la collision identifiée, le calcul de la réponse.

Détection de la collision :

La détection est centralisée au niveau de la plate-forme de simulation et impose une approche commune à tous les modèles. Cette approche se base sur une approximation des objets par un ensemble de sphères. Donc, chaque modèle doit être en mesure de fournir au moteur de simulation une liste de sphères supposant approximer sa géométrie.

Dans notre cas, le modèle est une courbe dotée d'une certaine épaisseur lors de l'affichage, il est donc judicieux de définir les sphères de collisions sur la courbe avec un rayon correspondant à l'épaisseur perçue par l'utilisateur de manière à avoir une correspondance entre la visualisation et l'interaction. Il reste à définir le centre de chacune des sphères.

Cela peut être effectué en déterminant une discrétisation de la spline paramétriquement homogène. Ce qui a pour effet que le nombre de sphères est constant au cours du temps et que chaque sphère a une abscisse paramétrique constante. Il suffit donc de recalculer les positions et vitesses des sphères à chaque itération. Cependant, une telle distribution n'est pas adaptée à un modèle déformable car si les sphères sont réparties, de façon homogène, sur le modèle curviligne dans la configuration de départ (comme indiqué sur la sous figure (a) de la figure (2.14)), une configuration déformée apporte des zones hétérogènes où les sphères sont plus éloignées ou, au contraire, plus rapprochées suivant l'élongation du modèle comme le montre la sous figure (a) de la figure (2.14). Le problème avec une discrétisation paramétrique, c'est que le nombre d'éléments par segment spline est toujours le même quelle que soit la configuration. Ainsi, si la configuration est déformée, les éléments suivent également les déformations, introduisant des zones de grande concentration de sphères et d'autres avec des trous.

Il est donc plus judicieux de discrétiser le modèle continu par rapport à l'abscisse curviligne ainsi, l'écart entre deux sphères est fixe au cours de la simulation quelque soit la configuration du modèle. Cependant, cette technique suppose que le nombre des sphères peut varier au cours de la simulation,

(a) Configuration au repos (b) Configuration déformée

FIG. 2.14 – Distribution paramétrique des sphères de collision (en vert) sur une B-spline uniforme cubique. Les sphères bleues sont les points de contrôle.

car la courbe peut s'étirer ou se compresser suivant les déformations subies. Ceci implique que la distribution de sphères change d'une itération à l'autre et doit donc être recalculée (cf. figure (2.15)).

(a) Configuration au repos (b) Configuration déformée

FIG. 2.15 – Distribution linéique des sphères de collision (en vert) sur une B-spline uniforme cubique. Les sphères bleues sont les points de contrôle.

Voici l'algorithme de distribution des sphères de collision :

PLACESPHERESCOLLISION()

```
1  for j ← 0 to nbSegment
2  do
3     if j = 0
4        then ds ← densité ∗ r ∗ (finSegment(j) − debutSegment(j)) / longueurSegment[j]
5             s ← debutSegment(j)
6        else lprev ← longueurSegment[j − 1] ∗ [finSegment(j − 1) − (s − ds)]
7             lcur ← densité ∗ r − lprev
8             ds ← densité ∗ r ∗ (finSegment(j) − debutSegment(j)) / longueurSegment[j]
9             s ← debutSegment(j) + lcur ∗ (finSegment(j) − debutSegment(j)) / longueurSegment[j]
10    while s < finSegment(j)
11    do
12       Placer une sphère de rayon r à l'abscisse s
13       s+ = ds
```

où *nbSegment* indique le nombre de segments splines, r le rayon des sphères de collision, *densité* la densité des sphères de collision, *longueurSegment* est un tableau indiquant la longueur courante de chaque segment spline enfin, *debutSegment* et *finSegment* indique respectivement le début et la fin paramétrique du segment demandé (ces deux dernières informations sont données par le vecteur de nœuds dans le cas d'une B-spline).

Comme on définit le centre d'une sphère à l'aide de son abscisse paramétrique et que l'algorithme se base sur l'abscisse curviligne, une conversion est d'abord effectuée. Ceci intervient au niveau de la condition dans l'algorithme présenté ci-dessus. A cet endroit, on détermine l'écart paramétrique en se basant sur l'écart linéique (la distance curviligne qui sépare le centre de deux sphères voisines) qui est donné par la formule *r*densité* (cf. figure (2.16)), donc un simple produit en croix permet de déterminer l'écart paramétrique (cf. figure (2.17)). Ainsi, on remarque que le paramètre réel positif *densité* donne directement accès à la densité du recouvrement en sphère de l'objet.

Ensuite, une simple boucle distribue les sphères sur le segment de spline courant. Ceci étant effectué sur chaque segment en faisant particulièrement attention au niveau du changement de segment spline (cf. figure (2.18)) : en effet, le changement de segment peut apporter une distribution paramétrique et/ou curviligne différente, donc on recalcule l'écart paramétrique mais aussi le nouvel abscisse paramétrique de départ. Comme le nombre *r*densité* n'est pas forcément un multiple de la longueur

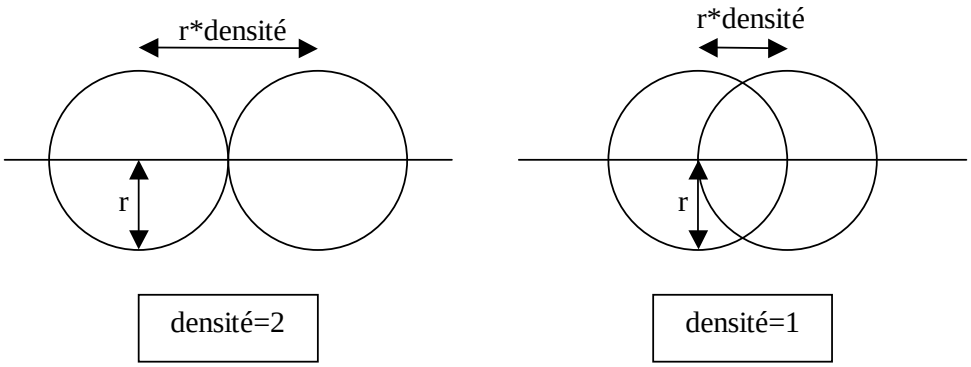


FIG. 2.16 – Densité des sphères de collision.

	Paramétrique	Curviligne
Segment courant	finSegment(j)-debutSegment(j)	longueurSegment[j]
Écart	ds= ?	dl=r*densité

FIG. 2.17 – Relation paramétrique/curviligne.

du segment précédent, la dernière sphère placée sur l'ancien segment déborde légèrement sur le nouveau segment. C'est pourquoi la première sphère du nouveau segment n'est pas placée en début de segment mais légèrement décalée (prise en compte de *lprev* sur la figure (2.18)). De manière à respecter la distance d'interpénétration entre la dernière sphère placée sur l'ancien segment et la première sphère placée sur le nouveau segment.

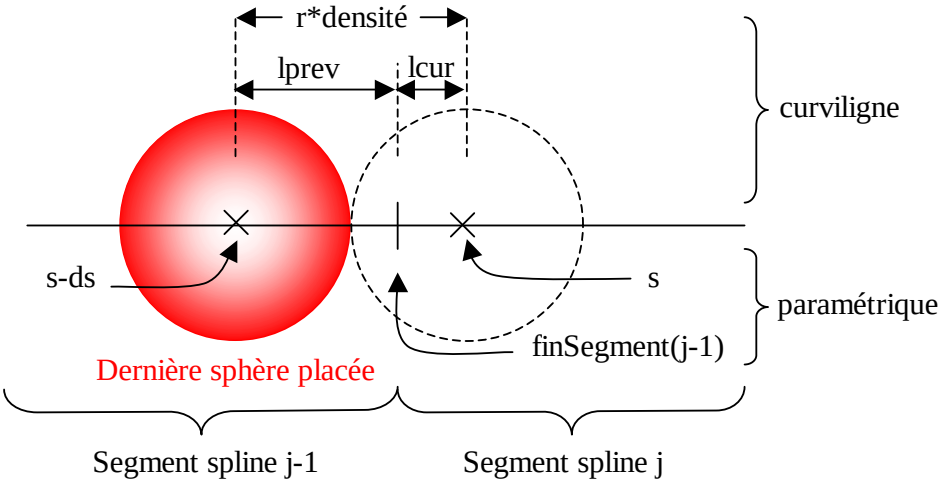


FIG. 2.18 – Changement de segment spline lors de la distribution des sphères de collision.

Réponse à la collision :

De la même manière, la réponse aux collisions étant centralisée au niveau du moteur de simulation, la réponse s'effectue avec la même méthode pour tous les objets à savoir une méthode à pénalité (cette méthode est exposée en page 50). Ceci implique que la réponse n'est pas forcément suffisante pour éviter toute interpénétration. La fonction de pénalité appliquée par la plate-forme de simulation est modélisée par un ressort (amorti uniquement pour un mouvement dans le sens de la pénétration)

(défini en 11) placée entre les centres des sphères et d'élongation, la distance d'interpénétration (la distance minimum permettant de séparer entièrement les deux sphères). Ce principe est schématisé sur la figure (2.19). Autrement dit, si les sphères ont le même rayon r, la longueur au repos l_0 du ressort est $2r$.

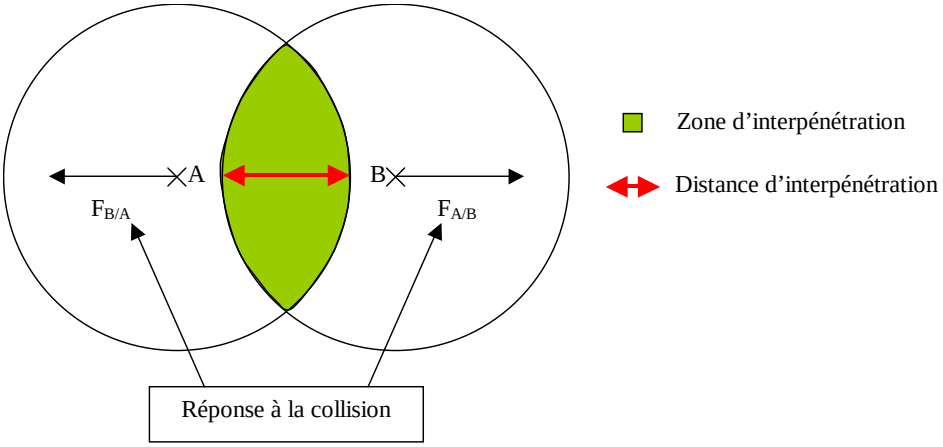


FIG. 2.19 – Schéma de calcul de la réponse à une collision.

La réponse étant basée sur une méthode à pénalité, les collisions ne sont pas robustes. Par exemple, elles n'assurent pas le contact, ou encore la cohérence physique (deux objets peuvent passer au travers l'un de l'autre si la réponse aux collisions n'est pas suffisante). Cependant, cette méthode a l'avantage d'assurer intrinsèquement la prise en compte de l'ensemble de toutes les collisions présentes à un instant donné. Là où d'autres méthodes (voir l'état de l'art sur les méthodes de contrainte page 49) demanderaient des calculs lourds à mettre en place (multiplicateurs de Lagrange) ou déploieraient des algorithmes itératifs (projection), la méthode à pénalité somme simplement les forces dues à toutes les collisions et permet donc de déterminer une configuration qui répond à toutes les collisions.

De plus, la plate-forme de simulation est capable de déterminer la liste des sphères en collision d'un même objet. Ceci permet notamment de définir les interactions d'un modèle sur lui même et donc donne une première possibilité pour gérer les auto-collisions d'un modèle. Si un objet demande à la plate-forme ce calcul d'auto-collision, seule la phase de détection est mise en place par la plate-forme, le modèle récupère alors une liste de couples de sphères en collision et traite la réponse comme il le souhaite. Il est à noter l'existence d'autres méthodes permettant de détecter les auto-collisions, comme par exemple la proposition de Raghupathi[RCFC03] qui se base à la fois sur un algorithme stochastique et sur la cohérence temporelle pour suivre l'évolution des auto-collisions.

Pour la spline, on donne la possibilité de définir les interactions de la spline sur elle même en gérant la réponse par la même méthode que dans le cas des collisions, à savoir, une pénalité basée sur un ressort amorti. Cependant, une difficulté vient de la discrétisation du modèle en sphères de collision et de la concentration variable de ces sphères. En effet, si la concentration est élevée, deux sphères voisines seront toujours en interpénétration et provoqueront toujours des forces d'auto-collision. Il faut donc mettre en place un mécanisme capable de détecter si un couple de sphères en collision est naturellement en collision (à cause de la concentration élevée) ou induit réellement une collision à prendre en compte. Pour cela, nous déterminons le nombre k de sphères voisines autorisées à être en collision. Sachant que la distance des centres de deux sphères voisines est $r * densite$, il suffit de déterminer la première sphère qui n'est plus en intersection avec la sphère de référence (comme indiqué sur la figure (2.20)) :

$$\begin{aligned} & k * r * densite && \geq 2 * r \\ \Leftrightarrow \quad & k && \geq \frac{2}{densite} \end{aligned}$$

Donc, il est possible de contrôler les interactions du modèle à l'aide du paramètre *densité* qui gére

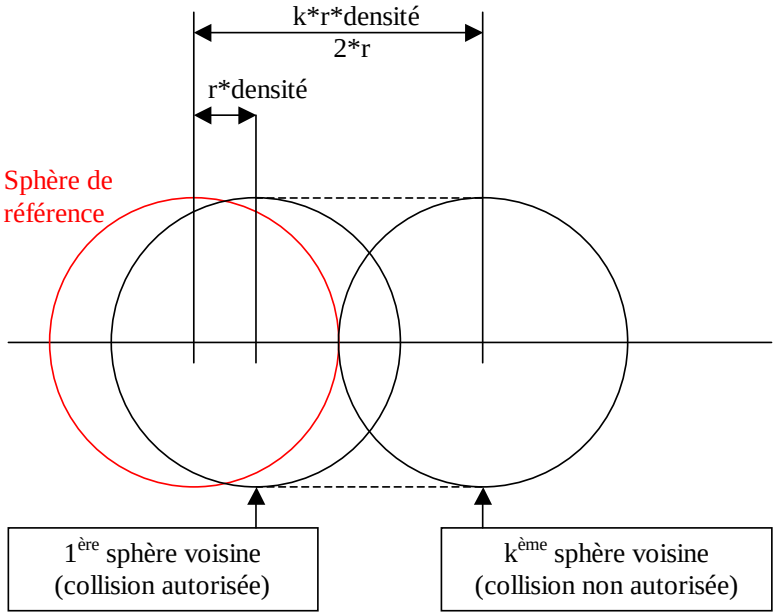


FIG. 2.20 – Schéma de détection de validité d'une auto-collision.

la concentration en sphères de collision, la raideur et l'amortissement liés aux ressorts placés sur les sphères en collision. De plus, on peut demander au modèle de gérer ces auto-collisions en utilisant les outils proposés par la plate-forme de simulation, dans ce cas, deux nouveaux paramétres (raideur et amortissement des ressorts placés entre les sphères en auto-collision) viennent compléter le modèle.

Le modèle d'interaction, qui vient d'être développé, permet de comprendre comment notre modèle peut entrer en interaction avec des modèles simulés (lui-même y compris) ou des outils manipulés par l'utilisateur.

2.5.2 Objets filiformes génériques (fils, cordes...)

Les tests présentés ici ont été réalisé à l'aide d'un pentium IV 2.4 GHz muni de 512 Mo de mémoire vive. L'ensemble des caractéristiques de chaque test ainsi que les temps de simulation sont présentés dans un tableau en page (92).

Le premier test de simulation mis en place consiste à simuler un filet constitué d'un ensemble de fils juxtaposés et indépendants comme le montre la figure (2.21). Les collisions d'un fil sur un autre fil engendrent des forces de contact ainsi que des collisions des différents fils avec une sonde manipulée par l'utilisateur. Cette première application montre la gestion des collisions du modèle et l'interaction de celui-ci avec des objets extérieurs. Dans cet exemple, la scène est constituée de plusieurs instances de notre modèle, mais les collisions sont gérées par la plate-forme et donc indépendantes du type d'objet simulé.

Le second test met en évidence l'interaction du modèle spline avec d'autres types de modèle comme des objets rigides et un tissu, modélisé par un réseau masses-ressorts, fixé à ses quatre extrémités sur des objets fixes dans l'espace. Le résultat visuel est présenté sur la figure (2.22). Le modèle physique intègre ici les collisions de la spline sur les autres objets ainsi que les collisions de la courbe sur elle même. Le modèle utilisé est une B-spline non uniforme cubique définie par 10 points de contrôle, une masse de 10 g et un coefficient d'amortissement de 0.008, une énergie continue d'élongation avec un module de Young de 1000 pour une épaisseur de section de 2 millimètres. Au niveau des collisions, les sphères de collisions ont un rayon de 2 millimètres et une concentration de 0.8. Le temps de simulation incluant uniquement le calcul d'une itération (intégration numérique comprise), est d'environ 7 millisecondes. L'affichage s'effectue à l'aide d'un cylindre généralisé.

Le test suivant met en évidence l'énergie continue de flexion du modèle au travers de deux simulations

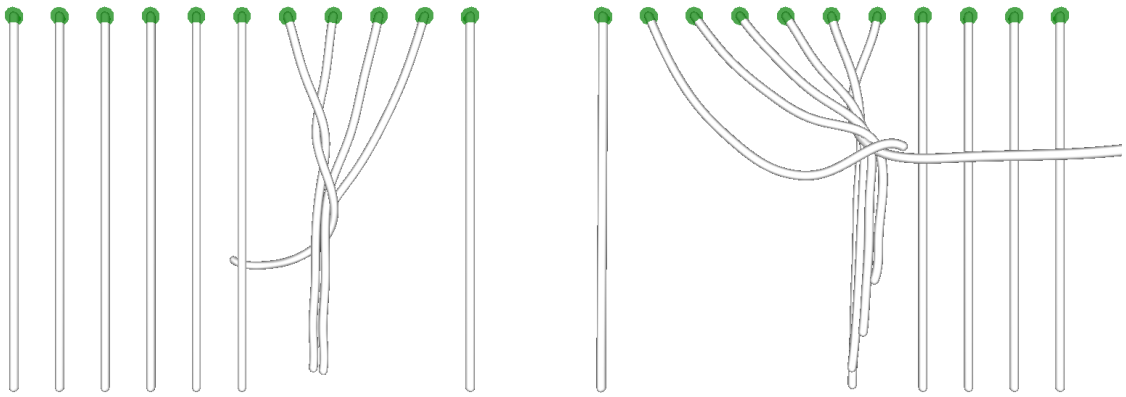


FIG. 2.21 – Simulation de plusieurs fils à base de B-splines uniformes cubiques. Chaque fil a une extrémité fixée (dans la scène) modélisée par une sphère verte.

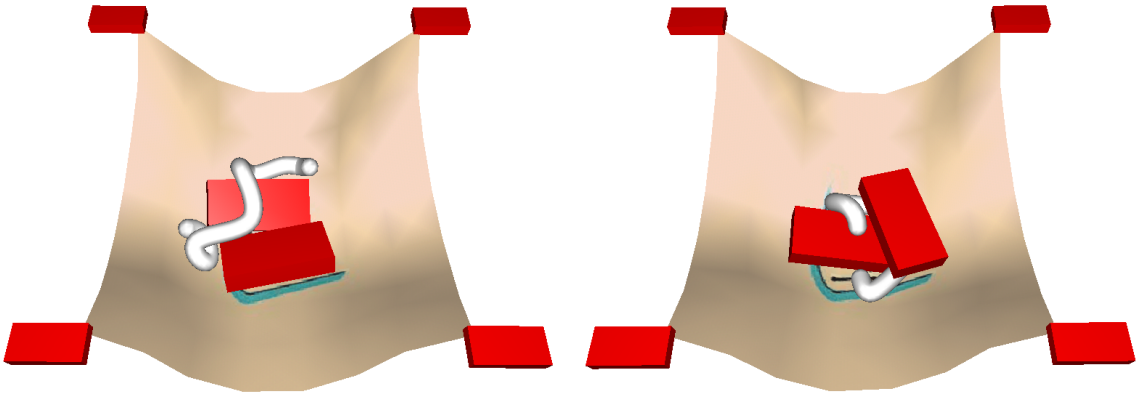


FIG. 2.22 – Interaction d'un fil avec les objets de la scène

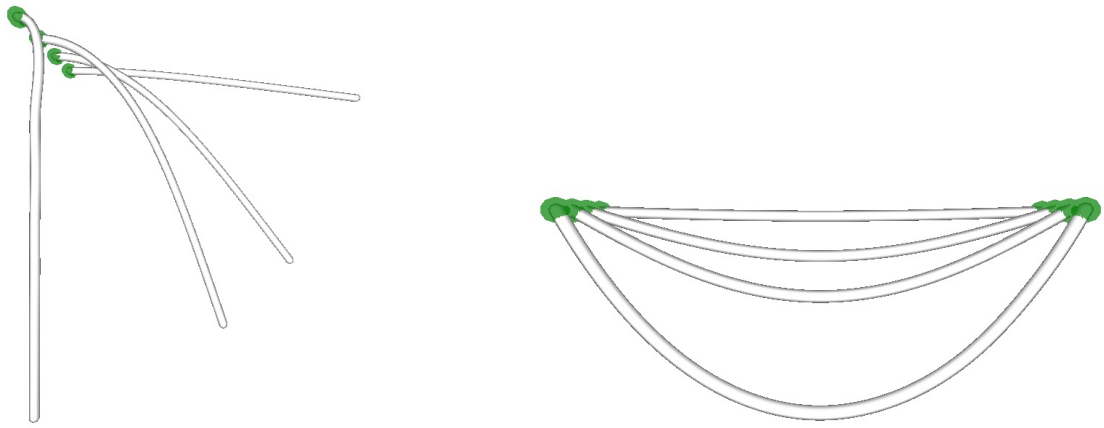
de plusieurs splines 1D possédant les mêmes caractéristiques physiques mis à part les coefficients relatifs aux énergies continues de déformation.

Sur le schéma (a) de la figure (2.23), on trouve quatre splines de 11 points de contrôle chacune, de masse 10 g, de coefficient d'amortissement 0.005. Elles sont toutes dotées de la même énergie continue d'élongation caractérisée par le module de Young d'une valeur de 5000 pour une épaisseur de 6 millimètres. Les splines sont toutes fixées à une extrémité avec en plus une contrainte de tangente qui impose, au niveau du point fixé, que la spline se dirige toujours dans la même direction. Les quatre splines sont donc dans une configuration type canne à pêche. Ce qui différencie les splines, c'est leur comportement vis à vis de la flexion. De la plus proche à la plus éloignée, on trouve un coefficient de flexion continue valant respectivement 0, 5, 10 et 100. On remarque sur le schéma que plus le coefficient de flexion est élevé et plus le comportement de la spline se rapproche d'un objet rigide.

Sur le schéma (b) de la figure (2.23), on trouve les quatre splines décrites précédemment, mis à part le fait que le module de Young est ici de 500 et que les deux points extrémités des splines sont fixés. On remarque à nouveau que plus le coefficient de flexion continue est élevé, plus la spline est rigide.

Le test suivant met également en évidence l'énergie de flexion mais cette fois-ci discrète. De la même manière, au travers de simulations de plusieurs spline 1D possédant les mêmes caractéristiques physiques mis à part les coefficients relatifs aux énergies de déformation.

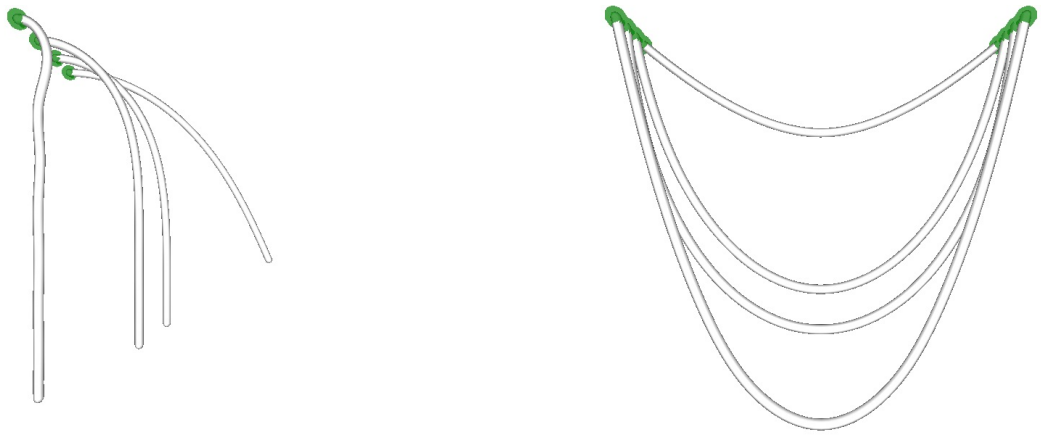
Les conclusions sur l'énergie continue de flexion se retrouve également pour les énergies discrètes de



(a) (b)

FIG. 2.23 – Mise en évidence de l'énergie de flexion continue

flexion engendrées par des ressorts angulaires. Les deux simulations mises en valeur sont présentées sur les schémas (a) et (b) de la figure 2.24.



(a) (b)

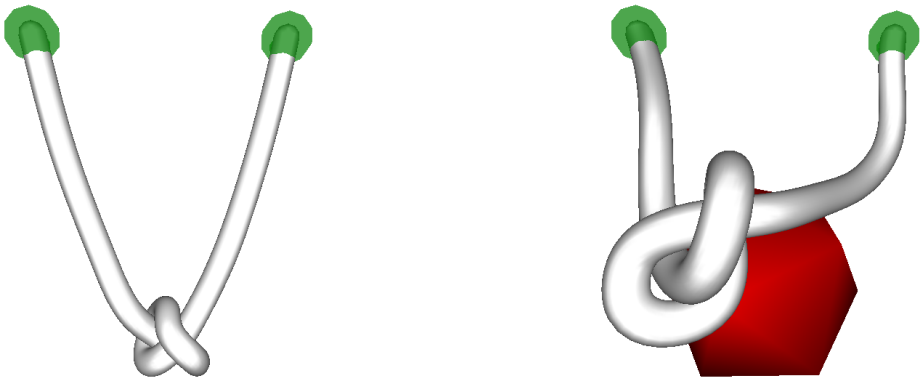
FIG. 2.24 – Mise en évidence de l'énergie de flexion discrète (ressorts angulaires)

Un autre test de cette première version du modèle physique est la simulation d'un nœud. On amorce la création d'un nœud en prenant une configuration de départ adéquate puis, en fixant les extrémités du fil, le poids de ce dernier va entraîner le serrage du nœud. Le résultat final est présenté sur la figure (2.25).

2.5.3 Simulation d'un intestin grêle

Une action de recherche coopérative (ARC) sur un simulateur de chirurgie intestinale (SCI) a permis, au travers du post doctorat de Laure France, de mettre en valeur ce modèle[FLMC02, FLA+04]. Ce projet a été réalisé en collaboration avec l'IRCAD (Institut de Recherche contre les Cancers de l'Appareil Digestif) de Strasbourg, l'équipe EVASION du laboratoire GRAVIR de Grenoble et l'ITM (Institut de Technologie Médicale) de Lille.

Un intestin grêle peut être considéré comme un objet filiforme possédant une épaisseur variable et d'aspect organique. Partant de cette constatation, l'intestin est simulé à l'aide d'une spline dynamique



(a) Simulation à l'équilibre.

(b) Interaction du modèle avec l'utilisateur via une sonde (polyèdre rouge).

FIG. 2.25 – Simulation d'un nœud. Les sphères vertes modélisent les points de la courbe fixés dans l'espace.

en définissant sa masse, son amortissement visqueux et les points de contrôle qui sont la base de la dynamique de l'objet.

Comme le modèle doit interagir avec les objets présents dans la scène, une discrétisation en sphère est calculée pour le modèle de collision. Ainsi, la densité et le rayon des sphères sont également des paramètres du modèle. On donne la possibilité de définir un rayon par points de contrôle, ce qui permet d'utiliser les fonctions splines pour interpoler ces rayons en n'importe quel point de la courbe. De cette manière, le rayon des sphères peut varier mais cette variation est continue, de même continuité que la spline.

Lors d'une visualisation de l'intestin par la méthode des *marching cubes*, les potentiels ponctuels utilisés sont situés sur la courbe et possèdent un rayon d'influence constant dans le temps mais variable sur la spline. De cette manière, l'effet organique apporté par les *marching cubes* est appuyé par la variation des influences des fonctions potentielles sommées. Ces sphères qui sont la base de la surface implicite à squelette ponctuel peuvent différer des sphères utilisées dans le modèle de collision. Ainsi, le modèle visuel possède sa propre structure de donnée discrétisant la spline.

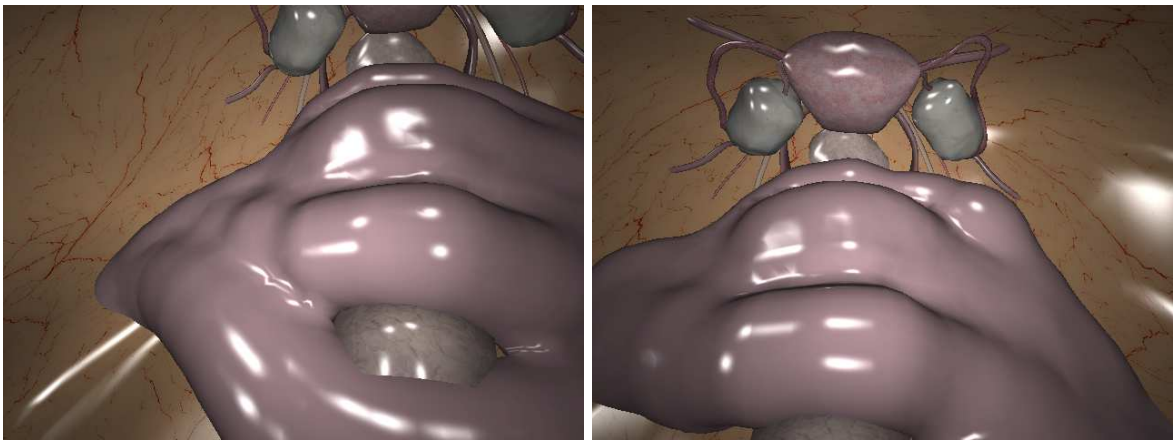


FIG. 2.26 – Simulation d'un intestin dans la cavité abdominale. Rendu à l'aide d'un *marching cubes*.

Il est envisageable d'utiliser cette même technique du rayon variable pour modéliser d'autres organes

comme le foie ou l'estomac. Ces organes n'étant pas vraiment de forme tubulaire, une telle simulation apporterait des déformations globales et aucunement surfaciques. L'utilisation de ce modèle s'avère donc possible si, par exemple, ces organes ne sont pas les centres d'intérêts principaux de la simulation.

Les tests ont étés menés sur un pentium IV 2.4 Ghz muni de 512Mo de mémoire vive. Voici un tableau récapitulatif des différents paramètres physiques utilisés et observés dans les différentes applications :

Figure	Spline	n	m	C	(k_e, a_e)	k_a	Y	r	k_f	Temps
2.21	11×UBS	10	0.01	0.03	$(100, 0)$	-	-	0.0005	-	19
2.22	NUBS	10	0.01	0.03	-	-	1000	0.002	-	7
2.23(a)	4×NUBS	11	0.01	0.005	-	-	5000	0.006	0,5,10,100	45
2.23(b)	4×NUBS	11	0.01	0.005	-	-	500	0.006	0,5,10,100	45
2.24(a)	4×NUBS	11	0.01	0.005	$(200, 0)$	0,5,10,100	-	0.006	-	6
2.24(b)	4×NUBS	11	0.01	0.005	$(1, 0)$	0,5,10,100	-	0.006	-	8
2.25	NUBS	11	0.01	0.01	-	-	800000	1.8	-	3
2.26	Cat.Rom	165	5	3	$(1000, 0)$	-	-	0.01	-	14

spline indique le type de spline dans l'ensemble $geom = \{$Catmull-Rom, B-spline uniforme cubique, B-spline non uniforme cubique$\}$
n est le nombre de points de contrôle
m est la masse totale du modèle en kilogrammes
C le coefficient d'amortissement visqueux
(k_e, a_e) sont les coefficients de raideur et d'amortissement des ressorts d'élongation
k_a est le coefficient de raideur des ressorts angulaires (flexion)
Y est le module d'Young pour l'énergie continue d'élongation
k_f est le coefficient d'énergie continue de flexion
Temps est le temps de calcul d'une itération de la simulation physique (en millisecondes).

Ce tableau montre que les temps de calcul d'une itération de simulation sont variables entre 3ms et 45ms pour 1ms virtuellement écoulée. La notion de temps réel est liée à la capacité qu'à un modèle de faire correspondre le temps de calcul d'une itération au temps virtuel écoulé durant cette itération. Nous pouvons donc conclure que nous n'avons pas tout à fait une simulation en temps réel, mais en temps interactif puisque la fréquence de simulation est supérieure au 25Hz perceptible par l'oeil humain. Ces temps de simulation permettent donc à l'utilisateur d'interagir avec le modèle spline dynamique.

De plus, au vue des résultats comportementaux, on peut conclure que les énergies d'élongations permettent de simuler des objets très élastiques à peu élastiques comme les spaghetti. Mais les objets inextensibles ne sont pas simulables à l'aide de ces énergies car elles posent des problèmes d'intégration numérique dit raides.

De même pour les énergies de flexions, elles permettent de simuler un grand nombre d'objets dont la courbure peut être très flexible ou inexistante (spaghetti...) à peu flexible (tige fine métalique légèrement flexible, antenne automobile, canne à pêche...).

2.6 Extension du modèle aux dimensions supérieures

Cette première version du modèle physique 1D peut être étendue aux dimensions supérieures à l'aide du produit tensoriel de spline, comme le propose Rémion et al.[RNN00]. On obtient alors une surface spline (2D) ou un volume spline (3D). Les équations de Lagrange qui régissent la dynamique de l'objet restent les mêmes mis à part le fait que le nombre de degrés de liberté augmente considérablement. Nous détaillons dans un premier temps le cas 2D et dans un second temps, le cas 3D.

2.6.1 Modèle de spline dynamique en dimension 2

En dimension deux, la surface spline est définie à l'aide d'une grille à deux dimensions $(n_1 \times n_2)$ de points de contrôle $\mathbf{q}_{\mathbf{i}_1,\mathbf{i}_2}$. Ainsi, les degrés de liberté sont maintenant indicés par deux entiers, le premier

indique le numéro de ligne sur la grille (dans l'intervalle $\{1..n_1\}$) et le second précise le numéro de colonne (dans l'intervalle $\{1..n_2\}$). Un point de la surface spline est alors défini à l'aide de deux abscisses paramétriques s_1 et s_2 par la formule :

$$\mathbf{P}(s_1, s_2, t) = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \mathbf{q_{i_1,i_2}}(t) b_{i_1}(s_1) b_{i_2}(s_2)$$

La vitesse de ce point est alors donnée par la dérivée première de la position par rapport au temps :

$$\dot{\mathbf{P}}(s_1, s_2, t) = \frac{d\mathbf{P}}{dt}(s_1, s_2, t) = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \dot{\mathbf{q}}_{\mathbf{i_1,i_2}}(t) b_{i_1}(s_1) b_{i_2}(s_2) \tag{2.25}$$

Le modèle ainsi étendu peut être animé par les équations de Lagrange suivantes :

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}^\alpha_{i_1,i_2}}\right) + \frac{\partial K}{\partial q^\alpha_{i_1,i_2}} = Q^\alpha_{i_1,i_2} - \frac{\partial E}{\partial q^\alpha_{i_1,i_2}} \quad , \ \forall (i_1, i_2) \in \{1..n_1\} \times \{1..n_2\} \ \texttt{et} \ \forall \alpha \in \{x, y, z\} \tag{2.26}$$

Partie gauche des équations de Lagrange :

En considérant à nouveau que la distribution de masse est constante (égale à m) par rapport à l'abscisse paramétrique, l'énergie cinétique d'une spline 2D est donnée par la formule

$$K(t) = \frac{1}{2} m \int_0^1 \int_0^1 \dot{\mathbf{P}}(s_1, s_2, t)^2 \, ds_1 \, ds_2 \tag{2.27}$$

En développant la partie gauche de l'équation (2.26) à l'aide des équations (2.27) et (2.25) de la même manière que pour une spline 1D, on aboutit à :

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}^\alpha_{i_1,i_2}}\right) + \frac{\partial K}{\partial q^\alpha_{i_1,i_2}} &= m \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \int_0^1 \int_0^1 b_{i_1}(s_1) b_{i_2}(s_2) b_{j_1}(s_1) b_{j_2}(s_2) \, ds_1 \, ds_2 \ddot{q}^\alpha_{j_1,j_2}(t) \\ &= m \sum_{j_1=1}^{n_1} \left[\int_0^1 b_{i_1}(s_1) b_{j_1}(s_1) \, ds_1\right] \sum_{j_2=1}^{n_2} \left[\int_0^1 b_{i_2}(s_2) b_{j_2}(s_2) \, ds_2\right] \ddot{q}^\alpha_{j_1,j_2}(t) \end{aligned} \tag{2.28}$$

Partie droite des équations de Lagrange :

La partie droite des équations de Lagrange pour une spline 2D se développe de la même manière que dans le cas 1D. On aboutit donc à une sommation des contributions dans un vecteur $\mathbf{B}$ de dimension $3n_1n_2$.

La contribution d'une force $\mathbf{F}$ appliquée en $\mathbf{P}(s_1, s_2, t)$ est définie par sa force généralisée associée à la coordonnée généralisée $q^\alpha_{i_1,i_2}$:

$$Q^\alpha_{i_1,i_2} = F^\alpha b_{i_1}(s_1) b_{i_2}(s_2)$$

Comme dans le cas 1D, si la force $\mathbf{F}$ dérive d'un potentiel, sa contribution au sein des équations de Lagrange se mesure par sa variation vis à vis de chaque coordonnée généralisée : $-\dfrac{\partial E}{\partial q^\alpha_{i_1,i_2}}$.

Parmi les forces qui s'exercent sur le modèle, on trouve les forces de collision qui proviennent de la discrétisation du modèle en sphères. Cette discrétisation est faite en définissant des sphères de manière paramétriquement régulière sur la spline 2D. Dans un souci de rapidité de calcul, la discrétisation est faite de manière paramétrique et permet donc de ne pas recalculer une discrétisation à chaque itération de la simulation. Les sphères sont distribuées avant le lancement de la simulation par rapport aux deux

abscisses paramétriques. A chaque itération, le modèle de collision ne fait que mettre à jour les positions et vitesses des sphères vis-à-vis des abscisses paramétriques fixées.

En ce qui concerne les énergies de déformation, une discrétisation paramétriquement régulière est calculée sur la surface spline 2D et sert de support à un réseau de ressorts d'élongations et de flexions [Pro95]. On trouve donc pour chaque carré issu de la discrétisation quatre ressorts d'élongation (placés sur les arrêtes) et deux ressorts de flexion (au niveau des diagonales).

Propriété du système d'équations linéaires :

Le système d'équations linéaires peut donc s'écrire sous la forme matricielle $\mathcal{M}.\mathbf{A} = \mathbf{B}$. On remarque que les éléments de la matrice des masses généralisées sont indépendants des axes, ce qui a pour conséquence que la matrice $\mathcal{M}$ est diagonale par bloc. De plus, pour un couple d'indices (i_1, i_2) donné, les facteurs des accélérations des degrés de liberté sont identiques pour les trois axes $\{x, y, z\}$, ce qui entraîne que les trois matrices présentes sur la diagonale de $\mathcal{M}$ sont identiques à une même matrice M.

Pour définir les éléments de M, il faut d'abord fixer l'organisation des données dans la structure matricielle. On définit donc un élément M_{ij} comme étant une partie de l'équation de Lagrange (2.28) où i_1 et i_2 sont déterminés à partir de l'indice i par les formules : [16]

$$i_1 = (i-1)/n_2 + 1 \ \texttt{et} \ i_2 = (i-1)\%n_2 + 1$$

Cette équation (2.28) définit une ligne de la matrice M dont chaque élément est déterminé par un terme de la somme qui intervient dans cette même équation. Ce terme peut donc être défini par le couple d'indices (j_1, j_2) de la sommation qui, de la même manière, est déterminé à partir de l'indice j par les formules :

$$j_1 = (j-1)/n_2 + 1 \ \texttt{et} \ j_2 = (j-1)\%n_2 + 1 \tag{2.29}$$

M ainsi structurée est alors composée des éléments M_{ij} :

$$M_{ij} = m \int_0^1 b_{i_1}(s_1) b_{j_1}(s_1)\, ds_1 \int_0^1 b_{i_2}(s_2) b_{j_2}(s_2)\, ds_2 = M_{ji}$$

La propriété de localité des splines (à savoir qu'une fonction de base de spline influence l segments paramétriques) permet de définir un motif particulier sur la matrice M. En effet, pour une ligne i de M, les termes non nuls sont les termes des colonnes $i-l$ à $i+l$ (dans l'intervalle limite $\{1..n_1 n_2\}$). De plus, la structure 2D de la spline permet à un point d'une ligne d'influencer de manière isotrope les points voisins. Ce qui implique que notre point d'indices (i, i) influence également les points sur les lignes voisines. Là encore, la propriété de localité permet de localiser les lignes influencées : $i-l$ à $i+l$. Donc, sur une ligne i de la matrice M, les termes non nuls sont les termes M_{ij} pour j dans les intervalles $\{i-l+kn_2..i+l+kn_2\}$ avec $k \in \{-l..l\}$. Ces intervalles désignent sur la structure 2D des points de contrôle, les points situés dans un carré d'arrête $2l+1$ centré sur le point d'indices (i, i), le tout bien évidemment tronqué à l'intervalle limite $\{1..n_1 n_2\}$.

A l'aide de cette remarque sur la structure de la matrice M, on peut définir une structure bande si le nombre de degrés de liberté est conséquent. En effet, si l'on considére sur une ligne i des blocs de données groupant les informations de la structure 2D par ligne, alors la localité nous dit que seul $2l+1$ blocs centrés sur i possèderont des termes non nuls. Ainsi, on obtient bien une matrice M avec une structure bande de largeur $((2l+1)n_2)$ puisqu'on a $2l+1$ blocs de la largeur d'une ligne dans la structure 2D, donc n_2.

Résultats

La visualisation d'une spline 2D passe par sa discrétisation en une grille 2D paramétriquement régulière. Il suffit alors d'afficher la grille de points discrets. La paramétrisation permet de calculer aisément

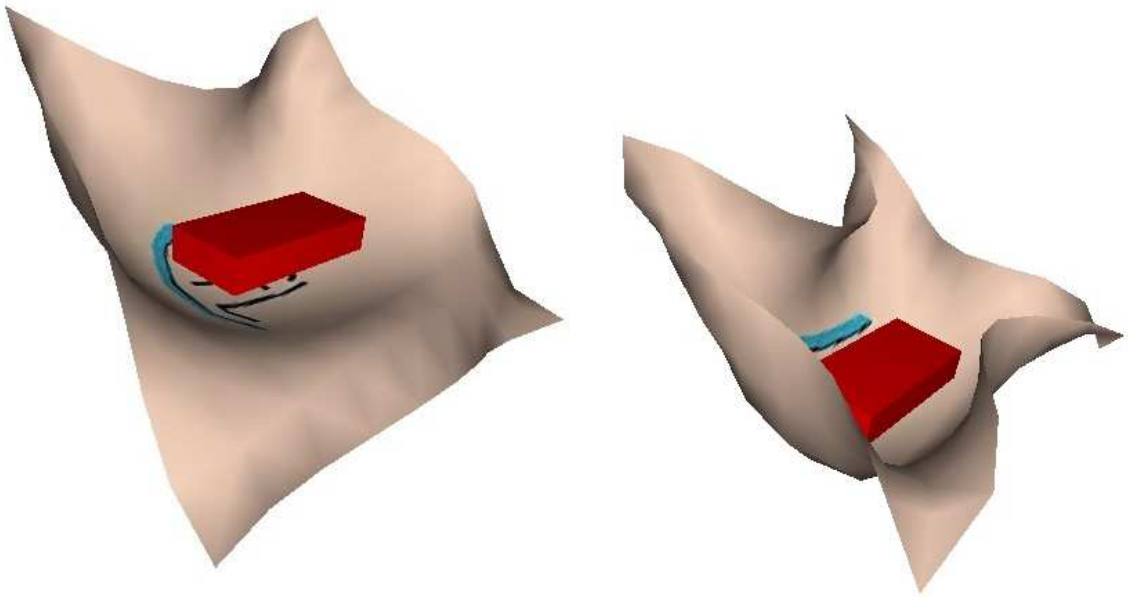
[16] Les symboles '/' et '%' désignent respectivement le quotient et le reste de la division entière

des coordonnées de texture et ainsi plaquer une texture sur le modèle. Les normales $\mathbf{N}(s_1, s_2, t)$, utilisées lors du calcul d'éclairement local, sont le résultat du produit vectoriel des tangentes dans les deux directions paramétriques $\mathbf{T_{s_1}}(s_1, s_2, t)$ et $\mathbf{T_{s_2}}(s_1, s_2, t)$ avec

$$\mathbf{T_{s_1}}(s_1, s_2, t) = \frac{d\mathbf{P}}{ds_1}(s_1, s_2, t) \text{ et } \mathbf{T_{s_2}}(s_1, s_2, t) = \frac{d\mathbf{P}}{ds_2}(s_1, s_2, t)$$

Un exemple de simulation d'une spline 2D est présenté sur la figure (2.27). Dans cette simulation, la spline est composée de (6×6) points de contrôle approximés par une B-spline uniforme cubique 2D. Le modèle physique possède une masse de 5 g, un coefficient d'amortissement visqueux de 0.003, des ressorts isotropes d'élongation de raideur 1 et des ressorts isotropes de flexion de raideur 1.

Dans cet exemple, la scène est composée de la spline 2D et d'un pavé rigide de coefficient d'amortissement moindre. De ce fait, le pavé tombe plus vite que la spline et permet donc de la déformer en tombant dessus. Le temps de calcul d'une itération de la simulation mécanique est de 30 millisecondes (le temps de calcul de la simulation sans le pavé est de 4 ms).

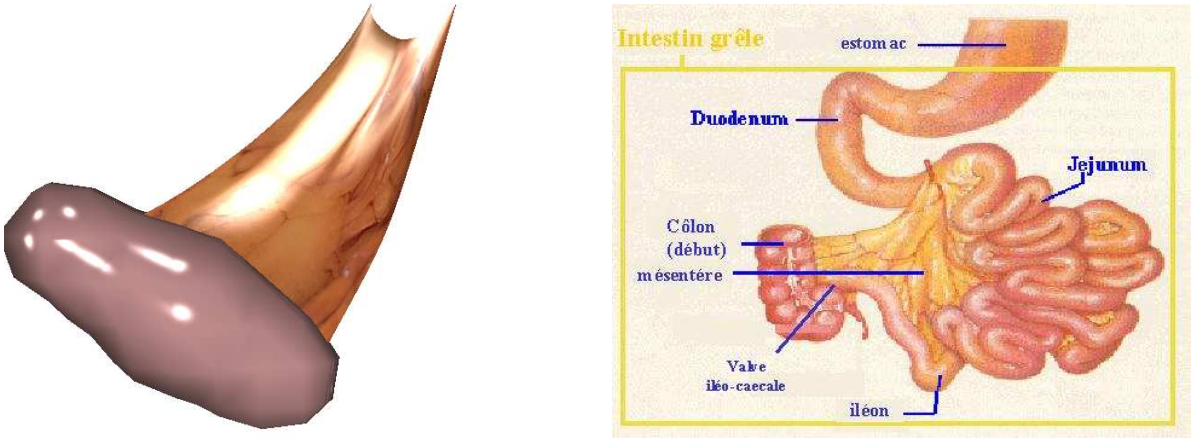


(a) (b)

FIG. 2.27 – Simulation d'un pavé tombant dans une spline 2D

Une application possible de la spline 2D dynamique réside dans la simulation chirurgicale. En effet, anatomiquement parlant, l'intestin grêle est fixé à la cavité abdominale par un tissu organique nommé le *mésentère*. Nous proposons donc de simuler le mésentère avec une spline dynamique 2D et de modéliser l'intestin grêle directement sur ce modèle à l'aide d'un coté du tissu. Le mésentère est affiché en utilisant le rendu classique d'une spline 2D avec une texture adéquate et l'intestin grêle est affiché avec la méthode des *marching cubes* (un état de la simulation est présenté sur la figure (2.28)).

Sur cet exemple, la spline 2D simulée a une masse de 30 g, un coefficient d'amortissement visqueux de 0.5 et l'intestin possède une épaisseur de 5 cm. La spline 2D est définie par 2×2 segments paramétriques formant un carré de 40 cm d'arête. Le temps de simulation d'une itération est de 3 ms. Il est à noter qu'un intestin grêle réel s'étend en moyenne sur 4 mètres. La simulation du mésentère lié à un intestin grêle de cette taille provoque des instabilités dans la simulation à cause notamment des forts coefficients de raideur des ressorts. De plus, une telle simulation s'effectuerait non plus en temps réel, mais en temps interactif (une vingtaine d'itérations par seconde). Dans le cadre de l'ARC, d'autres modèles ont été proposés.



(a) Un état de la simulation

(b) Schéma explicatif de l'intestin grêle

FIG. 2.28 – Simulation de l'intestin grêle et du mésentère à l'aide d'une spline 2D.

En terme de complexité, la simulation physique du modèle spline est la même dans les cas 1D et 2D. Cependant, le nombre de degrés de liberté est ici plus important. Si une spline 1D cubique demande un minimum de quatre points, une surface spline cubique en demande seize. De plus, la surface déformable requiert un plus grand nombre de ressorts qu'un modèle de spline 1D. Ainsi, l'évaluation des déformations nécessite également des temps de calcul plus importants. Enfin, le modèle de collision à base de sphère n'est pas adapté aux modèles surfaciques, soit les sphères ont un rayon faible pour approximer la faible épaisseur de la surfaceet on place un grand nombre de ces sphères, soit les sphères sont de rayon plus important et on s'éloigne de l'aspect surfacique au niveau des interactions.

2.6.2 Modèle de spline dynamique en dimension 3

En dimension trois, le volume spline est défini à l'aide d'une grille à trois dimensions $(n_1 \times n_2 \times n_3)$ de points de contrôle $\mathbf{q_{i_1,i_2,i_3}}$. Ainsi, les degrés de liberté sont maintenant indicés par trois entiers, le premier indique le numéro de ligne sur la grille (dans l'intervalle $\{1..n_1\}$), le second précise le numéro de colonne (dans l'intervalle $\{1..n_2\}$) et le troisième renseigne sur la profondeur dans la grille (dans l'intervalle $\{1..n_3\}$). Un point du volume spline est alors défini à l'aide de trois abscisses paramétriques s_1, s_2 et s_3 par la formule :

$$\mathbf{P}(s_1,s_2,s_3,t) = \sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\sum_{i_3=1}^{n_3} \mathbf{q_{i_1,i_2,i_3}}(t) b_{i_1}(s_1) b_{i_2}(s_2) b_{i_3}(s_3)$$

La vitesse de ce point est alors donnée par la dérivé première de la position par rapport au temps :

$$\dot{\mathbf{P}}(s_1,s_2,s_3,t) = \frac{d\mathbf{P}}{dt}(s_1,s_2,s_3,t) = \sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\sum_{i_3=1}^{n_3} \dot{\mathbf{q}}_{\mathbf{i_1,i_2,i_3}}(t) b_{i_1}(s_1) b_{i_2}(s_2) b_{i_3}(s_3) \tag{2.30}$$

Le modèle ainsi étendu peut être animé par les équations de Lagrange suivantes :

$$\forall (i_1,i_2,i_3) \in \{1..n_1\}\times\{1..n_2\}\times\{1..n_3\} \texttt{ et } \forall \alpha \in \{x,y,z\} \ : \\ \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}^{\alpha}_{i_1,i_2,i_3}}\right) + \frac{\partial K}{\partial q^{\alpha}_{i_1,i_2,i_3}} = Q^{\alpha}_{i_1,i_2,i_3} - \frac{\partial E}{\partial q^{\alpha}_{i_1,i_2,i_3}} \tag{2.31}$$

Partie gauche des équations de Lagrange :

En considérant à nouveau que la distribution de masse est constante (égale à m) par rapport à l'abscisse paramétrique, l'énergie cinétique d'une spline 3D est donnée par la formule

$$K(t) = \frac{1}{2} m \int_0^1 \int_0^1 \int_0^1 \dot{\mathbf{P}}(s_1, s_2, s_3, t)^2 \, ds_1 \, ds_2 \, ds_3 \tag{2.32}$$

En développant la partie gauche de l'équation (2.31) à l'aide des équations (2.32) et (2.30) de la même manière que pour les spline 1D et 2D, on aboutit à :

$$\begin{aligned}
&\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}^{\alpha}_{i_1,i_2,i_3}}\right) + \frac{\partial K}{\partial q^{\alpha}_{i_1,i_2,i_3}} \\
&= m \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \sum_{j_3=1}^{n_3} \int_0^1 \int_0^1 \int_0^1 b_{i_1}(s_1) b_{i_2}(s_2) b_{i_3}(s_3) b_{j_1}(s_1) b_{j_2}(s_2) b_{j_3}(s_3) \, ds_1 \, ds_2 \, ds_3 \ddot{q}^{\alpha}_{j_1,j_2,j_3}(t) \\
&= m \sum_{j_1=1}^{n_1} \left[\int_0^1 b_{i_1}(s_1) b_{j_1}(s_1) \, ds_1\right] \sum_{j_2=1}^{n_2} \left[\int_0^1 b_{i_2}(s_2) b_{j_2}(s_2) \, ds_2\right] \sum_{j_3=1}^{n_3} \left[\int_0^1 b_{i_3}(s_3) b_{j_3}(s_3) \, ds_3\right] \ddot{q}^{\alpha}_{j_1,j_2,j_3}(t)
\end{aligned} \tag{2.33}$$

Partie droite des équations de Lagrange :

La partie droite des équations de Lagrange pour une spline 3D se développe de la même manière que dans les cas 1D et 2D. On aboutit donc à une sommation des contributions dans un vecteur $\mathbf{B}$ de dimension $3n_1n_2n_3$.

La contribution d'une force $\mathbf{F}$ appliquée en $\mathbf{P}(s_1, s_2, s_3, t)$ est définie par sa force généralisée associée à la coordonnée généralisée $q^{\alpha}_{i_1,i_2,i_3}$:

$$Q^{\alpha}_{i_1,i_2,i_3} = F^{\alpha} b_{i_1}(s_1) b_{i_2}(s_2) b_{i_3}(s_3)$$

Comme dans les cas 1D et 2D, si la force $\mathbf{F}$ dérive d'un potentiel, sa contribution au sein des équations de Lagrange se mesure par sa variation vis-à-vis de chaque coordonnée généralisée : $-\dfrac{\partial E}{\partial q^{\alpha}_{i_1,i_2,i_3}}$.

La discrétisation du modèle en sphères de collision s'effectue de manière statique par rapport aux abscisses paramétriques. Ainsi, à chaque itération, il suffit de recalculer les positions et vitesses de chacune d'elle sans pour autant recalculer une distribution sur le modèle paramétrique.

L'énergie de déformation du modèle est assurée par des ressorts en élongation, flexion et torsion disposés sur une discrétisation paramétriquement régulière de la spline 3D. Ainsi, dans ce cas chaque cube de la discrétisation renferme douze ressorts d'élongations placés sur les arrêtes, douze ressorts de flexions placés sur les faces (deux ressorts croisés par face) et quatre ressorts de torsions occupant les diagonales internes du cube.

Propriété du système d'équations linéaires :

Le système d'équations linéaires obtenu peut s'écrire sous la forme matricielle $\mathcal{M}.\mathbf{A} = \mathbf{B}$. On retrouve ici les mêmes propriétés que dans le cas 2D, à savoir :

- M est diagonale par bloc.
- Les blocs sont tous les trois identiques, égaux à une matrice notée M.

L'organisation de la matrice M s'effectue en considérant d'abord le plan de coupe en profondeur, puis on détermine sur ce plan, l'indice de la ligne et de la colonne. De ce fait, un indice i code le point effectif d'indice (i_1, i_2, i_3) où

$$\begin{aligned}
i_1 &= ((i - i_3 n_3) - 1)/n_2 + 1 \\
i_2 &= ((i - i_3 n_3) - 1)\%n_2 + 1 \\
i_3 &= (i - 1)/(n_1 n_2) + 1
\end{aligned} \tag{2.34}$$

Ainsi structurée, la matrice M est alors définie par ses éléments M_{ij}. Des indices i et j, on peut déduire les deux points de contrôle mis en relation par leurs indices respectifs (i_1, i_2, i_3) et (j_1, j_2, j_3). De cette manière, la matrice M est composée des éléments suivants :

$$M_{ij} = m \int_0^1 b_{i_1}(s_1) b_{j_1}(s_1)\, ds_1 \int_0^1 b_{i_2}(s_2) b_{j_2}(s_2)\, ds_2 \int_0^1 b_{i_3}(s_3) b_{j_3}(s_3)\, ds_3$$

Les mêmes propriétés que dans le cas 2D s'appliquent ici, à savoir que la matrice M est bande si la taille des données est conséquente vis-à-vis de la localité des splines. Dans ce cas, la largeur de la bande est de $((2l+1)n_1n_2)$ puisqu'un point de profondeur i_3 influence des points sur les plans de coupe d'indices $i_3 - l$ à $i_3 + l$ et, la taille de ces plans de coupe est n_1n_2.

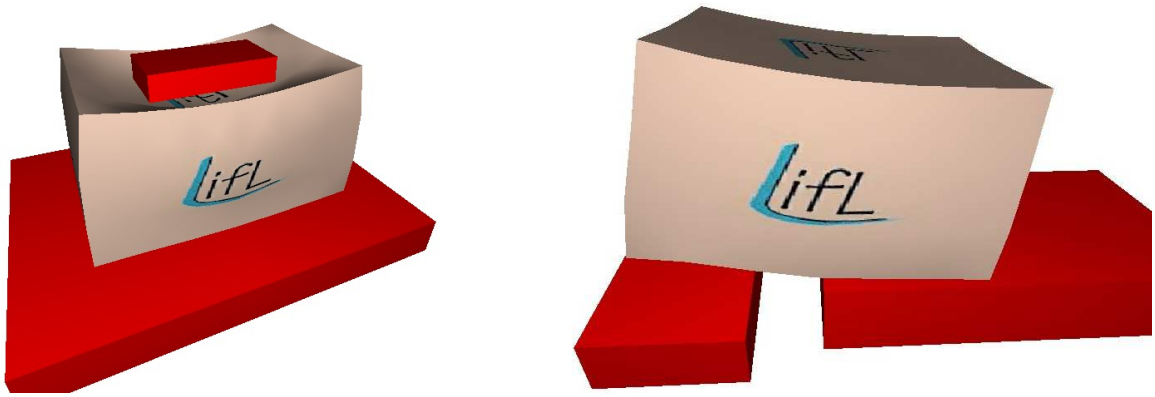
Résultats

Le rendu d'une spline 3D consiste à tracer les six faces du paralélléppipède rectangle déformable. Ainsi, on reprend la technique de rendu d'une spline 2D que l'on applique ici sur les six faces du volume déformable.

Le premier résultat présenté sur le schéma (a) de la figure (2.29) est une simulation d'une spline (catmull-rom) 3D de dimensions $(5 \times 4 \times 4)$ de masse 5 kg, de coefficient d'amortissement 0.01. La structure énergétique du modèle est constituée de ressorts d'élongation de raideur 8, de ressorts de flexion de raideur 4 et de ressorts de torsion de raideur 4. Dans cet exemple, le modèle est posé sur des objets fixes dans l'espace et il interagit avec un pavé tombant dessus. La figure montre bien les déformations du modèle : le pavé s'enfonce légèrement dans le volume, provoquant un gonflement du modèle sur les autres bords.

Le second résultat présenté sur le schéma (b) de la figure (2.29) est la simulation d'une spline (catmull-rom) 3D possédant les mêmes caractéristiques physiques sauf les raideurs des ressorts, qui sont maintenant de 10 en élongation, 8 en flexion et 6 en torsion. Dans cet exemple, le modèle tombe sur des objets fixes dans l'espace et rebondit sur eux. La configuration montrée est le moment où la spline 3D est complètement aplatie sur les deux objets et s'apprête à rebondir. Les objets fixes ne sont pas alignés, ce qui entraîne le fait que la spline 3D prend une configuration déformée pour épouser les formes des deux objets.

Les temps de calcul d'une itération du système mécanique dans les deux exemples présentés sont à peu près les mêmes, environ 50 millisecondes.



(a) (b)

FIG. 2.29 – Simulation d'une spline 3D

Comme pour le cas 2D, la simulation d'une spline 3D possède la même complexité que les splines 1D et 2D. A ceci près que le nombre de degrés de liberté est beaucoup plus important (n^3 par rapport à une spline 1D). De plus, le volume spline déformable demande l'utilisation de trois types de ressorts distribués dans tout le volume. De ce fait, l'évaluation des déformations est considérablement alourdie. Enfin, en ce qui concerne la gestion des collisions, le modèle à base de sphères est renseigné par un échantillonage

paramétrique homogène dans le volume en utilisant un rayon constant pour les sphères. Cependant, il existe des techniques[BO04] d'approximation en sphères d'un volume quelconque permettant d'obtenir des résultats plus précis, en minimisant le nombre de sphères utilisées. Il s'agit de placer des sphères de rayon variable sur l'axe median de l'objet en commencant par celles de rayon plus important pour finir avec les sphères plus petites qui donnent du détail à l'objet.

2.7 Conclusion

Ce chapitre reprend les bases des splines dynamiques et propose d'appliquer ces principes dans le cadre d'une simulation temps-réel. Des énergies de déformation discrètes sont détaillées. De même, une énergie d'élongation proposée par Olivier Nocent est décrite et une discussion est menée sur son utilisation en temps-réel. Nous proposons une énergie de flexion continue qui est corrélé avec l'élongation du modèle. Cette énergie est donc de flexions mais aussi d'élongations. Quelques tests et résultats sont présentés pour montrer le caractère temps-réel du modèle et comparer les différentes énergies proposées ou détaillées. Enfin, le chapitre décrit l'extension du modèle aux objets 2D et 3D.

L'une des perspectives de ces travaux est de compléter le modèle énergétique de la spline dynamique par une énergie continue de torsion. De cette manière, le modèle intégrerait l'ensemble des énergies de déformation d'une courbe et pourrait donc s'adapter à tous les matériaux.

Pai[Pai02] propose d'utiliser une énergie de déformation qui se base directement sur un repère local au modèle, différent du repère de Frenet, qui se déplace le long de la courbe. Ainsi, le modèle intègre directement les trois types de déformation possibles, à savoir l'élongation, la flexion et la torsion.

Terzopoulos et al.[TPBF87] proposent un modèle d'énergie continue de torsion, mais les auteurs ne précisent pas comment ils déterminent ce terme relatif à la torsion. Cependant, les auteurs établissent l'ensemble de leurs calculs sur un modèle paramétrique, on peut donc en conclure que la torsion doit être relative à la géométrie et non un paramètre physique.

Il est tout à fait envisageable d'introduire une énergie continue de torsion sur le modèle de spline 1D présenté dans ce chapitre. Pour cela, nous étudions une extension de la spline dans un espace à quatre dimensions où chaque point de contrôle est augmenté par une donnée scalaire qui informe sur la torsion locale du modèle. Ainsi, en utilisant les fonctions de base des splines, il est possible de définir un paramètre de torsion de manière continue sur la courbe. De plus, ce paramètre est indépendant du paramètre de torsion intrinsèque au modèle paramétrique (cf. annexe (D)). Pour introduire le phénomène de torsion lors de la manipulation, il faut pouvoir définir les interactions extérieures sur un volume englobant de la courbe. Si l'on prend le modèle de collision de la plate-forme de simulation, ce volume englobant est formé par une succession de sphères centrées sur la spline. Dès lors qu'une action est détectée sur l'une de ces sphères, il suffit de gérer cette interaction sur l'objet rigide que représente la section du modèle à cet endroit. On introduit donc une force sur le point de la spline (comme cela était déjà fait) et, de plus, on introduit un couple qui va introduire une torsion à cette abscisse paramétrique. A l'aide de l'équation de la spline, on reporte ce paramètre sur l'ensemble des paramètres de torsion voisins. Ainsi, on a bien introduit une torsion locale due à une interaction du modèle avec l'extérieur. De ces paramètres de torsion, il suffit de déterminer une énergie de torsion qu'il faut alors introduire dans les équations de Lagrange par évaluation de leurs variations vis-à-vis des degrés de liberté. Cette énergie de torsion continue peut très bien prendre la forme définie par Pai[Pai02] ou tout autre forme. Le calcul des variations peut, selon la forme de l'énergie, être effectué formellement ou numériquement par différences finies.

Le fait que la torsion soit définie par un paramètre indépendant des degrés de liberté originaux suppose que l'énergie de torsion établisse les dépendances entre ce paramètre et les points de contrôle de la spline. Sinon, la torsion reste un phénomène à part qui n'influencera jamais la configuration spatiale de la courbe. Le comportement du modèle dépend donc entièrement de l'énergie de torsion qui sera choisie.

De cette manière, on aurait bien une torsion du modèle spline induite uniquement par les interactions extérieures au modèle. Par exemple, si l'utilisateur a la possibilité de saisir la courbe et d'imprégner un

mouvement circulaire autour de l'axe, il peut alors manipuler la torsion du modèle directement.

Le modèle proposé permet de simuler en temps réel des objets déformables 1D. Cependant son utilisation reste limitée à des applications où le modèle est complètement libre dans l'environnement de simulation (comme la simulation d'intestin grêle dans la cavité abdominale). En réalité, les objets sont très souvent contraints par des contacts, des extrémités maintenues, des zones de mouvements limitées... Il apparaît donc clairement que le modèle proposé n'est qu'une première version qui doit être étendue pour permettre d'intégrer des contraintes.

Chapitre 3

Modèle contraint

Sommaire

Dans la plupart des manipulations, les objets sont contraints d'une manière ou d'une autre. Les contacts ou collisions sont les contraintes les plus souvent rencontrées en simulation et font généralement l'objet d'une attention toute particulière.

Mis à part les collisions, il arrive assez souvent qu'un objet soit contraint de manière définitive ou temporaire par un point particulier de sa surface. Par exemple, le fil du téléphone est fixé par l'une de ses extrémités à une prise téléphonique. On trouve également dans la réalité des articulations d'objets de différentes nature comme le fil déformable du téléphone qui est fixé à l'objet rigide qu'est le combiné, ou encore l'opération de suture qui impose des liens dynamiques. Enfin, il existe dans la réalité un type de contraintes particulier rarement pris en compte dans les travaux de simulation, il s'agit de toutes les contraintes dues à un contact avec glissement sur les fils. On trouve par exemple le système mécanique d'une corde sur une poulie ou d'un fil de couture glissant au travers un tissu ou encore d'un lacet passant de part et d'autre d'un trou d'une chaussure.

Ce chapitre présente les diverses propositions d'extension du modèle spline dynamique pour la prise en compte de contraintes bilatérales (cf. section (1.3)).

Les contraintes sont ici scindées en deux groupes, le premier concerne les contraintes relatives à la spline dynamique seule et le second regroupe les contraintes liant la spline dynamique à un autre objet dynamique. Nous exposons ensuite une proposition d'extension du modèle pour prendre en considération une nouvelle classe de contraintes appelées *contraintes glissantes*. Ensuite, quelques tests et applications sont détaillées pour montrer l'efficacité des méthodes mises en place. Avant de conclure, une discussion est menée sur l'utilisation de ces contraintes dans les modèles splines de dimension supérieure.

3.1 Contraintes simples

Imposer des contraintes à un corps demande de faire un choix quant à la méthode utilisée. Un grand nombre de ces méthodes ont été exposées dans la section (1.3). Notre choix s'est porté sur la méthode des multiplicateurs de Lagrange munie d'un schéma de Baumgarte pour sa simplicité et son compromis fiabilité/stabilité, mais également par le fait qu'elle s'insère directement dans les équations de la dynamique.

3.1.1 Extension du modèle spline dynamique

L'extension du modèle de spline dynamique (cf. équation (2.5), l'ensemble des notations employées dans le chapitre précédent sont conservées) aux multiplicateurs de Lagrange s'effectue en considérant le Lagrangien comme fonction objective de la minimisation[Noc99], cela apporte les équations :

$$\left\{\begin{array}{lcll} \dfrac{d}{dt}\left(\dfrac{\partial K}{\partial \dot{q}_i^\alpha}\right)+\dfrac{\partial K}{\partial q_i^\alpha} & = & Q_i^\alpha-\dfrac{\partial E}{\partial q_i^\alpha}+\displaystyle\sum_{j=1}^{c}\lambda_j L_{ji} & , \quad \forall i\in\{1..n\} \text{ et } \forall\alpha\in\{x,y,z\} \\ g_j(\mathbf{q}(t),\dot{\mathbf{q}}(t),t) & = & 0 & , \quad \forall j\in\{1..c\} \end{array}\right. \tag{3.1}$$

où g_j sont les c équations bilatérales de contrainte issues du schéma de Baumgarte, λ_j sont les c multiplicateurs de Lagrange (relatifs à l'intensité de la violation de la contrainte dont ils sont respectivement en charge), L est la matrice (de taille $c\times 3n$) qui lie chaque équation scalaire de contrainte avec l'ensemble des degrés de liberté du système physique.

> **Définition :** On appelle équation atomique, une équation de forme scalaire. Par extension, on appelle contrainte atomique, une contrainte dont l'équation est atomique.
> De ce fait, les équations de contraintes vectorielles se décomposent en plusieurs équations de contraintes atomique. Une contrainte atomique correspond à une ligne de la matrice L alors qu'une contrainte vectorielle en regroupe plusieurs. Ainsi, il faut un multiplicateur de Lagrange λ_j par contrainte atomique.

Le schéma de Baumgarte se base sur deux coefficients permettant de contrôler la dynamique de la contrainte. Si on note C une contrainte holonome de type scalaire ou vectorielle (scalaire pour l'exemple), un schéma de Baumgarte possible est alors

$$\ddot{C} + 2a\dot{C} + a^2 C = 0$$

Les coefficients sont ainsi choisis de manière à obtenir la réalisation de la contrainte la plus rapide possible sans introduire d'oscillation autour de la solution (cf. section 1.3.4 pour plus de détails). Afin d'assurer une certaine réalisation de la contrainte dans le système dynamique, on lie le coefficient a au pas de temps δt de l'intégration numérique par la relation $a = \frac{1}{\delta t}$[BB88]. Toutes les équations de contraintes holonome subissent donc le schéma de Baumgarte suivant :

$$\ddot{C} + \frac{2}{\delta t}\dot{C} + \frac{1}{\delta t^2} C = 0 \tag{3.2}$$

Le système d'équations (3.1) régissant la dynamique de la spline se met sous forme matricielle :

$$\begin{pmatrix} \mathcal{M} & L^T \\ L & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ -\boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{E} \end{pmatrix} \tag{3.3}$$

où $\mathbf{E}$ est le vecteur mesurant l'intensité de la violation de chacune des contraintes.

Remarques sur la matrice L :

Une ligne de la matrice L traite une contrainte atomique.

- Si cette contrainte est holonome, elle ne dépend pas des vitesses des degrés de liberté, donc les seuls paramètres dépendants du temps sont les degrés de liberté :

  $$C(\mathbf{q}(t), t)$$

  De plus, la contrainte s'intègre dans le système (3.3) à l'aide du schéma de Baumgarte (3.2). Or :

  $$\dot{C}(\mathbf{q}(t), t) = \frac{dC}{dt}(\mathbf{q}(t), t) = \frac{\partial C}{\partial \mathbf{q}}(\mathbf{q}(t), t).\frac{d\mathbf{q}}{dt} = \dot{\mathbf{q}}(t).\frac{\partial C}{\partial \mathbf{q}}(\mathbf{q}(t), t)$$

  et

  $$\ddot{C}(\mathbf{q}(t), t) = \frac{d}{dt}\left(\dot{\mathbf{q}}(t).\frac{\partial C}{\partial \mathbf{q}}(\mathbf{q}(t), t)\right) = \ddot{\mathbf{q}}(t).\frac{\partial C}{\partial \mathbf{q}}(\mathbf{q}(t), t) + \dot{\mathbf{q}}(t).\frac{d}{dt}\left(\frac{\partial C}{\partial \mathbf{q}}(\mathbf{q}(t), t)\right)$$

  donc, dans l'équation de Baumgarte, les coefficients des accélérations des degrés de liberté sont les variations de la contrainte par rapport aux degrés de liberté.
  Donc, pour une contrainte C de type holonome, la ligne i (de la matrice L) correspondant à la contrainte vaut :

  $$L_i = \frac{\partial C}{\partial \mathbf{q}}(\mathbf{q}(t), t)$$

- Si la contrainte est cinématique, elle prend la forme :

  $$C(\mathbf{q}(t), \dot{\mathbf{q}}(t), t)$$

  Si l'on emploie une combinaison linaire de l'équation de contrainte et de sa dérivée pour former l'équation finale à insérer dans le système, cette équation peut se mettre sous la forme :

  $$\dot{C}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) + kC(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) = 0 \tag{3.4}$$

  avec

  $$\dot{C}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) = \frac{dC}{dt}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) = \dot{\mathbf{q}}(t).\frac{\partial C}{\partial \mathbf{q}}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) + \ddot{\mathbf{q}}(t).\frac{\partial C}{\partial \dot{\mathbf{q}}}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t)$$

  donc, l'équation (3.4) qui est introduite dans le système dynamique définit la ligne i (de la matrice L) qui lui est dédiée par :

  $$L_i = \frac{\partial C}{\partial \dot{\mathbf{q}}}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t)$$

La résolution d'un tel système d'équations peut s'opérer en deux phases. D'abord calculer les accélérations du système non contraint (accélérations dites de tendance : $\mathbf{A_t}$) puis, à l'aide des multiplicateurs de Lagrange ($\boldsymbol{\lambda}$), calculer les accélérations de corrections ($\mathbf{A_c}$). Ainsi, on obtient les accélérations finales en sommant ces deux termes d'accélération :

$$\mathbf{A} = \mathbf{A_t} + \mathbf{A_c} \tag{3.5}$$

Cette technique de décomposition des accélérations est décrite par Rémion[Rém00].

En opérant cette décomposition, le système d'équations se scinde en :

$$\left\{\begin{array}{rcl} \mathcal{M}.\mathbf{A_t} & = & \mathbf{B} \\ \mathcal{M}.\mathbf{A_c} & = & L^T.\boldsymbol{\lambda} \\ L.(\mathbf{A_t}+\mathbf{A_c}) & = & \mathbf{E} \end{array}\right. \Leftrightarrow \left\{\begin{array}{rcll} \mathcal{M}.\mathbf{A_t} & = & \mathbf{B} & (1) \\ \mathbf{A_c} & = & \mathcal{M}^{-1}.L^T.\boldsymbol{\lambda} & (2) \\ \left(L.\mathcal{M}^{-1}.L^T\right).\boldsymbol{\lambda} & = & \mathbf{E}-L.\mathbf{A_t} & (3) \end{array}\right. \tag{3.6}$$

Afin de résoudre les équations dynamiques du modèle contraint, il suffit d'effectuer les opérations suivantes sur le système d'équations (3.6) :

- Résoudre le système non contraint défini par l'équation (3.6(1)). Pour cela, on applique la technique de résolution décrite dans le chapitre précédent en section (2.2.3).
- Evaluer l'intensité des efforts nécessaires à la réalisation des contraintes en calculant les λ à l'aide de l'équation (3.6(3)). Pour cela, il suffit de résoudre le système

  $$\left(L.\mathcal{M}^{-1}.L^T\right).\boldsymbol{\lambda} = \mathbf{E} - L.\mathbf{A_t}$$

  On calcule donc un vecteur $\mathbf{O} = \mathbf{E} - L.\mathbf{A_t}$, la matrice $\mathcal{M}^{-1}.L^T$ (en signalant que $\mathcal{M}$ est constante donc précalculé et son inverse $\mathcal{M}^{-1}$ également) puis la matrice $R = L.\mathcal{M}^{-1}.L^T$. Ensuite, le système d'équations

  $$R.\boldsymbol{\lambda} = \mathbf{O}$$

  est résolu à l'aide de la méthode de Gauss ou d'une méthode itérative de type gradient conjugué. En termes de complexité temporelle, la construction du vecteur $\mathbf{O}$ est en $\mathcal{O}(cn)$. La construction de la matrice intermédiaire $\mathcal{M}^{-1}.L^T$ est optimisée puisque même si aucune supposition n'est faite sur les contraintes (donc sur la structure de la matrice L), la matrice $\mathcal{M}$ est diagonale par blocs où chaque bloc est identique à une matrice M de structure bande. Donc l'inverse de $\mathcal{M}$ est une matrice diagonale par blocs où chaque bloc est identique à la matrice M^{-1} (qui est précalculée). Malheureusement, la structure bande de la matrice M est perdue lors de l'inversion pour aboutir à une matrice M^{-1} pleine. Ainsi, l'algorithme du produit de deux matrices est optimisé en :

COMPUTE$\mathcal{M}^{-1}.L^T()$

```
1  for i ← 0 to n
2  do
3      for j ← 0 to c
4      do
           M^-1.L^T[i][j]       ←  0
5          M^-1.L^T[n+i][j]     ←  0
           M^-1.L^T[2n+i][j]    ←  0
6          for k ← 0 to n
7          do
               M^-1.L^T[i][j]     ←  M^-1.L^T[i][j]     +  M^-1[i][k] * L[j][k]
8              M^-1.L^T[n+i][j]   ←  M^-1.L^T[n+i][j]   +  M^-1[i][k] * L[j][n+k]
               M^-1.L^T[2n+i][j]  ←  M^-1.L^T[2n+i][j]  +  M^-1[i][k] * L[j][2n+k]
```

De ce fait, la complexité en temps de l'algorithme est exactement de $3cn^2$ donc en $\mathcal{O}(cn^2)$. Ce qui est trois fois plus rapide que le simple produit matrice-matrice de base. Puis, vient la construction de la matrice R qui est le résultat du produit des matrices L (de taille $c \times 3n$) et $\mathcal{M}^{-1}.L^T$ (de taille $3n \times c$). Etant donné le peu de contrôle sur les structures de ces deux matrices, aucune optimisation ne peut être apportée ici. Ainsi, la complexité du calcul de R est de $\mathcal{O}(c^2n)$.

La résolution de l'équation linéaire $R.\boldsymbol{\lambda} = \mathbf{O}$ est effectuée soit avec un pivot de Gauss, soit à l'aide de la méthode itérative du bi-gradient conjugué (une étude comparative est menée dans la section (3.4.2.3). Cette étude est menée dans le cas de contraintes externes, mais les résultats sont également valides pour le cas des contraintes internes).

- Déterminer les accélérations de correction à l'aide des $\boldsymbol{\lambda}$ calculés précédemment en utilisant l'équation (3.6(2)).
  Pour cela, la matrice $\mathcal{M}^{-1}.L^T$ a déjà été calculée à l'étape précédente, il suffit donc d'effectuer le produit de cette matrice (de taille $3n \times c$) avec le vecteur des $\boldsymbol{\lambda}$. Ce calcul a donc une complexité temporelle en $\mathcal{O}(cn)$.

En résumé, la résolution du système d'équations dynamiques de l'objet contraint s'effectue avec une complexité temporelle en $\mathcal{O}(cn + cn^2 + c^2n)$. On peut donc en déduire que l'utilisation de contraintes alourdit substantiellement la résolution du sytème dynamique. Cependant, l'algorithme est valable pour toute contrainte holonome ou cinématique. On peut donc se permettre de définir des contraintes extrêmement complexes, l'algorithme les intègre dans le système avec la même facilité qu'une contrainte plus simple. Notamment, une contrainte n'est pas obligée de fixer un degré de liberté *direct* de l'objet (i.e. un q_i^α dans le cas des splines), mais peut souhaiter fixer un degré de liberté plus complexe exprimé en fonction des coordonnées généralisées (relatif à un point de la spline par exemple).

Nous présentons maintenant quelques contraintes qui ont pu être mises en place dans cette version contrainte du modèle de spline dynamique.

3.1.2 Exemples de contraintes

Nous allons maintenant détailler les contraintes utiles[LMGC02] que nous avons mis en place dans le modèle.

L'une des contraintes les plus immédiates et directes à écrire est de fixer un point d'un objet simulé, dans l'espace. Ceci permet de bloquer les trois coordonnées de ce point et donc de définir une contrainte de point fixe. On peut tout aussi bien vouloir fixer une seule ou deux coordonnées d'un point, imposant à ce point de se déplacer sur un ou deux axes du repère de l'espace. Avec une approche similaire, on peut fixer un ou deux degrés de liberté d'un objet en obligeant l'un de ses points à se mouvoir sur une droite ou un plan quelconque de l'espace.

3.1.2.1 Contrainte de point fixe

Une contrainte désirable est le fait de fixer dans l'espace un point de l'objet simulé. Cependant, comme les points de contrôle (degrés de liberté) de la spline ne sont pas forcément sur l'objet, la contrainte ne se contente pas de fixer un degré de liberté q_i^α, mais doit fixer un point de la spline. Donc une combinaison linéaire des degrés de liberté. De plus, on remarque qu'une telle contrainte impose de fixer les trois coordonnées d'un point, elle se décline donc en trois équations *atomiques*.

Ainsi, pour fixer le point de la courbe d'abscisse paramétrique s_0 au point $\mathbf{A}$ de l'espace, la contrainte holonome s'écrit sous forme vectorielle :

$$\mathbf{C}_{\mathbf{point\_fixe}}(s_0, t, \mathbf{A}) = \mathbf{P}(s_0, t) - \mathbf{A}$$

Si l'équation de contrainte est vérifiée, on a alors $\mathbf{C}_{\mathbf{point\_fixe}}(s_0, t, \mathbf{A}) = \mathbf{P}(s_0, t) - \mathbf{A} = 0$ et donc le point $\mathbf{P}(s_0, t)$ est bien en $\mathbf{A}$.

Afin de déterminer l'équation de contrainte $\mathbf{g}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t)$ à intégrer dans le système dynamique (équation 3.1), on utilise le schéma de Baumgarte de l'équation (3.2) :

$$\begin{aligned} \mathbf{g}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) &= \ddot{\mathbf{C}}_{point\_fixe}(s_0, t, \mathbf{A}) + \frac{2}{\delta t}\dot{\mathbf{C}}_{point\_fixe}(s_0, t, \mathbf{A}) + \frac{1}{\delta t^2}\mathbf{C}_{point\_fixe}(s_0, t, \mathbf{A}) &= 0 \\ &= \ddot{\mathbf{P}}(s_0, t) + \frac{2}{\delta t}\dot{\mathbf{P}}(s_0, t) + \frac{1}{\delta t^2}(\mathbf{P}(s_0, t) - \mathbf{A}) &= 0 \end{aligned}$$

En substituant la position $\mathbf{P}$ et la vitesse $\dot{\mathbf{P}}$ du point contraint par leur expression respective (cf. équations

(2.3) et (2.4)) et en faisant de même pour l'accélération $\ddot{\mathbf{P}}$, on aboutit à l'équation :

$$\mathbf{g}(\mathbf{q}(t),\dot{\mathbf{q}}(t),t) = \sum_{i=1}^{n}\ddot{\mathbf{q}}_\mathbf{i}(t)b_i(s) + \frac{2}{\delta t}\sum_{i=1}^{n}\dot{\mathbf{q}}_\mathbf{i}(t)b_i(s) + \frac{1}{\delta t^2}\left(\sum_{i=1}^{n}\mathbf{q}_\mathbf{i}(t)b_i(s) - \mathbf{A}\right) = 0 \tag{3.7}$$

Afin de renseigner la matrice L des contraintes, il faut séparer les termes relatifs aux accélérations des degrés de liberté des autres termes. Ainsi, l'équation (3.7) s'organise comme :

$$\sum_{i=1}^{n}\ddot{\mathbf{q}}_\mathbf{i}(t)b_i(s) = -\frac{2}{\delta t}\sum_{i=1}^{n}\dot{\mathbf{q}}_\mathbf{i}(t)b_i(s) - \frac{1}{\delta t^2}\left(\sum_{i=1}^{n}\mathbf{q}_\mathbf{i}(t)b_i(s) - \mathbf{A}\right) \tag{3.8}$$

La partie gauche de l'équation (3.8) permet de calculer les coefficients des trois lignes de la matrice L relatives à cette contrainte (qui se décline en trois contraintes *atomiques*). En supposant que les lignes de la matrice L relatives à cette contrainte sont i (pour l'axe des x), j (pour l'axe des y) et k (pour l'axe des z), on détermine alors leurs coefficients par :

$$L_{il} = L_{j(l+n)} = L_{k(l+2n)} = b_l(s)$$

on peut remarquer au passage que la propriété de localité des splines permet de ne calculer qu'une partie de ses coefficients.

Voici la structure de la matrice L résultante :

$$L = \begin{pmatrix} \overbrace{\begin{matrix}\dots&\dots&\dots&\dots&\dots&\dots&\dots\\ \dots&0&a&b&c&0&\dots\\ \dots&0&0&0&0&0&\dots\\ \dots&0&0&0&0&0&\dots\\ \dots&\dots&\dots&\dots&\dots&\dots&\dots\end{matrix}}^{x} & \overbrace{\begin{matrix}\dots&\dots&\dots&\dots&\dots&\dots&\dots\\ \dots&0&0&0&0&0&\dots\\ \dots&0&a&b&c&0&\dots\\ \dots&0&0&0&0&0&\dots\\ \dots&\dots&\dots&\dots&\dots&\dots&\dots\end{matrix}}^{y} & \overbrace{\begin{matrix}\dots&\dots&\dots&\dots&\dots&\dots&\dots\\ \dots&0&0&0&0&0&\dots\\ \dots&0&0&0&0&0&\dots\\ \dots&0&a&b&c&0&\dots\\ \dots&\dots&\dots&\dots&\dots&\dots&\dots\end{matrix}}^{z} \end{pmatrix} \begin{matrix}\\ \longrightarrow i\\ \longrightarrow j\\ \longrightarrow k\\ \\ \end{matrix}$$

On retrouve les trois lignes i, j et k de contrainte et l'organisation de la matrice des masses généralisées avec les coordonnées x en premier, suivi des coordonnées y et enfin les coordonnées z. A noter la présence des mêmes coefficients pour chaque coordonnée.

La partie droite de l'équation (3.8) permet d'évaluer la violation de la contrainte vis-à-vis des positions et des vitesses. Ainsi, les corrections engendrées par les multiplicateurs de Lagrange prennent en compte la dynamique de l'objet. Cette partie de l'équation permet donc de calculer le vecteur $\mathbf{E}$ dans l'équation matricielle (3.3) de la manière suivante :

$$\mathbf{E} = \begin{pmatrix} \dots \\ -\frac{2}{\delta t}\sum_{i=1}^{n}\dot{q}_i^x(t)b_i(s) - \frac{1}{\delta t^2}\left(\sum_{i=1}^{n}q_i^x(t)b_i(s) - A^x\right) \\ -\frac{2}{\delta t}\sum_{i=1}^{n}\dot{q}_i^y(t)b_i(s) - \frac{1}{\delta t^2}\left(\sum_{i=1}^{n}q_i^y(t)b_i(s) - A^y\right) \\ -\frac{2}{\delta t}\sum_{i=1}^{n}\dot{q}_i^z(t)b_i(s) - \frac{1}{\delta t^2}\left(\sum_{i=1}^{n}q_i^z(t)b_i(s) - A^z\right) \\ \dots \end{pmatrix} \begin{matrix}\\ \longrightarrow i\\ \longrightarrow j\\ \longrightarrow k\\ \\ \end{matrix}$$

3.1.2.2 Contraindre un point sur un axe

Pour contraindre un point d'abscisse s_0 de la spline sur une droite passant par $\mathbf{A}$ de vecteur directeur unitaire $\mathbf{u}$, on détermine deux vecteurs unitaires $\mathbf{n_1}$ et $\mathbf{n_2}$ formant avec $\mathbf{u}$ une base orthonormée $(\mathbf{u},\mathbf{n_1},\mathbf{n_2})$ (cf. figure (3.1)). Le calcul des deux vecteurs n_1 et n_2 est détaillé dans l'annexe (F).

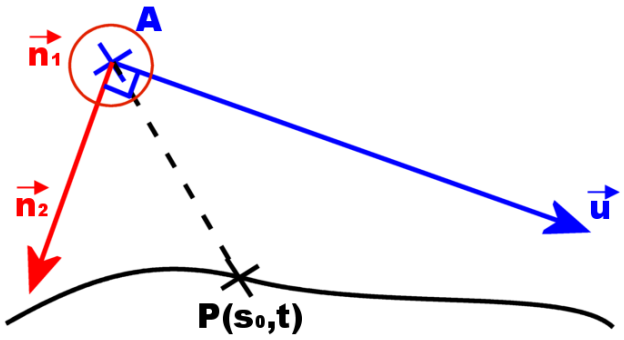


FIG. 3.1 – Schéma de fonctionnement pour contraindre un point sur un axe

Le point $\mathbf{P(s_0, t)}$ est sur la droite $(\mathbf{A}, \mathbf{u})$ si et seulement si le vecteur $\mathbf{AP(s_0, t)}$ est orthogonal aux vecteurs $\mathbf{n_1}$ et $\mathbf{n_2}$. Ainsi, la contrainte s'exprime sous forme de deux contraintes atomiques d'orthogonalité :

$$\begin{cases} C_1(s_0, t, \mathbf{A}) = \mathbf{AP}(s_0, t).\mathbf{n_1} &= 0 \\ C_2(s_0, t, \mathbf{A}) = \mathbf{AP}(s_0, t).\mathbf{n_2} &= 0 \end{cases}$$

Ces équations expriment l'appartenance du point $\mathbf{P(s_0, t)}$ à deux plans passant par $\mathbf{A}$ et de vecteurs normaux $\mathbf{n_1}$ et $\mathbf{n_2}$. Chaque équation d'orthogonalité supprime donc un degré de liberté de la spline.

Comme les deux équations de contraintes sont similaires, voyons de manière générale, comment introduire une contrainte d'appartencance à un plan dans le système d'équations dynamiques.

3.1.2.3 Contraindre un point dans un plan

Pour contraindre un point d'abscisse paramétrique s_0 de la spline à être sur un plan défini dans l'espace par un point $\mathbf{A}$ (appartenant au plan) et un vecteur normal $\mathbf{n}$, il suffit que le vecteur $\mathbf{AP}(s_0, t)$ soit orthogonal au vecteur $\mathbf{n}$. On aboutit donc à une contrainte atomique d'orthogonalité.

Soit $C(s_0, t, \mathbf{A})$ cette contrainte atomique d'orthogonalité, elle prend la forme :

$$C(s_0, t, \mathbf{A}) = \mathbf{AP}(s_0, t).\mathbf{n} = 0$$

Cette contrainte impose bien que le point d'abscisse paramétrique s_0 de la spline soit dans le plan passant par $\mathbf{A}$ de vecteur normal $\mathbf{n}$. Donc, en combinant deux équations de ce type avec deux vecteurs $\mathbf{n}$ non colinéaires ($\mathbf{n_1}$ et $\mathbf{n_2}$), on contraint le point $\mathbf{P(s_0, t)}$ de la courbe à être sur l'intersection des deux plans, donc sur la droite passant par $\mathbf{A}$ de vecteur directeur $(\mathbf{n_1} \wedge \mathbf{n_2})$.

Sachant que le vecteur $\mathbf{n}$ est constant ainsi que le point $\mathbf{A}$, l'application du schéma de Baumgarte (3.2) à cette équation atomique aboutit à :

$$\begin{aligned} g(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) &= \ddot{C}(s_0, t, \mathbf{A}) + \frac{2}{\delta t}\dot{C}(s_0, t, \mathbf{A}) + \frac{1}{\delta t^2}C(s_0, t, \mathbf{A}) &= 0 \\ &= \ddot{\mathbf{P}}(s_0, t).\mathbf{n} + \frac{2}{\delta t}\dot{\mathbf{P}}(s_0, t).\mathbf{n} + \frac{1}{\delta t^2}\mathbf{AP}(s_0, t).\mathbf{n} &= 0 \end{aligned}$$

On utilise alors cette équation pour renseigner les coefficients de la matrice L et du vecteur $\mathbf{E}$ en regroupant les termes relatifs aux accélérations des degrés de liberté d'un coté de l'équation :

$$\begin{aligned} & g(\mathbf{q}(t), \dot{\mathbf{q}}(t), t) &= 0 \\ \Leftrightarrow \quad & \left(\sum_{i=1}^{n} \ddot{\mathbf{q}}_\mathbf{i}(t) b_i(s_0)\right).\mathbf{n} &= -\frac{2}{\delta t}\left(\sum_{i=1}^{n} \dot{\mathbf{q}}_\mathbf{i}(t) b_i(s_0)\right).\mathbf{n} - \frac{1}{\delta t^2}\left(\sum_{i=1}^{n} \mathbf{q}_\mathbf{i}(t) b_i(s_0) - \mathbf{A}\right).\mathbf{n} \end{aligned} \tag{3.9}$$

La partie droite de l'équation (3.9) permet de déterminer l'intensité de la violation de la contrainte. Elle permet donc de renseigner le coefficient du vecteur $\mathbf{E}$ relatif à la contrainte. La partie gauche de

l'équation (3.9) permet, quant à elle, de former la ligne k de la matrice L relative à la contrainte. Pour cela, developpons cette équation pour faire apparaître les accélérations des degrés de liberté :

$$\left(\sum_{i=1}^{n} \ddot{\mathbf{q}}_\mathbf{i}(t) b_i(s_0)\right).\mathbf{n} = \sum_{\alpha\in\{x,y,z\}} \sum_{i=1}^{n} \ddot{q}_i{}^{\alpha}(t) b_i(s_0) n^{\alpha}$$

Ainsi, la ligne k de la matrice L se remplit comme suit :

$$L = \left(\begin{array}{ccccccc|ccccccc|ccccccc} \overbrace{\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots}^{x} & \overbrace{\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots}^{y} & \overbrace{\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots}^{z} \\ \cdots & 0 & a^x & b^x & c^x & 0 & \cdots & \cdots & 0 & a^y & b^y & c^y & 0 & \cdots & \cdots & 0 & a^z & b^z & c^z & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{array}\right) \longrightarrow k$$

avec j l'indice de la colonne du coefficient a^x, on a

$$\begin{array}{lll} a^x = b_j(s_0)n^x & b^x = b_{j+1}(s_0)n^x & c^x = b_{j+2}(s_0)n^x \\ a^y = b_j(s_0)n^y & b^y = b_{j+1}(s_0)n^y & c^y = b_{j+2}(s_0)n^y \\ a^z = b_j(s_0)n^z & b^z = b_{j+1}(s_0)n^z & c^z = b_{j+2}(s_0)n^z \end{array}$$

Les zéros figurant autour des triplets de valeurs, dans la matrice L, témoignent de la propriété de localité des splines. Sur cet exemple, seuls trois coefficients sont non nuls.

3.1.2.4 Contraintes de tangente et courbure

Si le modèle spline est suffisament continu, on peut définir des contraintes sur les vecteurs tangente ou courbure locale. Pour une contrainte de tangente fixe au point d'abscisse s_0 par exemple, l'équation de contrainte s'écrit :

$$\mathbf{C}(s_0,t) = \mathbf{T}(s_0,t) - \mathbf{T_0} = \sum_{i=1}^{n} \mathbf{q}_\mathbf{i}(t) b_i'(s_0) - \mathbf{T_0}$$

où $\mathbf{T_0}$ est le vecteur tangente désiré.

3.1.2.5 Autre type de contraintes

Toute équation holonome ou cinématique définie sur les coordonnées généralisées et leur dérivées premières temporelles sont potentiellement des contraintes. On peut donc imaginer un grand nombre de contraintes mais, toutes ces contraintes n'ont pas forcément de réalité physique. Le simple fait de bloquer une coordonnée généralisée permet de fixer un point de contrôle. Ceci est donc une contrainte purement géométrique qui n'a de sens physique que si les points de contrôle sont confondus avec le modèle spline. Dans le cas d'une B-spline, par exemple, une telle contrainte apporterait un comportement incompréhensible puisque cela contraint le modèle à un endroit de l'espace où il ne passe pas. On peut également imaginer introduire une contrainte de distance. En posant l'égalité des carrés des distances, l'application du schéma de Baumgarte apporte bien une contrainte linéaire sur les accélérations des degrés de liberté et peut donc être utilisé directement dans le système d'équations.

Les contraintes détaillées dans cette section permettent de fixer des invariants sur le modèle relativement à l'environnement ou à lui même. Autrement dit, chaque contrainte porte sur un vecteur fixe ou une donnée de référence constante. Il peut être utile de pouvoir définir cette référence non plus comme constante mais liée à un objet mobile. De cette manière, il est possible de définir des liaisons entre objets et donc de créer des articulations. C'est le problème étudié dans la section suivante.

3.2 Contraintes inter-objets

On appelle contrainte inter-objets une contrainte qui lie deux modèles simulés. Ainsi, les collisions font partie intégrante de cette catégorie de contrainte mais, sont gérées par le moteur de simulation à

l'aide d'une méthode à pénalité (voir section sur les collisions (2.5.1)). On exclut donc les collisions de notre étude. Il s'agit donc d'assurer des liaisons entre divers objets, d'une même simulation, qui peuvent être de nature très différentes.

Dans [LF04], nous avons proposé un cadre logiciel pour résoudre ce type de contrainte. Pour cela, nous ne faisons aucune supposition sur les objets. Ils peuvent être simulés dynamiquement, cinématiquement, statiquement ou fixes. Ils peuvent être animés par n'importe quelle méthode (méthode éléments finis, masses-ressorts, équations de Lagrange...). Les objets peuvent être rigides ou déformables, discrets ou continus.

Divers travaux proposent également un modèle unifié pour résoudre des contraintes. Par exemple, Rémion et al.[RNN01] simulent tous les objets de la scène à l'aide des équations physiques de Lagrange puis imposent des contraintes avec la méthode des multiplicateurs de Lagrange sur ces divers objets. Ou encore, Faure[Fau98] propose une simulation d'objets rigides articulés dont les liaisons sont assurées par une méthode post-stabilisation. Dans ces deux propositions, le formalisme physique utilisé pour animer les objets est imposé.

Baraff et Witkin[BW97] proposent une technique permettant de contraindre des objets simulés dans des simulations différentes. Pour cela, les auteurs proposent une méthode itérative basée sur les multiplicateurs de Lagrange. Pour chaque couple d'objets contraints, l'un des objets est fixé pendant que l'autre résout la contrainte, puis les rôles sont inversés. Ce procédé est appliqué itérativement jusqu'à la résolution complète des contraintes. Cette proposition prend en compte des objets simulés avec différentes lois physiques mais, étant donné la restriction des accès aux informations, le processus est itératif.

Baraff[Bar96] propose d'utiliser les multiplicateurs de Lagrange pour contraindre divers objets (pas forcément rigides et sans aucune supposition sur les lois physiques employées) avec une complexité des calculs linéaires en temps. Cependant, l'auteur teste l'algorithme présenté sur des simulations de scène comportant uniquement des objets rigides.

Jansson et Vergeest [JV03] proposent une solution pour simuler des modèles rigides et déformables tout en donnant la possibilité à un objet de changer d'état en cours de simulation. Pour cela, tout objet possède une structure masses-ressorts volumique et une structure rigide. A tout moment, la simulation est capable de passer d'une représentation à l'autre. Cependant, le modèle présenté ne permet de simuler que des objets rigides ou déformables discrets. Tout objet continu (comme la spline dynamique) ne peut être simulé directement par cette technique, il doit forcément subir une discrétisation lui faisant perdre ses propriétés intrinsèques (continuité, localité...). Cette proposition permet de lier des objets entre eux à l'aide d'une interface commune permettant de définir des connexions gérées par des ressorts (relatifs à une méthode à pénalité). Les méthodes à pénalité n'étant pas robustes, les connexions entre objets ne sont assurées que si les raideurs des ressorts sont élevées. Ce qui pose des problèmes d'intégration numérique.

Une première section présente notre proposition d'architecture pour simuler des objets quelconques liés (ou non) les uns aux autres. Pour cela, la méthode définit une interface commune et, quelques exemples de modèle implémentant cette interface sont présentés dans une seconde section.

3.2.1 Aspect mécanique de notre proposition

Notre proposition consiste à regrouper tous les objets contraints dans un même système matériel. Ainsi, il est possible d'employer une technique non itérative (au contraire de [BW97]) pour imposer des contraintes entre les divers objets.

Pour cela, on se base sur le fait que tout modèle simulé dynamiquement peut s'écrire sous la forme :

$$M\mathbf{A} = \mathbf{B}$$

où M est la matrice des masses, $\mathbf{A}$ le vecteur des accélérations et $\mathbf{B}$ le vecteur des forces internes (déformations) et externes (interactions).

En remarquant qu'on retrouve également un système d'équations linéaires pour une simulation cinématique :

$$C\mathbf{V} = \mathbf{B}$$

avec C la matrice d'amortissement de l'objet et $\mathbf{V}$ le vecteur des vitesses des degrés de liberté.

Si la simulation est statique, le système s'écrit sous la même forme sauf que les données impliquées ne sont pas les mêmes :

$$K\mathbf{U} = \mathbf{F}$$

avec K la matrice de rigidité de l'objet (relative aux propriétés de déformation du matériau), $\mathbf{U}$ le vecteur des déplacements et $\mathbf{F}$ le vecteur des forces externes (interactions) appliquées à l'objet.

On peut également définir un système d'équations linéaires pour un objet fixe :

$$I\mathbf{X} = X$$

où I est la matrice identité et $\mathbf{X}$ le vecteur des positions des points caractéristiques de l'objet fixe.

Ainsi, pour simuler tous ces objets en même temps, il suffit de regrouper les équations dans un même système matériel où les inconnues sont disparates :

$$M_g\mathbf{A_g} = \mathbf{B_g} \tag{3.10}$$

avec M_g une matrice diagonale bloc dont chaque bloc représente la matrice M, C, K ou I d'un objet considéré, $\mathbf{A_g}$ le vecteur des inconnues (accélérations, vitesses, déplacements, positions fixes) et $\mathbf{B_g}$ le vecteur relatif au bilan de force. Par exemple, un système matériel possédant n objets dynamiques $(o_1, ..., o_n)$ a la structure suivante :

$$M_g\mathbf{A_g} = \begin{pmatrix} M_{o_1} & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & M_{o_n} \end{pmatrix} \begin{pmatrix} \mathbf{A}_{o_1} \\ \vdots \\ \mathbf{A}_{o_n} \end{pmatrix} = \begin{pmatrix} \mathbf{B}_{o_1} \\ \vdots \\ \mathbf{B}_{o_n} \end{pmatrix} = \mathbf{B}_g$$

En l'abscence de contraintes liant les objets, ceux-ci évoluent indépendamment les uns des autres, leur résolution peut donc être décorrélée. Ainsi, la résolution du système d'équations (3.10) revient à demander à chaque objet de se résoudre indépendamment des autres objets.

Pour des raisons déjà évoquées dans la section (3.1), nous proposons d'utiliser la méthode des multiplicateurs de Lagrange pour introduire des liaisons entre les objets simulés.

Ainsi, le système d'équations (3.10) est étendu par les multiplicateurs de Lagrange pour aboutir au nouveau système :

$$\begin{pmatrix} M_g & L^T \\ L & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A_g} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{B_g} \\ \mathbf{E} \end{pmatrix} \tag{3.11}$$

avec L la matrice des contraintes, $\boldsymbol{\lambda}$ les multiplicateurs de Lagrange et $\mathbf{E}$ le vecteur indiquant l'intensité de la violation pour chaque contrainte.

Les équations étant posées, voyons maintenant une architecture logicielle adaptée à la simulation d'objets contraints.

3.2.2 Aspect informatique de notre proposition : architecture logicielle

Nous proposons l'architecture schématisée sur la figure (3.2) pour simuler des objets contraints entre eux.

L'*articulation* est en charge d'assurer les contraintes entre les différents objets. Pour cela, elle possède une liste des objets et des contraintes. Chaque objet répond à une certaine interface relative aux contraintes. Ainsi, l'*articulation* peut interroger les objets pour remplir ou mettre à jour ses structures de donnée. Ou encore déterminer si un objet est capable de définir une contrainte d'un certain type. Par exemple, on peut contraindre un objet rigide en rotation par rapport à un autre objet, alors qu'une

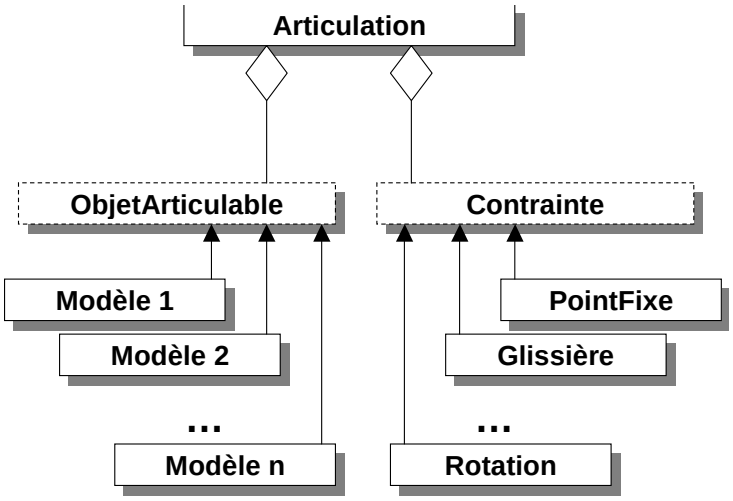


FIG. 3.2 – Architecture logicielle - Diagramme de classe

telle contrainte n'a pas vraiment de sens sur un modèle de type masses-ressorts. Dans ce cas, le modèle masses-ressorts ne renseigne pas l'ensemble des méthodes relatives à la contrainte de rotation.

Pour mieux comprendre comment la simulation prend place, revenons sur le système d'équations (3.11). Nous proposons de reprendre la technique, vue en section (3.1.1), de décomposition du vecteur des inconnues :

$$\mathbf{A_g} = \mathbf{A_{g_t}} + \mathbf{A_{g_c}}$$

Ainsi, le système d'équations à résoudre est donné par :

$$\left\{\begin{array}{rcl} M_g.\mathbf{A_{g_t}} & = & \mathbf{B_g} \\ M_g.\mathbf{A_{g_c}} & = & L^T.\boldsymbol{\lambda} \\ L.(\mathbf{A_{g_t}} + \mathbf{A_{g_c}}) & = & \mathbf{E} \end{array}\right. \Leftrightarrow \left\{\begin{array}{rcll} M_g.\mathbf{A_{g_t}} & = & \mathbf{B_g} & (1) \\ \mathbf{A_{g_c}} & = & M_g^{-1}.L^T.\boldsymbol{\lambda} & (2) \\ \left(L.M_g^{-1}.L^T\right).\boldsymbol{\lambda} & = & \mathbf{E} - L.\mathbf{A_{g_t}} & (3) \end{array}\right. \tag{3.12}$$

C'est donc l'*articulation* qui doit résoudre ce système d'équations. Pour cela, elle possède l'ensemble des structures de donnée présentes dans l'équation (3.12) à savoir : les matrices L, $M_g^{-1}.L^T$ et $L.M_g^{-1}.L^T$ (nommée R), les vecteurs $\mathbf{A_{g_t}}$, $\mathbf{A_{g_c}}$, $\boldsymbol{\lambda}$ et $\mathbf{E} - L.\mathbf{A_{g_t}}$ (nommé $\mathbf{O}$). Ainsi, la résolution du système passe par le calcul des trois vecteurs inconnus :

Calcul des $\mathbf{A_{g_t}}$:

Afin d'obtenir les accélérations de tendance, l'articulation demande simplement à chaque objet de se résoudre (relativement à l'équation (3.12(1))). Ceux-ci ne connaissant ni l'existence des autres objets ni les contraintes inter-objets, ils se simulent sans prendre en compte la moindre contrainte. Les objets fournissent donc à l'*articulation* leurs inconnues de tendance (accélérations pour les systèmes dynamiques, déplacement pour les systèmes statiques...).

Calcul des $\boldsymbol{\lambda}$:

Le calcul des multiplicateurs de Lagrange s'effectue à l'aide de l'équation (3.12(3)). Pour cela, l'*articulation* demande à chaque objet impliqué dans une liaison de mettre à jour les structures de donnée utile : $\mathbf{E}$, L et $M_g^{-1}.L^T$. En retour, l'objet informe l'*articulation* si des modifications ont eu lieu ou non. Ensuite, l'*articulation* ne met à jour la matrice R qu'aux endroits utiles (elle connaît les données relatives à chaque objet (et leur emplacement dans les matrices) et elle sait si des modifications ont eu lieu pour chaque objet).

Chaque objet modifie la partie des données qui le concerne. Ainsi, si pour un couple <objet,contrainte> donné, la structure de la contrainte ne change pas, aucune modification n'est apportée à la matrice L, l'objet rend alors la main immédiatement.

Pourquoi considérer le couple <objet,contrainte> :

Une contrainte de point fixe apporte une matrice L constante pour les splines dynamiques mais variable pour les objets rigides. De même, une contrainte de point glissant (vue dans la section (3.3)) sur une spline dynamique impose une structure variable pour la matrice L. Il faut donc bien considérer le couple <objet,contrainte> pour déterminer si L est constante ou non.

Une fois la mise à jour effectuée par tous les objets, l'*articulation* peut résoudre l'équation (3.12(3)) à l'aide d'un algorithme direct de type Gauss ou décomposition LU ou à l'aide d'une méthode itérative de type bi-gradient conjugué.

Le système d'équations à résoudre étant de la taille du nombre de contraintes, on emploie la méthode de Gauss sur des simulations de moins de 15 contraintes atomiques. Au delà, d'autres techniques de résolution sont à utiliser, plus particulièrement les méthodes itératives (une étude comparative est menée dans la section dédiée aux applications (cf. section (3.4.2.3))).

Calcul des $\mathbf{A_{g_c}}$:

Le calcul des corrections s'effectue par le simple produit matrice vecteur de l'équation (3.12(2)) sachant que les structures de données nécessaires ont été mises à jour à l'étape précédente.

De cette manière, la résolution se déroule en effectuant les modifications minimales nécessaires sur les structures de donnée. Les phases d'initialisation et de mises à jour des matrices peuvent se mettre sous forme d'automate, comme le montre la figure (3.3).

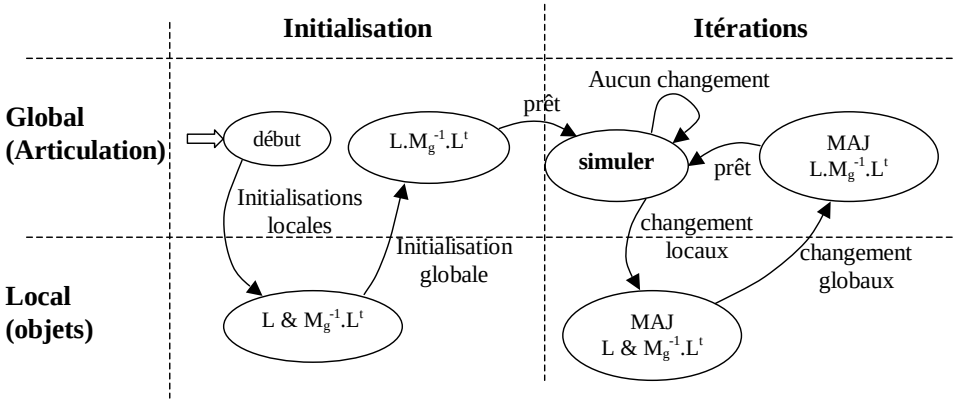


FIG. 3.3 – Automate des mises à jour de matrices

Il est à noter que l'utilisation, dans une même articulation, d'objets dynamiques et statiques (au sens mobiles avec une résolution mécanique par recherche d'états d'équilibres) apporte un certain décalage.

Pour l'objet statique, la contrainte sera forcément réalisée puisque son équation est directement utilisée dans le système mécanique. Les multiplicateurs de Lagrange assureront alors son intégrité. Ainsi, la contrainte est réalisée de manière quasi-statique par une succession d'états d'équilibres (un état d'équilibre par itération mécanique pour être plus précis).

Pour l'objet dynamique, l'équation de contrainte insérée dans le système mécanique est celle donnée par le schéma de Baumgarte, ce qui introduit potentiellement une violation de la contrainte. Ceci n'est pas du à l'utilisation d'objets statiques, mais uniquement à l'équation de Baumgarte. Par contre, le fait que l'objet statique ne possède aucune dynamique impose à l'objet dynamique de considérer à chaque itération qu'il est contraint avec un objet immobile. Or, ceci peut perturber la dynamique de l'objet si l'objet statique subit de grands déplacements.

Ce problème est ici souligné mais n'a pas fait l'objet de recherches plus poussées puisque notre étude se cantonne aux objets dynamiques.

Voyons maintenant plus en détails l'interface proposée pour les objets articulés.

3.2.3 Interface commune des objets et exemples d'implantations

Etant donnée la généricité de l'algorithme, une interface commune (cf. figure (3.2)) nommée *ObjetArticulable* permet d'établir un dialogue entre l'articulation et les différents objets articulés.

Cette interface permet notamment de renseigner et de mettre à jour les structures de donnée de l'*articulation* comme nous l'avons vu dans la section précédente. Cependant, ces mises à jour sont différentes selon le type de contrainte, ces méthodes se déclinent donc en plusieurs exemplaires relatifs aux contraintes.

Voici une partie de l'interface *ObjetArticulable* en $C++$:

```
class ObjetArticulable {
        ...
        virtual void setContraintePointFixeL(L,...,iemecontrainte,indiceDebutCorps,...)
        {};
        virtual void setContraintePointFixeE(E,...,iemecontrainte,...)
        {};
        virtual bool miseAJourContraintePointFixeL (L,...,iemecontrainte,indiceDebutCorps,...)
        {return false;};
        virtual void setContrainteTangenteFixeL(L,...,iemecontrainte,indiceDebutCorps,...)
        {};
        virtual void setContrainteTangenteFixeE(E,...,iemecontrainte,...)
        {};
        virtual bool miseAJourContrainteTangenteFixeL (L,...,iemecontrainte,indiceDebutCorps,...)
        {return false;};
        ...
}
```

Un objet articulable utilise les paramétres (`iemecontrainte` et `indiceDebutCorps`) de la méthode d'appel pour déterminer l'emplacement qui lui est réservé dans la structure : La variable `iemecontrainte` détermine la ligne dans la matrice L et l'indice dans le vecteur $\mathbf{E}$ alors que `indiceDebutCorps` indique la première colonne de la matrice L relative à l'objet.

L'*articulation*, ne sachant pas quelle méthode appeler sur l'objet, demande simplement à la contrainte de se résoudre en lui donnant l'information des deux objets contraints. C'est donc la contrainte qui appelle la méthode adéquate sur les deux objets, comme indiqué sur le schéma (3.4). Ce schéma présente le déroulement d'un appel de méthode de la classe articulation lors de l'exécution. Les cadres sur le schéma représentent donc des instances de classe.

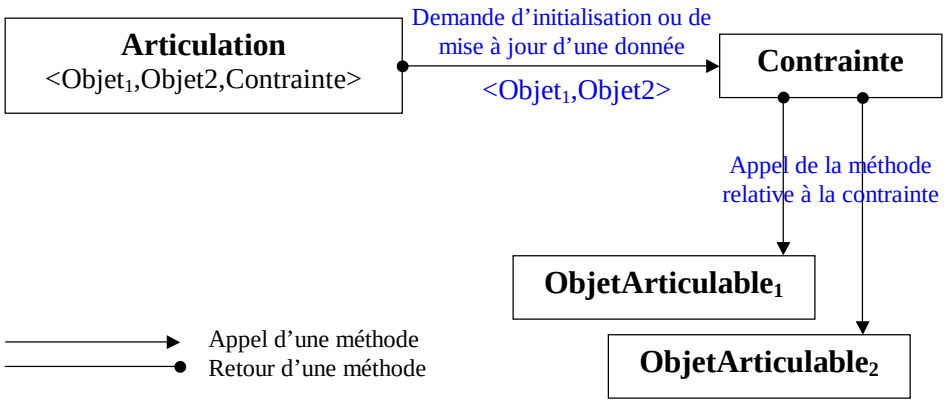


FIG. 3.4 – Schéma d'exécution d'un appel de méthode dans l'architecture

L'interface *Contrainte* sert donc d'intermédiaire entre l'*articulation* et les *ObjetArticulable*. Elle est donc composée de quelques méthodes permettant la généricité du coté de l'*articulation* et la spécificité du coté des *ObjetArticulable*. Voici les trois méthodes principales de l'interface :

```
class Contrainte {
        ...
        virtual void setContrainteL (L,iemeContrainte,...)
        {};
        virtual void setContrainteE (E,iemeContrainte,...)
        {};
        virtual void miseAJourContrainteL (L,iemeContrainte,...,
                *indiceDebutCorps1,*indiceDebutCorps2)
        {};
        ...
}
```

Nous allons maintenant détailler l'exemple d'une contrainte de point fixe sur trois objets simulés (spline dynamique, objet rigide et maillage masses-ressorts) à l'aide de formalismes physiques différents :

Spline dynamique (formalisme de Lagrange) :

Les équations dynamiques d'une spline sont données par (2.18) et la contrainte de point fixe pour une spline dynamique a été détaillée dans la section (3.1.2.1). Nous ne nous étendons donc pas sur cet exemple.

On remarque que les coefficients intervenant dans la matrice L sont des évaluations des fonctions de base des splines : $b_i(s)$. Ainsi, comme le point fixe est défini par une abscisse paramétrique constante, $b_i(s)$ est constant également et donc la matrice L n'a pas besoin de mises à jour.

Les méthodes `setContraintePointFixeL` et `setContraintePointFixeE` remplissent leurs parties des structures de donnée à l'aide des informations ci-dessus.

Tandis que la méthode `miseAJourContraintePointFixeL` se contente de retourner *faux*. Comme c'est le comportement par défaut de la méthode, l'objet n'a pas besoin de redéfinir cette méthode.

Objet rigide dynamique (formalisme de Newton-Euler) :

La dynamique du corps rigide a été détaillée en section (1.1.1) (on reprend donc les même notations) ainsi, on peut affirmer qu'un objet rigide est simulé dynamiquement à l'aide des équations (1.1) qui peuvent se mettre sous la forme :

$$\begin{pmatrix} M & 0 \\ 0 & I \end{pmatrix} \cdot \begin{pmatrix} \ddot{\mathbf{OG}} \\ \dot{\boldsymbol{\omega}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\rho} \\ \boldsymbol{\tau} \end{pmatrix} \tag{3.13}$$

avec

$$\begin{aligned} M &= \begin{pmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{pmatrix} \\ \boldsymbol{\rho} &= \sum_{i=0}^{n} \mathbf{F_i} \\ \boldsymbol{\tau} &= \sum_{i=0}^{n} \mathbf{GP_i} \wedge \mathbf{F_i} + (I\boldsymbol{\omega}) \wedge \boldsymbol{\omega} \\ I &\quad \text{la matrice d'inertie (cf. section (1.1.1))} \end{aligned}$$

Ainsi, un objet rigide possède bien les équations nécessaires pour être articulable (cf. équation (3.10)).

Soit $\mathbf{P}$ un point lié à l'objet rigide, on souhaite que $\mathbf{P}$ soit fixe (par rapport à un point de l'espace ou d'un autre objet). L'équation de contrainte pour fixer le point $\mathbf{P}$ à un point $\mathbf{A}$ de l'espace est :

$$\mathbf{C} = \mathbf{P} - \mathbf{A}$$

l'utilisation du schéma de Baumgarte (3.2) permet d'aboutir à l'équation :

$$\ddot{\mathbf{C}} + \frac{2}{\delta t}\dot{\mathbf{C}} + \frac{1}{\delta t^2}\mathbf{C} = 0$$

or, $\dot{\mathbf{C}} = \dot{\mathbf{OP}} = \dot{\mathbf{OG}} + \dot{\mathbf{GP}} = \dot{\mathbf{OG}} + \boldsymbol{\omega} \wedge \mathbf{GP}$
et, $\ddot{\mathbf{C}} = \ddot{\mathbf{OG}} + \dot{\boldsymbol{\omega}} \wedge \mathbf{GP} + \boldsymbol{\omega} \wedge \dot{\mathbf{GP}} = \ddot{\mathbf{OG}} + \dot{\boldsymbol{\omega}} \wedge \mathbf{GP} + \boldsymbol{\omega} \wedge (\boldsymbol{\omega} \wedge \mathbf{GP})$
donc :

$$\ddot{\mathbf{OG}} + \dot{\boldsymbol{\omega}} \wedge \mathbf{GP} = -\boldsymbol{\omega} \wedge (\boldsymbol{\omega} \wedge \mathbf{GP}) - \frac{2}{\delta t}\left(\dot{\mathbf{OG}} + \boldsymbol{\omega} \wedge \mathbf{GP}\right) - \frac{1}{\delta t^2}(\mathbf{P} - \mathbf{A}) \tag{3.14}$$

L'équation de contrainte (3.14) permet d'obtenir les coefficients de la matrice L et du vecteur $\mathbf{E}$. On remarque que les coefficients de L sont des 1 pour l'accélération du centre de masse ($\ddot{\mathbf{OP}}$) et une combinaison linéaire des coordonnées du vecteur $\mathbf{GP}$ pour l'accélération angulaire ($\dot{\boldsymbol{\omega}}$). Or, le vecteur $\mathbf{GP}$ est potentiellement en mouvement, donc la partie de L relative à l'objet rigide est variable.

La méthode *setContraintePointFixeE* remplit les trois coefficients du vecteur $\mathbf{E}$ qui lui sont réservés avec la partie droite de l'équation (3.14).
De même, la méthode *setContraintePointFixeL* remplit les coefficients de la matrice L qui lui sont réservés. Voici la structure de la matrice L avec une contrainte de point fixe pour un objet rigide défini par l'équation (3.13) :

$$L = \begin{array}{c} \texttt{indiceDebutCorps} \\ \downarrow \\ \left(\begin{array}{cccccccccc} \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & ? & 1 & 0 & 0 & 0 & GP_z & -GP_y & ? & \ldots \\ \ldots & ? & 0 & 1 & 0 & -GP_z & 0 & GP_x & ? & \ldots \\ \ldots & ? & 0 & 0 & 1 & GP_y & -GP_x & 0 & ? & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \end{array}\right) \end{array} \longrightarrow \texttt{iemeContrainte}$$

Au passage, la contrainte de point fixe s'écrit sous forme d'une équation vectorielle qui se décompose en trois équations atomiques. D'où la présence de trois lignes caractéristiques dans la structure de L. Les points d'interrogation montrent simplement que cette partie de la matrice n'est pas connue de l'objet rigide. Ce sont des zones réservées pour d'autres objets de l'articulation.
Le remplissage et la mise à jour des matrices et vecteurs sont effectués par chaque corps. Ainsi, chacun d'eux peut choisir l'organisation des données dans la structure. Par exemple, les splines séparent les données relatives à chaque axe (cela permet d'accélérer la résolution du système sans contrainte). Si l'on désire faire de même pour les corps rigides, la matrice des masses s'organise ainsi :

$$\begin{pmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & I_{11} & 0 & I_{12} & 0 & I_{13} \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & I_{21} & 0 & I_{22} & 0 & I_{23} \\ 0 & 0 & 0 & 0 & m & 0 \\ 0 & I_{31} & 0 & I_{32} & 0 & I_{33} \end{pmatrix} \cdot \begin{pmatrix} \ddot{OG}_x \\ \dot{\omega}_x \\ \ddot{OG}_y \\ \dot{\omega}_y \\ \ddot{OG}_z \\ \dot{\omega}_z \end{pmatrix} = \begin{pmatrix} \rho_x \\ \tau_x \\ \rho_y \\ \tau_y \\ \rho_z \\ \tau_z \end{pmatrix}$$

et la matrice des contraintes est alors renseignée pour une contrainte de point fixe avec le schéma suivant :

$$L = \begin{array}{c} \texttt{indiceDebutCorps} \\ \downarrow \\ \left(\begin{array}{cccccccccc} \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & ? & 1 & 0 & 0 & GP_z & 0 & -GP_y & ? & \ldots \\ \ldots & ? & 0 & -GP_z & 1 & 0 & 0 & GP_x & ? & \ldots \\ \ldots & ? & 0 & GP_y & 0 & -GP_x & 1 & 0 & ? & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \end{array}\right) \end{array} \longrightarrow \texttt{iemeContrainte}$$

La méthode *miseAJourContraintePointFixeL* modifie simplement les 6 coefficients de la matrice L relatifs aux coordonnées du vecteur $\mathbf{GP}$ (2 coefficients par contrainte atomique) et retourne *vrai*.

Maillage masses-ressorts (formalisme de Newton) :

Les modèles masses-ressorts ont été développés en section (1.1.2). Ainsi, la simulation dynamique de n particules $\mathbf{q_i}$ de masse m_i passe par la résolution du système d'équations :

$$M\mathbf{A} = \mathbf{F}$$

où M est la matrice des masses, $\mathbf{A}$ le vecteur des accélérations et $\mathbf{F}$ le vecteur des forces internes et externes. En choisissant une organisation des données par axe, le vecteur A s'écrit comme :

$$\mathbf{A} = \begin{pmatrix} \mathbf{A_x} \\ \mathbf{A_y} \\ \mathbf{A_z} \end{pmatrix}$$

et la matrice M posséde la structure diagonale suivante :

$$M = \begin{pmatrix} m_1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & m_n & & & & 0 & & \\ & & & m_1 & & & & & \\ & & & & \ddots & & & & \\ & & & & & m_n & & & \\ & & 0 & & & & m_1 & & \\ & & & & & & & \ddots & \\ & & & & & & & & m_n \end{pmatrix}$$

La contrainte qui fixe un point $\mathbf{q_i}$ du maillage en un point $\mathbf{A}$ de l'espace s'écrit :

$$\mathbf{C} = \mathbf{q_i} - \mathbf{A}$$

donc, le schéma de Baumgarte (3.2) permet d'introduire l'équation suivante dans le système :

$$\ddot{\mathbf{q}}_\mathbf{i} = -\frac{2}{\delta t}\dot{\mathbf{q}}_\mathbf{i} - \frac{1}{\delta t^2}\mathbf{q_i} \tag{3.15}$$

On remarque que les coefficients des accélérations $\ddot{\mathbf{q}}_\mathbf{i}$ sont constants égaux à 1. Ainsi, la matrice L est constante.

La méthode *setContraintePointFixeE* remplit les trois coefficients du vecteur $\mathbf{E}$ qui lui sont réservés avec la partie droite de l'équation (3.15).
De même, la méthode *setContraintePointFixeL* remplit les coefficients de la matrice L qui lui sont réservés. Voici la structure de la matrice L avec une contrainte de point fixe pour un maillage masses-ressorts :

$$L = \begin{pmatrix} \overbrace{\begin{matrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{matrix}}^{\downarrow\ x} & \overbrace{\begin{matrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{matrix}}^{\downarrow\ y} & \overbrace{\begin{matrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{matrix}}^{\downarrow\ z} \end{pmatrix} \longrightarrow \texttt{iemeContrainte}$$

`indiceDebutCorps`

l'*indiceDebutCorps* indique que le point contraint est le second point du modèle masses-ressorts.
Comme les coefficients de L sont constants, la méthode *miseAJourContraintePointFixeL* se contente de retourner *faux*. Comme c'est le comportement par défaut de la méthode, l'objet n'a pas besoin de redéfinir cette méthode.

Les matrices sont constantes pour les splines dynamiques et les maillages masses-ressorts. Pour les objets rigides, la mise à jour de $M_g^{-1}L^T$ s'effectue avec une complexité temporelle linéaire par rapport à la dimension de la matrice M_g ($\mathcal{O}(n)$ avec $n = dim(M_g)$). Il s'en suit la mise à jour de la matrice R au niveau de l'*articulation*, celle ci s'effectue avec une complexité en $\mathcal{O}(c^2n)$ où c est le nombre total de contraintes.

Certains modèles peuvent définir des contraintes qui leur sont particulières. C'est le cas par exemple des splines qui peuvent être contraintes au niveau des tangentes ou courbures (cf. section (3.1)). Mais aussi des objets rigides qui peuvent subir une contrainte de glissière ou de rotation (cf. annexe (G)). Ainsi, plusieurs objets peuvent cohabiter dans le même système, se contraindre les uns les autres sans pour autant avoir des interfaces de contraintes équivalentes.

Les collisions d'une articulation avec les objets extérieurs sont détectées par la plate-forme de simulation. On peut également utiliser la plate-forme pour détecter les auto-collisions. Cependant, dans le cas d'une articulation, certaines liaisons peuvent imputer des auto-collisions naturelles, intrinsèques à la configuration articulée. Ces auto-collisions doivent donc être ignorées pour assurer le maintien et la stabilité de la liaison. Il incombe donc à l'articulation de décider pour chaque auto-collision détectée par la plate-forme si celle-ci doit être prise en compte ou non. Dans notre logiciel de test, basé sur notre plate-forme de simulation SPORE, les collisions sont soumises à une discrétisation en sphère des objets. Les auto-collisions peuvent être détectées si l'articulation le demande. La détection se fait alors à l'aide de la même discrétisation en sphères que les collisions. Ainsi, une auto-collision met en jeu un couple de sphères de l'articulation en inter-pénétration. Pour chaque couple de sphères, l'articulation doit décider si la force générée est prise en compte ou non. Pour ce faire, plusieurs choix s'offrent à nous : on peut considérer que la configuration au départ nous indique les couples de sphères naturellement en auto-collision, ou encore, attendre le premier état d'équilibre du système articulé pour déterminer les couples de sphères naturellement en auto-collision. Ensuite, il suffit d'utiliser cette table de correspondance pour déterminer si un couple de sphères est naturellement en auto-collision ou si celle-ci doit générer une force.

Toutes les contraintes vues jusqu'à présent concernent des points bien référencés d'un objet. Cependant certaines contraintes peuvent vouloir être relatives à des points variables d'un objet. Pour ce type de contrainte, les formulations vues au dessus sont insuffisantes. Donc, nous proposons à présent une nouvelle classe de contraintes applicable à tout objet paramétrique continu. Cette classe porte le nom de *contraintes glissantes* et permet de définir une contrainte mobile sur un modèle continu possédant une paramétrisation.

3.3 Contraintes glissantes (*Smooth Constraints*)

Nous proposons maintenant une nouvelle classe de contraintes appelées *contraintes glissantes*. On appelle contrainte *glissante* toute contrainte relative à un point mobile (selon la dynamique du modèle) d'un objet. Autrement dit, la contrainte possède un mouvement libre et non imposé. En effet, pour effectuer une contrainte capable de se déplacer suivant un mouvement imposé, il est possible d'utiliser une contrainte de type *point fixe* en déplacement simplement le point fixe aux endroits souhaités. Ici, il s'agit de laisser libre le mouvement de la contrainte.

Ce type de contrainte peut s'avérer très utile dans beaucoup de domaines liés à la simulation.

Par exemple, on peut utiliser cette classe de contraintes pour simuler la suture d'un organe avec un fil chirurgical, le fil est contraint de passer de part et d'autre de la surface de l'organe à un endroit précis, mais il doit pourvoir glisser sur ce point d'insertion. De plus, une proposition d'extension de la contrainte est proposée pour introduire un phénomène de friction locale au point contraint de la spline, simulant ainsi le frottement sec induit par l'interaction des deux modèles[LMGC04].

De même, on peut utiliser cette classe de contraintes glissantes pour simuler un lacet qui glisse dans les trous d'une chaussure. De manière générale, cela permet de simuler un fil contraint de passer à un endroit précis de l'espace (ou en un point d'un autre objet, en utilisant une *articulation* vue dans la section précédente sur les contraintes externes (cf. section (3.2))).

Il est à noter que ce type d'interactions est en pratique géré par les contacts, les collisions, les frottements... Malheureusement, il n'est pas possible de simuler en temps réel l'ensemble de ces interactions avec la précision nécessaire pour obtenir le résultat souhaité. C'est pourquoi nous proposons une solution au travers d'une classe de contraintes spécifique à ce type d'interactions.

Nous proposons[LMGC04, LGM+04] une méthode pour gérer des contraintes glissantes sur des modèles continus paramétriques quelconques. On suppose donc que l'objet est capable de fournir une équation paramétrique définissant continuement ses points (on suppose également que l'objet est au moins de continuité C^0). Cette classe générique de contraintes est détaillée dans une première section et modifiée pour accélérer les temps de calculs. Dans une seconde section, nous étudions l'application de cette classe de contraintes au cas de la spline dynamique. Enfin, nous proposons dans une troisième section une approche pour introduire de la friction locale dans le modèle spline au niveau d'une contrainte de type point glissant. En effet, une telle contrainte s'emploie lorsqu'un modèle coulisse sur un objet ou un point d'insertion. Autrement dit, l'emploi de cette contrainte nécessite de tenir compte de l'interaction du modèle avec l'autre objet en contact. Il est donc naturel de proposer un modèle capable d'intégrer des frottements modélisant ce contact.

3.3.1 Contrainte glissante paramétrique : cadre général

Cette section introduit de manière générale les équations liées aux contraintes glissantes. Pour cela, elle n'est pas particulièrement attachée au modèle de spline dynamique étudié précédemment. Le couplage de cette classe de contraintes avec le modèle de spline dynamique est étudié dans la section suivante (3.3.2).

Dans cette section, on considère un système matériel défini à l'aide de n degrés de liberté q_i, composé d'objets paramétriques de dimensions quelconques. Le système peut éventuellement intégrer des contraintes (au nombre de c) à l'aide de la méthode des multiplicateurs de Lagrange. Un tel système matériel possède un espace des paramètres de dimension n (défini par les degrés de liberté) et sa dynamique est donnée par le système d'équations :

$$\begin{pmatrix} M & L^T \\ L & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ -\boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{E} \end{pmatrix}$$

Le but est alors d'ajouter au système matériel d équations atomiques de contraintes (toute équation vectorielle se décompose en équations atomiques) liées à des points qui se meuvent le long de, sur ou dans l'objet (suivant la dimension).

Le fait que les contraintes soient liées à des points mobiles d'un modèle paramétrique suppose simplement qu'elles sont définies pour des abscisses paramétriques s_i variables (on regroupe l'ensemble de ces l abscisses paramétriques dans un vecteur $\mathbf{s}$). Et comme les déplacements des contraintes sont relatifs à la dynamique du modèle, on définit simplement les abscisses paramétriques comme des variables dynamiques :

$$\mathbf{s(t)}$$

Ainsi, les variables $\mathbf{s}$ deviennent des paramètres du système matériel avec leur propre dynamique :

$$(\mathbf{s}(t), \dot{\mathbf{s}}(t), \ddot{\mathbf{s}}(t))$$

Comme ces paramètres ne sont ni des degrés de liberté, ni des multiplicateurs de Lagrange, ils portent le nom de ***variables libres***[Rém03] et étendent l'espace des paramètres du système matériel de la dimension n à la dimension $n + l$.

La dynamique des variables libres est régie par les d équations de contraintes glissantes. La forme générale des équations de contraintes glissantes (bilatérales et holonomes ou cinématiques) est l'équation vectorielle :

$$\mathbf{g}(\mathbf{q}(t), \dot{\mathbf{q}}(t), \mathbf{s}(t), \dot{\mathbf{s}}(t), t) = \mathbf{0}_{(\mathbf{d})} \tag{3.16}$$

on remarque qu'une contrainte glissante peut dépendre de plusieurs variables libres ou même de leur vitesse.

On fait le choix de la méthode des multiplicateurs de Lagrange pour ajouter ces contraintes dans le système matériel. Pour ne pas confondre les équations de contraintes glissantes des autres équations de contraintes, on distingue les multiplicateurs de Lagrange $\boldsymbol{\lambda_g}$ des contraintes glissantes des autres $\boldsymbol{\lambda}$. Ainsi, le système matériel est complété en :

$$\begin{pmatrix} M & 0 & L^T & \chi_5^T \\ \chi_1 & \chi_2 & \chi_3 & \chi_4 \\ L & 0 & 0 & 0 \\ \chi_5 & \chi_6 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ \ddot{\mathbf{s}} \\ -\boldsymbol{\lambda} \\ -\boldsymbol{\lambda_g} \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \boldsymbol{\chi_7} \\ \mathbf{E} \\ \boldsymbol{\chi_8} \end{pmatrix}$$

où les χ_i sont des données que nous allons déterminer dans la suite.

L'intégration des équations (3.16) dans le système d'équations dynamiques s'établit, pour chaque contrainte atomique, comme suit :

Contrainte holonome :

Dans le cas d'une contrainte de type holonome, l'équation se réécrit :

$$g(\mathbf{q}(t), \mathbf{s}(t), t) = 0$$

et on choisit la méthode de Baumgarte pour déterminer la dynamique de la contrainte. On propose donc d'utiliser le schéma de Baumgarte (cf. équation (3.2)) pour définir l'équation à ajouter au système matériel. Ce schéma fait intervenir une combinaison linéaire des termes g, $\dot{g}$ et $\ddot{g}$:

$$\begin{aligned} \dot{g}(\mathbf{q}(t), \mathbf{s}(t), t) &= \dot{\mathbf{q}}(t).\frac{\partial g}{\partial \mathbf{q}}(\mathbf{q}(t), \mathbf{s}(t), t) + \dot{\mathbf{s}}(t).\frac{\partial g}{\partial \mathbf{s}}(\mathbf{q}(t), \mathbf{s}(t), t) \\ \text{et} & \\ \ddot{g}(\mathbf{q}(t), \mathbf{s}(t), t) &= \ddot{\mathbf{q}}(t).\frac{\partial g}{\partial \mathbf{q}}(\mathbf{q}(t), \mathbf{s}(t), t) + \ddot{\mathbf{s}}(t).\frac{\partial g}{\partial \mathbf{s}}(\mathbf{q}(t), \mathbf{s}(t), t) + \\ & \quad \dot{\mathbf{q}}(t).\frac{d}{dt}\left(\frac{\partial g}{\partial \mathbf{q}}(\mathbf{q}(t), \mathbf{s}(t), t)\right) + \dot{\mathbf{s}}(t).\frac{d}{dt}\left(\frac{\partial g}{\partial \mathbf{s}}(\mathbf{q}(t), \mathbf{s}(t), t)\right) \end{aligned}$$

donc, le facteur des accélérations des degrés de liberté (χ_5) est $\dfrac{\partial g}{\partial \mathbf{q}}$ et le facteur des accélérations des variables libres (χ_6) est $\dfrac{\partial g}{\partial \mathbf{s}}$.

En formant l'équation de Baumgarte et en isolant les deux termes $\ddot{\mathbf{q}}.\dfrac{\partial g}{\partial \mathbf{q}}$ et $\ddot{\mathbf{s}}.\dfrac{\partial g}{\partial \mathbf{s}}$ d'un côté de l'équation, l'autre côté de l'équation forme le terme ($\boldsymbol{\chi_8}$). Donc,

$$\begin{aligned} \boldsymbol{\chi_8} &= -\dot{\mathbf{q}}(t).\frac{d}{dt}\left(\frac{\partial g}{\partial \mathbf{q}}(\mathbf{q}(t), \mathbf{s}(t), t)\right) - \dot{\mathbf{s}}(t).\frac{d}{dt}\left(\frac{\partial g}{\partial \mathbf{s}}(\mathbf{q}(t), \mathbf{s}(t), t)\right) \\ &- \frac{2}{\delta t}\left(\dot{\mathbf{q}}(t).\frac{\partial g}{\partial \mathbf{q}}(\mathbf{q}(t), \mathbf{s}(t), t) + \dot{\mathbf{s}}(t).\frac{\partial g}{\partial \mathbf{s}}(\mathbf{q}(t), \mathbf{s}(t), t)\right) \\ &- \frac{1}{\delta t^2} g(\mathbf{q}(t), \mathbf{s}(t), t) \end{aligned}$$

Contrainte cinématique :

Dans le cas d'une contrainte cinématique, l'équation de contrainte prend la forme :

$$g_c(\mathbf{q}(t), \dot{\mathbf{q}}(t), \mathbf{s}(t), \dot{\mathbf{s}}(t), t) = 0$$

et son intégration dans le système s'effectue par l'application du schéma décrit dans l'équation (3.4). Ce schéma fait intervenir une combinaison linéaire des termes g_c et $\dot{g}_c$ or :

$$\begin{aligned} \dot{g}_c(\mathbf{q}(t), \dot{\mathbf{q}}(t), \mathbf{s}(t), \dot{\mathbf{s}}(t), t) &= \dot{\mathbf{q}}(t).\frac{\partial g_c}{\partial \mathbf{q}}(\mathbf{q}(t), \dot{\mathbf{q}}(t), \mathbf{s}(t), \dot{\mathbf{s}}(t), t) + \ddot{\mathbf{q}}(t).\frac{\partial g_c}{\partial \dot{\mathbf{q}}}(\mathbf{q}(t), \dot{\mathbf{q}}(t), \mathbf{s}(t), \dot{\mathbf{s}}(t), t) \\ &+ \dot{\mathbf{s}}(t).\frac{\partial g_c}{\partial \mathbf{s}}(\mathbf{q}(t), \dot{\mathbf{q}}(t), \mathbf{s}(t), \dot{\mathbf{s}}(t), t) + \ddot{\mathbf{s}}(t).\frac{\partial g_c}{\partial \dot{\mathbf{s}}}(\mathbf{q}(t), \dot{\mathbf{q}}(t), \mathbf{s}(t), \dot{\mathbf{s}}(t), t) \end{aligned}$$

donc, le facteur des accélérations des degrés de liberté (χ_5) est $\frac{\partial g_c}{\partial \dot{\mathbf{q}}}$ et le facteur des accélérations des variables libres (χ_6) est $\frac{\partial g_c}{\partial \dot{\mathbf{s}}}$.

En formant l'équation de Baumgarte et en isolant les deux termes $\ddot{\mathbf{q}}.\frac{\partial g_c}{\partial \dot{\mathbf{q}}}$ et $\ddot{\mathbf{s}}.\frac{\partial g_c}{\partial \dot{\mathbf{s}}}$ d'un côté de l'équation, l'autre côté de l'équation forme le terme ($\boldsymbol{\chi_8}$). Donc,

$$\begin{aligned}\boldsymbol{\chi_8} &= -\dot{\mathbf{q}}(t).\frac{\partial g_c}{\partial \mathbf{q}}(\mathbf{q}(t),\dot{\mathbf{q}}(t),\mathbf{s}(t),\dot{\mathbf{s}}(t),t) - \dot{\mathbf{s}}(t).\frac{\partial g_c}{\partial \mathbf{s}}(\mathbf{q}(t),\dot{\mathbf{q}}(t),\mathbf{s}(t),\dot{\mathbf{s}}(t),t)\\ &- kg_c(\mathbf{q}(t),\dot{\mathbf{q}}(t),\mathbf{s}(t),\dot{\mathbf{s}}(t),t)\end{aligned}$$

Ainsi, le système étendu aux d contraintes glissantes se complète en :

$$\begin{pmatrix} M & 0 & L^T & L_g^T \\ \chi_1 & \chi_2 & \chi_3 & \chi_4 \\ L & 0 & 0 & 0 \\ L_g & L_{gs} & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ \ddot{\mathbf{s}} \\ -\boldsymbol{\lambda} \\ -\boldsymbol{\lambda_g} \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \boldsymbol{\chi_7} \\ \mathbf{E} \\ \mathbf{E_g} \end{pmatrix}$$

où les matrices L_g et L_{gs} ainsi que le vecteur $\mathbf{E_g}$ sont déterminés par la méthode adéquate définie ci-dessus.

Il reste à déterminer les équations qui complètent le système mécanique. Cette détermination est différente suivant les considérations faites sur la contrainte. Nous supposons dans un premier temps que la contrainte est parfaite, sans aucune perte d'énergie. Puis nous proposons une modification des équations pour accélérer la résolution du système d'équations.

3.3.1.1 Contraintes glissantes parfaites

Pour déterminer l'ensemble des équations supplémentaires à ajouter, on considère les variations virtuelles compatibles avec les contraintes (supposées holonomes dans cette démonstration) $\begin{pmatrix} \hat{\mathbf{q}} & \hat{\mathbf{s}} \end{pmatrix}^T$, qui sont définies par[Rém03] :

$$\begin{pmatrix} \frac{\partial \mathbf{g}}{\partial \mathbf{q}} & \frac{\partial \mathbf{g}}{\partial \mathbf{s}} \end{pmatrix} . \begin{pmatrix} \hat{\mathbf{q}} \\ \hat{\mathbf{s}} \end{pmatrix} = 0$$

Ce vecteur $\begin{pmatrix} \hat{\mathbf{q}} & \hat{\mathbf{s}} \end{pmatrix}^T$ représente les variations possibles, dans l'espace des paramètres, qui n'entraînent pas de violation de la contrainte. Autrement dit, ce vecteur balaie l'ensemble des déplacements virtuels permettant de conserver la réalisation de la contrainte.
On se place maintenant dans le cas d'une contrainte parfaite (sans frottement) dans l'espace des paramètres. Si on considère une variation virtuelle des paramètres, compatible avec la contrainte, et comme la liaison est non dissipative, la contrainte ne produit aucun effort de maintien. Donc, si l'on nomme Q_g les composantes généralisées des efforts de maintien de cette contrainte $\mathbf{g}$, on a la relation (principe des liaisons non dissipatives)[Rém03] :

$$Q_g^T . \begin{pmatrix} \hat{\mathbf{q}} \\ \hat{\mathbf{s}} \end{pmatrix} = \mathbf{0}$$

où l'on peut décomposer la matrice Q_g^T en :

$$Q_g^T = \begin{pmatrix} Q_{g_q}^T & Q_{g_s}^T \end{pmatrix}$$

On obtient donc deux familles ($\begin{pmatrix} \frac{\partial \mathbf{g}}{\partial \mathbf{q}} & \frac{\partial \mathbf{g}}{\partial \mathbf{s}} \end{pmatrix}$ et $\begin{pmatrix} Q_{g_q}^T & Q_{g_s}^T \end{pmatrix}$) de $n+l$ vecteurs définissant le même espace vectoriel orthogonal à celui des variations compatibles. Si l'on suppose que les variables libres sont indépendantes alors la famille $\frac{\partial \mathbf{g}}{\partial \mathbf{s}}$ est libre. Elle devient donc une base de cet espace othogonal. Il existe donc d scalaires (les multiplicateurs de Lagrange (notés sous forme vectorielle $\boldsymbol{\lambda}_\mathbf{g}$) relatifs à la contrainte vectorielle $\mathbf{g}$) liant ces deux familles génératrices :

$$Q_g^T = \boldsymbol{\lambda}_\mathbf{g}^T . \begin{pmatrix} \frac{\partial \mathbf{g}}{\partial \mathbf{q}} & \frac{\partial \mathbf{g}}{\partial \mathbf{s}} \end{pmatrix}$$

ou, en décomposant :

$$Q_{g_q} = \frac{\partial \mathbf{g}}{\partial \mathbf{q}}^T . \boldsymbol{\lambda}_\mathbf{g} \quad \text{et} \quad Q_{g_s} = \frac{\partial \mathbf{g}}{\partial \mathbf{s}}^T . \boldsymbol{\lambda}_\mathbf{g}$$

Or, la contrainte est considérée comme parfaite, donc dans le cas de seule variation des variables libres, on déduit la relation :

$$\frac{\partial \mathbf{g}}{\partial \mathbf{s}}^T . \boldsymbol{\lambda}_\mathbf{g} = \mathbf{0} \tag{3.17}$$

L'équation (3.17) compléte le système d'équations dynamiques. Ainsi, ce système complet prend la forme :

$$\begin{pmatrix} M & 0 & L^T & L_g^T \\ 0 & 0 & 0 & L_{gs}^T \\ L & 0 & 0 & 0 \\ L_g & L_{gs} & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ \ddot{\mathbf{s}} \\ -\boldsymbol{\lambda} \\ -\boldsymbol{\lambda}_\mathbf{g} \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{0} \\ \mathbf{E} \\ \mathbf{E_g} \end{pmatrix} \tag{3.18}$$

avec, pour rappel, les dimensions suivantes :

$$\begin{array}{cccc} M_{(n\times n)} & L_{(c\times n)} & L_{g(d\times n)} & L_{gs(d\times l)} \\ \mathbf{A_{(n)}}, \mathbf{B_{(n)}} & \boldsymbol{\lambda}_{\mathbf{(c)}}, \mathbf{E_{(c)}} & \boldsymbol{\lambda}_{\mathbf{g(d)}}, \mathbf{E_{g(c)}} & \ddot{\mathbf{s}}_{\mathbf{(l)}} \end{array}$$

Afin de résoudre efficacement ce système d'équations, Rémion[Rém03] propose une technique de résolution particulière basée sur une décomposition des accélérations en trois composantes :

$$\mathbf{A} = \mathbf{A_t} + \mathbf{A_c} + \mathbf{A_{cg}}$$

où $\mathbf{A_t}$ correspond à l'accélération de tendance (i.e. l'accélération du système sans contraintes), $\mathbf{A_c}$ est l'accélération de correction relative aux contraintes non glissantes et $\mathbf{A_{cg}}$ est l'accélération de correction due aux contraintes glissantes.

L'algorithme proposé s'appuie sur neuf étapes de calcul demandant chacune des opérations lourdes sur des systèmes matriciels. Dans la suite, nous proposons de modifier le système (3.18) pour accélérer sa résolution.

3.3.1.2 Cas particulier des contraintes qui travaillent

Si l'on considère que la contrainte n'est pas parfaite, un terme supplémentaire apparaît dans l'équation, soit pour renforcer la contrainte soit pour lutter contre elle. Ce terme supplémentaire est schématisé sur la figure (3.5) et peut être considéré comme une force de pénalité.

Dans le cas de contraintes parfaites, les forces dues aux contraintes sont orthogonales aux tangentes locales respectives des contraintes (ce qui procure un travail physique nul). Si l'on considère que ce n'est plus le cas, alors les forces travaillent au sens physique du terme. Sur une itération, elles produisent donc des énergies (positives ou négatives) influençant la dynamique des variables libres.

Ainsi, on propose de modifier les équations de contraintes parfaites (3.17) en liant les travaux des contraintes aux accélérations des variables libres :

$$\epsilon . \ddot{\mathbf{s}} - \frac{\partial \mathbf{g}}{\partial \mathbf{s}}^T . \boldsymbol{\lambda}_\mathbf{g} = \mathbf{0} \tag{3.19}$$

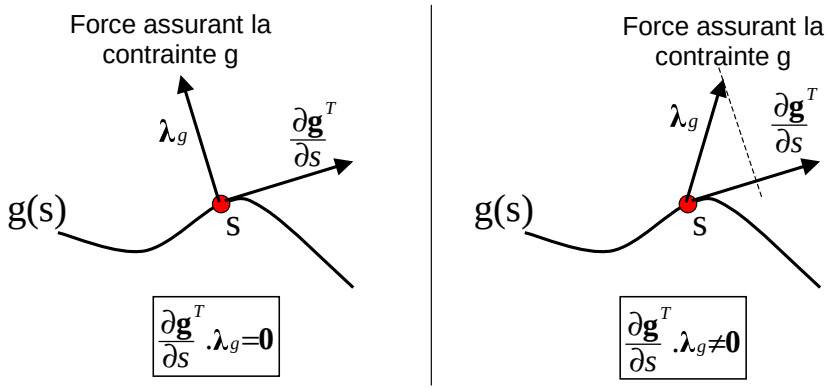


FIG. 3.5 – Contrainte glissante

avec ϵ la matrice qui fait la jonction entre le travail fourni par chaque contrainte aux accélérations des variables libres. Autrement dit, chaque ligne de cette matrice renseigne sur les dépendances d'une contrainte glissante vis-à-vis de l'ensemble des variables libres.

Si par exemple, chaque contrainte glissante atomique est dépendante d'une seule variable libre et que réciproquement chaque variable libre est liée à une contrainte, on obtient alors une matrice diagonale (où l'on remarque aisément la relation "une contrainte $\longleftrightarrow$ une variable libre") :

$$\epsilon = \begin{pmatrix} \epsilon_1 & 0 & \dots & 0 \\ 0 & \epsilon_2 & \dots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \dots & \epsilon_l \end{pmatrix}$$

Chaque élément diagonal est un scalaire constant positif ou négatif permettant de lier le travail de la force, due à une contrainte, à l'accélération de la variable libre liée à cette contrainte.

Par exemple, si les coefficients de la matrice sont positifs, un travail positif de la force induit une décélération de la variable libre. On obtient donc une résistance sur la contrainte. On peut également choisir des coefficients négatifs, dans ce cas, tout mouvement de la contrainte produisant un travail positif induira une accélération de la variable libre, appuyant le déplacement de la contrainte.

Le terme ϵ introduit des forces liées aux accélérations dans le système dynamique, ce sont donc des forces d'inerties. Ainsi, le terme ϵ peut être qualifié de terme inertiel. Autrement dit, le fait que les contraintes glissantes soient freinées ou accélérées est dû à une masse plutôt qu'à un frottement.

L'introduction de cette nouvelle équation (3.19) dans le système matériel permet d'aboutir à :

$$\begin{pmatrix} M & 0 & L^T & L_g^T \\ 0 & \epsilon & 0 & L_{gs}^T \\ L & 0 & 0 & 0 \\ L_g & L_{gs} & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ \ddot{\mathbf{s}} \\ -\boldsymbol{\lambda} \\ -\boldsymbol{\lambda}_\mathbf{g} \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{0} \\ \mathbf{E} \\ \mathbf{E_g} \end{pmatrix} \tag{3.20}$$

Par le biais de cette matrice ϵ, il est possible d'appuyer ou de contrer une contrainte(en choisissant ϵ positif ou négatif, cela induit une accélération ou une décélération des variables libres). De plus, la relation (3.19) donne une relation directe entre l'accélération des variables libres et les multiplicateurs de Lagrange des contraintes glissantes. Ceci permet de simplifier la résolution du système.

Résolution :

Afin de faciliter la compréhension de la résolution nous ne distinguons plus les contraintes glissantes des autres contraintes. On considère donc un système matériel à n degrés de liberté, l variables libres et $c + d$ contraintes (d contraintes glissantes et c contraintes non glissantes). Le système d'équations

dynamiques résultant prend donc la forme :

$$\begin{pmatrix} M & 0 & L^T \\ 0 & \epsilon & L_s^T \\ L & L_s & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ \ddot{\mathbf{s}} \\ -\boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{0} \\ \mathbf{E} \end{pmatrix}$$

donc, la partie de la matrice L_s correspondant aux contraintes non glissantes est nulle.

On procède en décomposant l'accélération $\mathbf{A}$ en deux composantes (cf. équation (3.5)) : $\mathbf{A_t}$ et $\mathbf{A_c}$. Ainsi, les équations à résoudre sont :

$$\left\{\begin{array}{lcl} \mathbf{A} & = & \mathbf{A_t} + \mathbf{A_c} \\ M.\mathbf{A_t} & = & \mathbf{B} \\ M.\mathbf{A_c} & = & L^T.\boldsymbol{\lambda} \\ \epsilon.\ddot{\mathbf{s}} & = & L_s^T.\boldsymbol{\lambda} \\ L.(\mathbf{A_t} + \mathbf{A_c}) + L_s.\ddot{\mathbf{s}} & = & \mathbf{E} \end{array}\right. \Leftrightarrow \left\{\begin{array}{lcl} \mathbf{A} & = & \mathbf{A_t} + \mathbf{A_c} \\ \mathbf{A_t} & = & M^{-1}.\mathbf{B} \\ \mathbf{A_c} & = & M^{-1}.L^T.\boldsymbol{\lambda} \\ \ddot{\mathbf{s}} & = & \epsilon^{-1}.L_s^T.\boldsymbol{\lambda} \\ L.\mathbf{A_c} + L_s.\ddot{\mathbf{s}} & = & \mathbf{E} - L.\mathbf{A_t} \end{array}\right.$$

$$\left\{\begin{array}{lcll} \mathbf{A} & = & \mathbf{A_t} + \mathbf{A_c} & (1) \\ \mathbf{A_t} & = & M^{-1}.\mathbf{B} & (2) \\ \mathbf{A_c} & = & M^{-1}.L^T.\boldsymbol{\lambda} & (3) \\ \ddot{\mathbf{s}} & = & \epsilon^{-1}.L_s^T.\boldsymbol{\lambda} & (4) \\ \left(L.M^{-1}.L^T + L_s.\epsilon^{-1}.L_s^T\right).\boldsymbol{\lambda} & = & \mathbf{E} - L.\mathbf{A_t} & (5) \end{array}\right. \tag{3.21}$$

Si les contraintes sont indépendantes vis-à-vis des variables libres, la matrice ϵ est non singulière, on peut donc calculer son inverse.

De cette manière, le système d'équations est résolu en quatre étapes.

1) Résoudre l'équation (2).

2) Résoudre l'équation (5).

3) Calculer les accélérations de corrections (3).

4) Calculer les accélérations des variables libres (4).

Enfin, il suffit de reconstruire l'accélération totale en sommant les deux composantes $\mathbf{A_t}$ et $\mathbf{A_c}$.

Il peut être relevé que l'utilisation d'une contrainte glissante produisant un travail induit une inertie au niveau de la contrainte. Cette inertie étant mal contrôlé, cette technique est utilisée en pratique avec un coefficient ϵ faible pour éviter que l'inertie soit trop importante, ce qui pourrait être gênant.

3.3.2 Contraintes glissantes dans le modèle spline dynamique

L'intégration de la classe des contraintes glissantes dans le modèle spline permet d'étendre le système d'équations (3.3) en celui décrit à l'équation (3.20) :

$$\begin{pmatrix} \mathcal{M} & 0 & L^T & L_g^T \\ 0 & \epsilon & 0 & L_{gs}^T \\ L & 0 & 0 & 0 \\ L_g & L_{gs} & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ \ddot{\mathbf{s}} \\ -\boldsymbol{\lambda} \\ -\boldsymbol{\lambda_g} \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{0} \\ \mathbf{E} \\ \mathbf{E_g} \end{pmatrix} \tag{3.22}$$

La classe des contraintes glissantes se décline, sur le modèle de spline dynamique, en plusieurs contraintes types comme la contrainte de point glissant, de tangente glissante ou de courbure glissante.

Mais on trouve également des contraintes de type direction de tangente glissante. D'une manière générale, on introduit également la notion de contrainte glissante double applicable à la plupart des contraintes citées auparavant.

3.3.2.1 Contrainte de point glissant

Une contrainte de point glissant désigne simplement une contrainte de point fixe (cf. section (3.1.2.1)) où l'abscisse paramétrique du point considéré devient dynamique. Ainsi, cela revient à fixer la position d'un point de la spline dans l'espace (un point immobile $\mathbf{A}$), sachant que ce point n'est pas le même à chaque itération. Autrement dit, cette contrainte impose à la spline de passer par un point de l'espace, peu importe le point de la spline présent à cet endroit. Une telle contrainte dépend donc d'une seule variable libre et prend la forme :

$$\mathbf{P}(s(t),t) = \mathbf{A}$$

On remarque qu'elle ne dépend que des positions des points de contrôle et des valeurs des variables libres, ainsi la contrainte est de type holonome.

Si l'on définit l'indice k, de la variable libre utilisée ($s(t)$), dans le vecteur $\mathbf{s}$ des variables libres du système matériel, l'équation de contrainte $\mathbf{g}$ résultante est :

$$\mathbf{g}(\mathbf{q}(t),\mathbf{s}(t),t) = \mathbf{P}(s_k(t),t) - \mathbf{A} = \mathbf{0}_{(\mathbf{3})}$$

L'équation de contrainte étant de type holonome, le schéma de Baumgarte (3.2) permet d'obtenir l'équation dynamique de la contrainte qui est insérée dans le système en calculant

$$\begin{aligned}
\dot{\mathbf{g}}(\mathbf{q}(t),\mathbf{s}(t),t) &= \dot{\mathbf{P}}(s_k(t),t) = \frac{d}{dt}\left(\sum_{i=1}^{n}\mathbf{q}_i(t)b_i(s_k(t))\right) = \sum_{i=1}^{n}\left[\dot{\mathbf{q}}_i(t)b_i(s_k(t)) + \mathbf{q}_i(t)b_i'(s_k(t))\dot{s_i}(t)\right] \\
\ddot{\mathbf{g}}(\mathbf{q}(t),\mathbf{s}(t),t) &= \sum_{i=1}^{n}\left[\ddot{\mathbf{q}}_i(t)b_i(s_k(t)) + 2\dot{\mathbf{q}}_i(t)b_i'(s_k(t))\dot{s_i}(t) + \mathbf{q}_i(t)b_i''(s_k(t))\dot{s_k}(t)^2 + \mathbf{q}_i(t)b_i'(s_k(t))\ddot{s_k}(t)\right]
\end{aligned} \tag{3.23}$$

Ainsi, l'équation vectorielle de contrainte à ajouter au système d'équations est :

$$\begin{aligned}
\sum_{i=1}^{n}\left[\ddot{\mathbf{q}}_i(t)b_i(s_k(t)) + \mathbf{q}_i(t)b_i'(s_k(t))\ddot{s_k}(t)\right] &= -\sum_{i=1}^{n}\left[2\dot{\mathbf{q}}_i(t)b_i'(s_k(t))\dot{s_k}(t) + \mathbf{q}_i(t)b_i''(s_k(t))\dot{s_k}(t)^2\right] \\
&- \frac{2}{\delta t}\sum_{i=1}^{n}\left[\dot{\mathbf{q}}_i(t)b_i(s_k(t)) + \mathbf{q}_i(t)b_i'(s_k(t))\dot{s_k}(t)\right] \\
&- \frac{1}{\delta t^2}\left(\sum_{i=1}^{n}\mathbf{q}_i(t)b_i(s_k(t)) - A\right)
\end{aligned} \tag{3.24}$$

Donc, les trois lignes de la matrice L_g se remplissent à l'aide des coefficients $b_i(s_k(t))$ et les trois lignes de la matrice L_{gs} se remplissent avec les termes $\mathbf{q}_i(t)b_i'(s_k(t))$. Les trois entrées du vecteur E_g récupèrent l'ensemble des termes de la partie droite de l'équation (3.24).

On remarque que les matrices L_g et L_gs sont variables au cours du temps et doivent donc être recalculées à chaque itération.

3.3.2.2 Contrainte de tangente glissante

Pour certaines applications, il peut être utile de pouvoir contraindre un point mobile de la spline à avoir une tangente fixe $\mathbf{T_0}$. Cette contrainte est relative à une seule variable libre $s_k(t)$ (indicée par k dans le vecteur $\mathbf{s}$ des variables libres) et prend la forme vectorielle :

$$\mathbf{g}(\mathbf{q}(t),\mathbf{s}(t),t) = \mathbf{T}(s_k(t),t) - \mathbf{T_0} = \frac{d\mathbf{P}}{ds}(s_k(t),t) - \mathbf{T_0} = \mathbf{0}_{(\mathbf{3})}$$

L'équation de contrainte étant également de type holonome, le schéma de Baumgarte (3.2) permet d'obtenir l'équation dynamique de la contrainte qui est insérée dans le système en calculant :

$$\begin{aligned}
\dot{\mathbf{g}}(\mathbf{q}(t),\mathbf{s}(t),t) &= \dot{\mathbf{T}}(s_k(t),t) = \frac{d}{dt}\left(\sum_{i=1}^{n}\mathbf{q}_i(t)b_i'(s_k(t))\right) = \sum_{i=1}^{n}\left[\dot{\mathbf{q}}_i(t)b_i'(s_k(t)) + \mathbf{q}_i(t)b_i''(s_k(t))\dot{s_k}(t)\right] \\
\ddot{\mathbf{g}}(\mathbf{q}(t),\mathbf{s}(t),t) &= \sum_{i=1}^{n}\left[\ddot{\mathbf{q}}_i(t)b_i'(s_k(t)) + 2\dot{\mathbf{q}}_i(t)b_i''(s_k(t))\dot{s_k}(t) + \mathbf{q}_i(t)b_i'''(s_k(t))\dot{s_k}(t)^2 + \mathbf{q}_i(t)b_i''(s_k(t))\ddot{s_k}(t)\right]
\end{aligned} \tag{3.25}$$

Ainsi, l'équation vectorielle de contrainte à ajouter au système d'équations est :

$$\begin{aligned}\sum_{i=1}^{n}\left[\ddot{\mathbf{q}}_i(t)b_i'(s_k(t))+\mathbf{q}_i(t)b_i''(s_k(t))\ddot{s_k}(t)\right] &= -\sum_{i=1}^{n}\left[2\dot{\mathbf{q}}_i(t)b_i''(s_k(t))\dot{s_k}(t)+\mathbf{q}_i(t)b_i'''(s_k(t))\dot{s_k}(t)^2\right]\\ &- \frac{2}{\delta t}\sum_{i=1}^{n}\left[\dot{\mathbf{q}}_i(t)b_i'(s_k(t))+\mathbf{q}_i(t)b_i''(s_k(t))\dot{s_k}(t)\right]\\ &- \frac{1}{\delta t^2}\left(\sum_{i=1}^{n}\mathbf{q}_i(t)b_i'(s_k(t))-A\right)\end{aligned} \tag{3.26}$$

Les trois lignes de la matrice L_g se remplissent donc à l'aide des coefficients $b_i'(s_k(t))$ et les trois lignes de la matrice L_{gs} se remplissent avec les termes $\mathbf{q}_i(t)b_i''(s_k(t))$. Les trois entrées du vecteur E_g récupèrent l'ensemble des termes de la partie droite de l'équation (3.26).

On peut faire la même remarque que pour la contrainte de point glissant sur le fait que les matrices L_g et L_{gs} ne sont pas constantes. Cette contrainte demande donc de renseigner les matrices à chaque itération.

3.3.2.3 Contrainte de direction de tangente glissante

La contrainte de tangente glissante impose à un point variable de la courbe de posséder un vecteur tangent fixe. Cette contrainte peut s'avérer être un peu trop forte et apporter trop d'énergie dans le système. On propose pour cela une contrainte qui impose simplement la direction de la tangente. Ainsi, la tangente du point garde un degré de liberté et peut donc plus facilement réaliser la contrainte qui lui est demandée.

Ainsi, soit T_0 un vecteur indiquant la direction souhaitée, $\mathbf{u}$ et $\mathbf{v}$ deux vecteurs unitaires orthogonaux formant avec T_0 une base orthonormée. Soit s_k la variable libre, du vecteur $\mathbf{s}$, impliquée, la contrainte s'écrit alors sous la forme de deux équations de contraintes atomiques orthogonales glissantes :

$$g_1(\mathbf{q}(t),\mathbf{s}(t),t) = \mathbf{T}(s_k(t),t).\mathbf{u} = \frac{d\mathbf{P}}{ds}(s_k(t),t).\mathbf{u} = 0$$

$$g_2(\mathbf{q}(t),\mathbf{s}(t),t) = \mathbf{T}(s_k(t),t).\mathbf{v} = \frac{d\mathbf{P}}{ds}(s_k(t),t).\mathbf{v} = 0$$

Détaillons les calculs pour la première équation (sachant que le procédé s'applique également pour la seconde équation). L'équation est de type holonome, on utilise donc le schéma de Baumgarte (3.2) pour déterminer l'équation à intégrer dans le système :

$$\begin{aligned}&\ddot{g_1}+\frac{2}{\delta t}\dot{g_1}+\frac{1}{\delta t^2}g_1 = 0\\ \Leftrightarrow\ &\sum_{i=1}^{n}\left[\ddot{\mathbf{q}}_\mathbf{i}(t)b_i'(s_k(t))+\mathbf{q}_\mathbf{i}(t)b_i''(s_k(t))\ddot{s_k}(t)\right].\mathbf{u} = -\sum_{i=1}^{n}\left[2\dot{\mathbf{q}}_\mathbf{i}(t)b_i''(s_k(t))\dot{s_k}(t)+\mathbf{q}_\mathbf{i}(t)b_i'''(s_k(t))\dot{s_k}(t)^2\right].\mathbf{u}\\ &- \frac{2}{\delta t}\sum_{i=1}^{n}\left[\dot{\mathbf{q}}_\mathbf{i}(t)b_i'(s_k(t))+\mathbf{q}_\mathbf{i}b_i''(s_k(t))\dot{s_k}(t)\right].\mathbf{u}\\ &- \frac{1}{\delta t^2}\sum_{i=1}^{n}\left[\mathbf{q}_\mathbf{i}(t)b_i'(s_k(t))\right].\mathbf{u}\end{aligned}$$

La partie gauche de cette équation permet de renseigner les matrices L_g et L_{gs} alors que la partie droite informe le vecteur E_g.

Par exemple, la ligne de la L_{gs} est relative à l'ensemble des variables libres. Or, cette contrainte ne dépend que d'une seule variable libre s_k. Donc, seul le coefficient de la ligne relatif à cette variable s_k est non nul, il vaut : $\sum_{i=1}^{n}\mathbf{q}_\mathbf{i}(t)b_i''(s_k(t)).\mathbf{u}$. C'est le produit scalaire entre le vecteur courbure de la spline évalué en $s_k(t)$ et le vecteur $\mathbf{u}$.

3.3.2.4 Contrainte de courbure glissante

De la même manière que pour la tangente glissante, on peut souhaiter avoir une contrainte glissante sur la courbure d'une spline 1D. Cette contrainte est également relative à une seule variable libre $s_k(t)$ (indicé par k dans le vecteur $\mathbf{s}$ des variables libres) et prend la forme vectorielle :

$$\mathbf{g}(\mathbf{q}(t), \mathbf{s}(t), t) = \mathbf{C}(s_k(t), t) - \mathbf{C_0} = \frac{d^2\mathbf{P}}{ds^2}(s_k(t), t) - \mathbf{C_0} = \mathbf{0_{(3)}}$$

où $\mathbf{C_0}$ est le vecteur constant de courbure souhaitée.

Les calculs étant similaires à ceux rencontrés pour les contraintes de point glissant et de tangente glissante, nous ne les détaillons pas.

Il est intéressant de voir que chacune de ces contraintes se base sur une donnée de référence constante (un point pour une contrainte de point fixe, un vecteur pour une contrainte de tangente ou courbure...). Ainsi, la contrainte souhaite simplement que la courbe approche cette valeur fixe. On peut imaginer rendre dynamique cette donnée en la liant à un objet (la même spline ou un autre objet).

3.3.2.5 Contrainte glissante liée

Une contrainte glissante liée est une contrainte glissante dont la donnée de référence est constante relativement à un objet et non plus de façon statique dans la scène. Autrement dit, dans le cas d'une contrainte de point glissant liée, on obtient l'équation suivante :

$$\mathbf{g}(\mathbf{q}(t), \mathbf{s}(t), t) = \mathbf{P}(s_k(t), t) - \mathbf{P_0}(t) = \mathbf{0_{(3)}}$$

où $\mathbf{P_0}(t)$ est un point de référence fixe par rapport à un objet mais mobile puisque cet objet peut potentiellement se déplacer. Ainsi, l'équation de contrainte prend en considération la dynamique de ce point de référence.

Ce point de référence peut être lié à la même spline dynamique que le point glissant. Dans ce cas, la contrainte est entièrement gérée par la spline dynamique. L'équation de contrainte s'écrit alors :

$$\mathbf{g}(\mathbf{q}(t), \mathbf{s}(t), t) = \mathbf{P}(s_k(t), t) - \mathbf{P}(s_{fixe}, t) = \mathbf{0_{(3)}}$$

où s_{fixe} désigne l'abscisse du point de la spline par lequel le point glissant doit passer. Ce type de contrainte permet par exemple de simuler un nœud de pendu.

On peut très bien imaginer un objet de référence différent de la spline. Dans ce cas, le point glissant serait contraint de passer par un point donné d'un autre objet. Dans cette hypothèse, la contrainte lie deux objets dynamiques différents. Ces deux objets doivent donc être intégrés dans un même système matériel, une articulation (cf. section (3.2)). Ce type de contrainte peut être pratique pour la simulation de suture.

3.3.2.6 Contrainte glissante double

Une contrainte glissante double est une contrainte liant deux données glissantes. Par exemple, une contrainte de point glissant lie un point glissant à un point fixe de l'espace, une contrainte de point glissant double transforme le point fixe de l'espace en point glissant :

$$\mathbf{P}(s_{k1}(t), t) = \mathbf{P}(s_{k2}(t), t)$$

où s_{k1} est la variable libre relative au premier point contraint et s_{k2} est la variable libre relative au second point contraint.

Ainsi, on obtient une contrainte vectorielle qui dépend de deux variables libres différentes :

$$\mathbf{g}(\mathbf{q}(t), \mathbf{s}(t), t) = \mathbf{P}(s_{k1}(t), t) - \mathbf{P}(s_{k2}(t), t) = \mathbf{0_{(3)}}$$

Le développement des équations est similaire au cas du point glissant, à ceci près qu'il faut le faire pour les deux variables. Ainsi, les trois lignes de contrainte dans les matrices L_g et L_{gs} possèdent deux fois plus de coefficients.

Les deux points glissants peuvent être relatifs à la même spline, dans ce cas, la contrainte est entièrement gérée par la spline. Une telle contrainte permet par exemple d'imposer le fait que deux points dynamiques d'une même courbe soient confondus. Ceci permet donc de fermer la courbe à un endroit et donc de créer une boucle. Cette contrainte peut donc être très utile dans certaines applications comme la simulation de nœud pour la chirurgie.

On peut également imaginer une contrainte liant des données glissantes appartenant à deux splines dynamiques différentes. Dans ce cas, les deux splines doivent être simulées au sein d'une articulation pour prendre correctement en compte la liaison.

Il est également possible d'utiliser cette technique de contrainte glissante double avec les contraintes de tangente ou de courbure. Si, par exemple, on simule un point glissant double combiné avec une tangente glissante double, on obtient la simulation d'une boucle possédant une certaine continuité à la jonction.

Les contraintes glissantes sont relatives à une donnée mobile sur une spline. Or la spline est définie sur un intervalle paramétrique fermé, ce qui impose des conditions de validités sur les contraintes glissantes.

3.3.2.7 Condition de validité d'une contrainte glissante

Dans l'ensemble des contraintes présentées, il faut noter que les variables libres sont des abscisses paramétriques, donc leur domaine de validité est borné par le type de spline. Si la valeur d'une variable libre sort de ce domaine après une étape d'intégration numérique, soit on supprime la contrainte du système et on considère qu'elle n'existe plus, soit on intégre une équation unilatérale avant l'intégration numérique pour s'assurer que la variable libre est bien dans le domaine de validité de la spline. Cependant, cette contrainte s'écrit comme :

$$C(s_k(t), t) = s_k(t) - \text{borneInf} \quad \text{ou} \quad C(s_k(t), t) = s_k(t) - \text{borneSup}$$

elle est donc de type holonome. Or, l'utilisation d'un schéma de Baumgarte n'assure pas la contrainte, il y a donc un risque que la variable libre soit quand même hors limite après la phase d'intégration numérique. Il est préférable de gérer cette contrainte unilatérale avec une méthode plus robuste comme une méthode projective. Donc, après l'intégration numérique, on vérifie l'équation de contrainte et, en cas de violation, on projette la position et la vitesse de cette variable libre sur les bornes du domaine valide.

Dans notre implémentation, seule la première méthode a été validée. A savoir que si une contrainte glissante sort du domaine de validité, elle est supprimée.

La classe des contraintes glissantes offre des possibilités supplémentaires intéressantes en simulation notamment au travers des points glissants. Ce type de contraintes permet de simuler le coulissement d'une corde ou d'un fil au travers d'un obstacle. En réalité, ce type de manipulation introduit des frottements entre les deux objets en contact. Nous proposons donc d'introduire un frottement au niveau des points glissants.

3.3.3 Introduction d'un frottement sur un point glissant

Il faut remarquer qu'une contrainte de point glissant contrôle la position d'un point de la spline. On peut donc introduire un frottement local à cet endroit pour modéliser une éventuelle interaction avec un autre objet. Pour cela, on peut employer la technique décrite en section (3.3.1.2) qui consiste à modifier l'équation de glissement pour manipuler la dynamique des variables libres. Cependant, l'utilisation de cette technique ne nous permet pas de maîtriser le frottement induit et peut donc difficilement être validée par des comparaisons au réel. Pour cela, nous proposons une technique permettant d'intégrer des frottements secs répondant à des équations plus physiques que celles employées dans la première technique.

Nous proposons une technique basée sur les frottements de Coulomb (cf. annexe (I)). Avant de commencer, on introduit quelques notations. On définit la tangente $\mathbf{T}$ à la spline au point glissant par

$$\mathbf{T}(s(t), t) = \frac{d\mathbf{P}}{ds}(s(t), t) = \sum_{i=1}^{n} \mathbf{q_i}(t) b_i'(s(t))$$

On définit le vecteur $\mathbf{t}$ en tant que le vecteur $\mathbf{T}$ normalisé. On introduit également la vitesse $\mathbf{V}$ du point glissant :

$$\mathbf{V}(s(t),t) = \frac{d\mathbf{P}}{dt}(s(t),t) = \sum_{i=1}^{n} \dot{\mathbf{q}}_\mathbf{i}(t) b_i(s(t)) + \mathbf{q}_\mathbf{i}(t) b_i'(s(t)) \dot{s}(t)$$

Ensuite, on détermine les composantes tangentielles et normales d'un vecteur $\mathbf{x}$ (quelconque) par les relations :

$$\begin{array}{rcl} \mathbf{x_t} & = & (\mathbf{x}.\mathbf{t})\mathbf{t} \\ \mathbf{x_n} & = & \mathbf{x} - \mathbf{x_t} \end{array} \tag{3.27}$$

Comme Coulomb étudie le frottement en séparant le cas statique du cas dynamique, nous commençons par déterminer le cas dans lequel se trouve le point glissant. Pour cela, il suffit de calculer la composante tangentielle de la vitesse $\mathbf{V_t}$ qui représente la vitesse de glissement. Si, pour un ξ petit, on a :

$$|\mathbf{V_t}| < \xi$$

la vitesse de glissement est alors considérée comme nulle. Le point glissant est donc dans le cas de frottement statique. Sinon, le point glissant se déplace, il est donc dans le cas dynamique.

Frottement statique :

Dans le cas statique, il faut déterminer la force de réaction du plan pour pouvoir conclure. Afin d'obtenir cette force, on considère le point glissant dans une configuration d'adhérence. Ainsi, il est fixé à l'abscisse paramétrique de la spline et ne doit pas y bouger. On transforme donc la contrainte de point glissant en une contrainte de point fixe, et on résoud le système d'équations dynamiques. Les trois multiplicateurs de Lagrange $\boldsymbol{\lambda}$ relatifs à la contrainte de point fixe nous fournissent la force due à la contrainte (cf. annexe (H)).

Ensuite, il suffit de vérifier si le point glissant est en adhérence ou s'il entre en glissement. Pour cela, on utilise la relation de Coulomb :

$$|\boldsymbol{\lambda}_\mathbf{t}| < \mu_s |\boldsymbol{\lambda}_\mathbf{n}|$$

Si la condition est vérifiée, alors la force est dans le cône d'adhérence. Donc, le point est considéré comme fixe jusqu'à la prochaine étape.

Si la condition n'est pas vérifiée, la force de frottement n'est pas suffisante pour contenir le mouvement, le point contraint doit donc se déplacer. Pour cela, on transforme le point fixe en point glissant, et on introduit la force de frottement pseudo-statique suivante dans le système :

$$\mathbf{F_f} = -\mu_s |\boldsymbol{\lambda}_\mathbf{n}| \frac{\mathbf{F_t}}{|\mathbf{F_t}|}$$

où $\mathbf{F_t}$ est la composante tangentielle de l'ensemble des forces appliquées au point glissant.

Le fait de transformer le point fixe en point glissant fait apparaître la dynamique de la variable libre associée à cette contrainte glissante. La valeur de la variable libre est donnée par l'abscisse paramétrique du point fixe et sa vitesse est considérée comme nulle puisque le point était adhérent jusqu'à présent, donc immobile.

On résoud alors le système une seconde fois pour obtenir les nouvelles positions et vitesses du système matériel et ainsi animer le point glissant.

Frottement dynamique :

Dans le cas d'un frottement dynamique, il suffit de considérer le point glissant avec une force de frottement opposée à la vitesse de glissement. Sachant que la force, due à la contrainte de point glissant, est donnée par les trois multiplicateurs de Lagrange relatifs à cette contrainte (cf. annexe (H)), on a :

$$\mathbf{F_f} = -\mu_d |\boldsymbol{\lambda}_\mathbf{n}| \frac{\mathbf{V_t}}{|\mathbf{V_t}|}$$

Il faut remarquer que si le système dynamique employé définit la contrainte de point glissant comme une contrainte parfaite, alors la force due à la contrainte ne travaille pas. Elle ne produit donc aucune composante tangentielle, ainsi $\boldsymbol{\lambda_n} = \boldsymbol{\lambda}$. Si maintenant le système dynamique employé est le système intégrant une équation de contrainte qui produit un travail (cf. section(3.3.1.2)), la force induite par la contrainte n'est plus normale à la contrainte. Il faut donc calculer explicitement $\boldsymbol{\lambda_n}$ par une projection comme cela est détaillé en début de section.

Dans le cas d'un frottement dynamique, il n'y a pas d'ambiguïté sur la nature du point contraint, il est considéré comme mobile (contrainte de point glissant). On introduit simplement la force de frottement définie par Coulomb. Le cas dynamique disparaît dès lors que la vitesse du point glissant faiblit et passe en dessous du seuil ξ. Dans ce cas, l'algorithme le détecte en tout début de procédure et oriente le point glissant sur un cas de frottement statique.

Dans cette proposition, liée au modèle de frottement de Coulomb, le point glissant possède deux états : glissant (contrainte de point glissant) ou adhérent (contrainte de point fixe). On met donc en place une structure de donnée adéquate pour gérer un frottement en intégrant la possibilité de changer d'état en cours de simulation (un simple booléen permet de connaître l'état courant de la contrainte). L'information relative à l'état du frottement est portée par la contrainte, ainsi il est possible de gérer plusieurs points glissants avec frottement.

3.4 Tests et résultats

Nous présentons dans cette section les différents résultats et tests obtenus sur les simulations de splines contraintes.

Nous établissons le plan de cette section en suivant le plan d'étude des contraintes, à savoir les contraintes simples sur le modèle spline dynamique 1D puis, les contraintes externes prises en charge par l'architecture proposée à la section (3.2) et enfin les contraintes glissantes.

Les tests ont été effectués sur un Pentium IV 2.4GHz avec 512Mo de mémoire vive.

3.4.1 Contraintes simples

Cette sous-section discute des résultats obtenus sur des simulations incluant des contraintes de point fixe, de point contraint sur un axe ou un plan. De plus, un dernier point est soulevé par la proposition d'éditer directement les contraintes.

Visuellement, une contrainte de point fixe est modélisée par une sphère verte transparente, la contrainte qui impose un point à être sur un axe est tracée au niveau du point contraint par une portion de l'axe en bleu transparent. La contrainte de point dans un plan est tracée au niveau du point contraint par une portion du plan en rouge transparent.

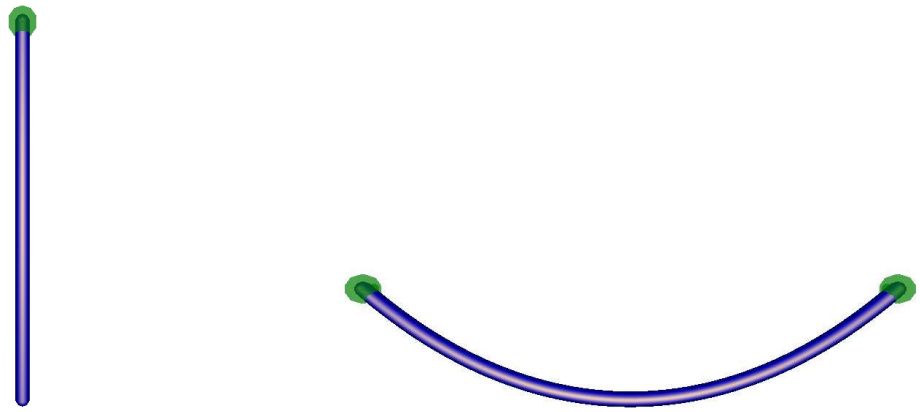
Numériquement, les différentes simulations ont été réalisées à l'aide de la méthode d'intégration numérique explicite Runge Kutta 4.

3.4.1.1 Contrainte de point fixe

Le premier test consiste à simuler une spline, de type B-spline uniforme cubique, définie à l'aide de 20 points de contrôle, de masse 50 g, de coefficient d'amortissement 0.03, d'énergies de déformations en élongation et flexion. Ces énergies sont discrètes, créées par des ressorts d'élongation de raideur 100 et des ressorts de courbure de raideur 100 également. Sachant que la discrétisation, pour placer les ressorts, est de 3 points intermédiaires par segment spline.

Le modèle sans contrainte est simulé avec un temps de calcul d'une itération mécanique de 0.81 ms. Le modèle avec une contrainte de point fixe, en fixant l'une des deux extrémités, se simule en 1.15 ms (état stable présenté sur le schéma (a) de la figure (3.6)). La même simulation en fixant les deux extrémités se simule en 1.55 ms (état stable présenté sur le schéma (b) de la figure (3.6)). Et en fixant trois points du modèle, le temps de simulation d'une itération est de 2.01 ms.

Le second test permet de mettre en évidence la complexité quadratique de la résolution d'un système de spline dynamique contraint. Afin de mieux appréhender la complexité de la résolution du système



(a) Une seule extrémité contrainte (b) Les deux extrémités contraintes

FIG. 3.6 – Contraintes de point fixe sur une spline 1D

en fonction du nombre de contraintes, nous présentons une étude menée sur le même modèle excepté le nombre de points de contrôle qui est de 50. La spline dynamique possède donc 150 degrés de liberté. L'étude porte sur les temps de calcul d'une itération de la simulation mécanique (pour calculer l'animation pendant 1ms virtuelle) en fonction du nombre de contraintes atomiques (cf. définition en section(173)) présentes dans le système. Voici le tableau des mesures relevées lors des différentes simulations :

Nombre de points fixes	Nombre de contraintes atomiques	Temps de simulation (en ms) pour 1ms virtuelle
0	0	1.75
1	3	3.73
2	6	5.74
3	9	7.88
4	12	10.21
5	15	12.81
6	18	15.43
8	24	21.38
10	30	28.07
12	36	35.62
14	42	43.73
20	60	73.09
25	75	104.86
30	90	142.91
35	105	187.23
40	120	239.43
45	135	301.05
50	150	385.94

(3.28)

En remarquant que la dernière ligne introduit 150 contraintes atomiques dans un système possédant 150 degrés de liberté. Cette dernière simulation est donc entièrement contrainte, la courbe est complétement figée.

Le graphique correspondant au tableau est présenté sur la figure (3.7).

On retrouve bien une allure quadratique des temps de calcul en fonction du nombre de contraintes atomiques comme cela a été démontré dans la section (3.1.1).

Voyons maintenant les résultats obtenus pour la contrainte qui oblige un point du modèle à être sur un axe.

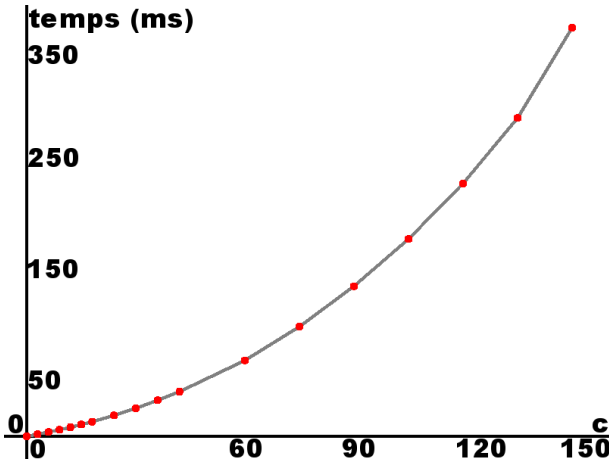


FIG. 3.7 – Temps de calcul en fonction du nombre (c) de contraintes

3.4.1.2 Contraindre un point sur un axe

Les deux tests proposés portent sur la même spline dynamique à savoir, une B-spline uniforme cubique de 20 points de contrôle, de masse 50 g et de coefficient d'amortissement 0.03. Les énergies de déformation sont discrètes avec des ressorts d'élongation et de courbure de raideur 100. Sachant que la discrétisation, pour placer les ressorts, est également de trois points intermédiaires par segment spline.

Pour bien comprendre la configuration des contraintes, nous détaillons également la position au départ de la spline. Elle est rectiligne entre les points $\begin{pmatrix} -0.35 & 0.08 & 0.0 \end{pmatrix}$ et $\begin{pmatrix} 0.15 & 0.08 & 0.0 \end{pmatrix}$. Autrement dit, la spline mesure moins de 50 cm et elle est tendue sur l'axe des x (elle mesure moins de 50 cm puisque ce sont les coordonnées des points de contrôle extrémité. En réalité, la spline mesure au départ 44.7 cm).

Le premier test consiste simplement à contraindre le point milieu de la spline à être sur l'axe des x. Comme l'abscisse paramétrique d'une B-spline uniforme cubique de 20 points de contrôle commence à 3 (cf. section (1.2.1.9)) et se termine[17] à 20^-. Donc l'abscisse paramétrique du point milieu est 11.5. Un vecteur directeur de l'axe de contrainte est $\begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$ et un point de cet axe est le point milieu de la courbe au début de la simulation (ainsi, la contrainte est vérifiée dès le début).

Le temps de calcul d'une itération de la simulation est de 1.11 ms. L'état stable de la simulation peut être observé sur la figure (3.8).

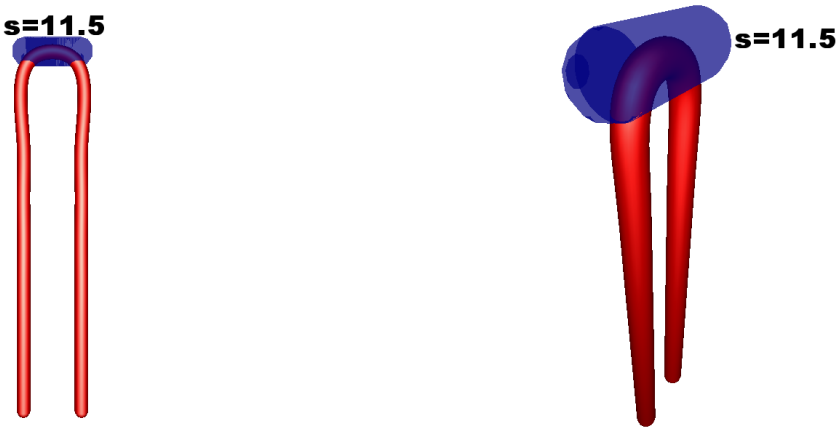
Le second test met en place trois contraintes d'axe (chaque contrainte d'axe fixe deux degrés de liberté, l'ensemble des trois en fixe donc six. Par comparaison, deux points fixes demandent le même nombre de contraintes atomiques). Avec la particularité que les trois axes choisis forment une base orthogonale. On choisit donc de contraindre une extrémité de la spline (d'abscisse paramétrique 3) sur l'axe des x, le point milieu (d'abscisse paramétrique 11.5) sur l'axe y et l'autre extrémité (d'abscisse paramétrique 20^-) sur l'axe des z.

Le temps de calcul d'une itération de la simulation est de 1.68 ms. On retrouve à peu près le même temps de simulation que la scène contrainte par deux points fixes vue dans la section précédente. L'état stable de la simulation, des trois contraintes d'axes, peut être observé sur la figure (3.9).

3.4.1.3 Contraindre un point dans un plan

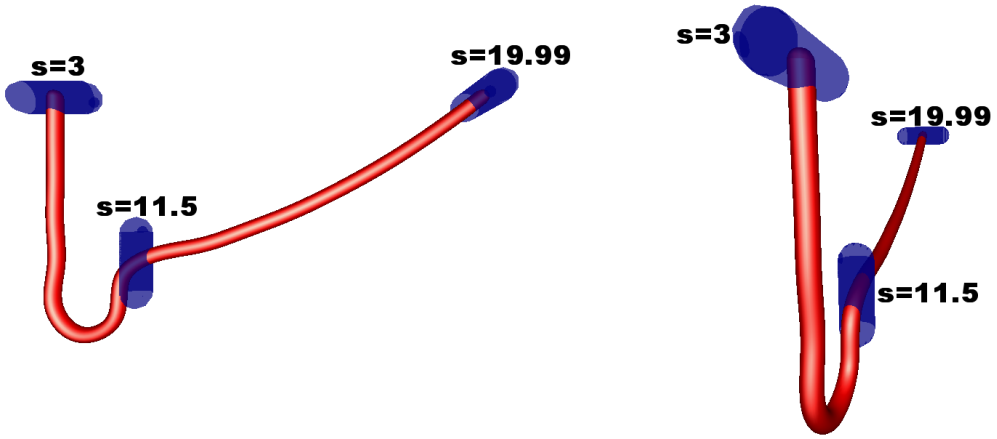
Les simulations qui mettent en valeur la contrainte de plan sont toutes basées sur la même spline dynamique, celle détaillée au début de la section précédente (cf. section (3.4.1.2)). La spline est toujours rectiligne et alignée sur l'axe des x au départ, seule sa longueur varie d'une simulation à l'autre.

[17] l'intervalle est ouvert de ce côté, donc la valeur 20 n'est pas valable. Le signe $-$ désigne simplement que la valeur est légèrement en dessous de 20



(a) Vue de face (b) Vue de profil

FIG. 3.8 – Une contrainte d'axe



(a) Vue de face (b) Vue de profil

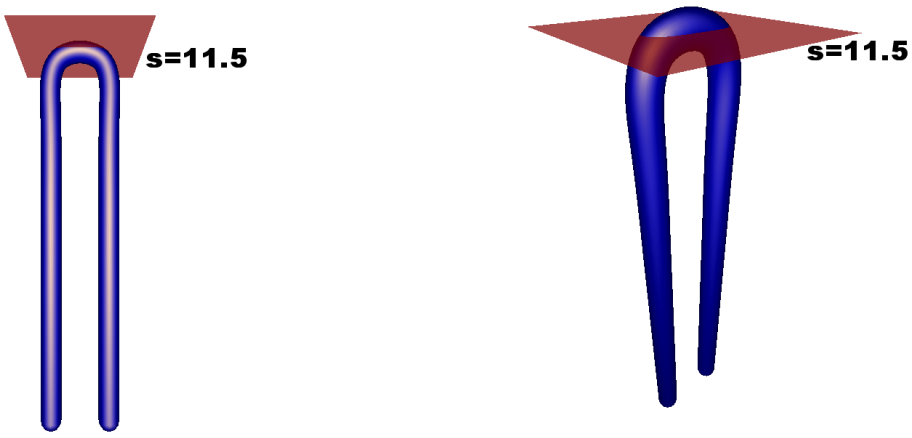
FIG. 3.9 – Trois contraintes d'axes orthogonaux

Le premier test met en valeur une simple contrainte de plan appliquée au point milieu d'une spline de longueur 35.7 cm. L'abscisse paramétrique du point contraint est donc 11.5. Le plan est, quant à lui, défini par sa normale $\begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^T$ et un point inclus dans ce plan qui n'est autre que le point milieu de la courbe au début de la simulation.

Cette simulation demande un temps de calcul de 0.83 ms par itération. L'état stable de la simulation est affiché sur la figure (3.10).

Le second test utilise une spline de 19.6 cm et contraint les points extrémités à être dans des plans orthogonaux ainsi que le point milieu à être dans un troisième plan orthogonal aux deux premiers. Ainsi, les trois points évoluent dans trois plans orthogonaux deux à deux. Le point d'abscisse paramétrique 3 est contraint par le plan de vecteur normal $\begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^T$ passant par ce point, le point d'abscisse paramétrique 11.5 est contraint par le plan de vecteur normal $\begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^T$ passant par lui et enfin, le point d'abscisse paramétrique 20^- est contraint par le plan de vecteur normal $\begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T$ passant par lui également.

Le temps de simulation de cette scène est de 1.05 ms. L'état stable de la simulation est affiché sur la figure (3.11).



(a) Vue de face (b) Vue de profil

FIG. 3.10 – Une contrainte de point dans un plan

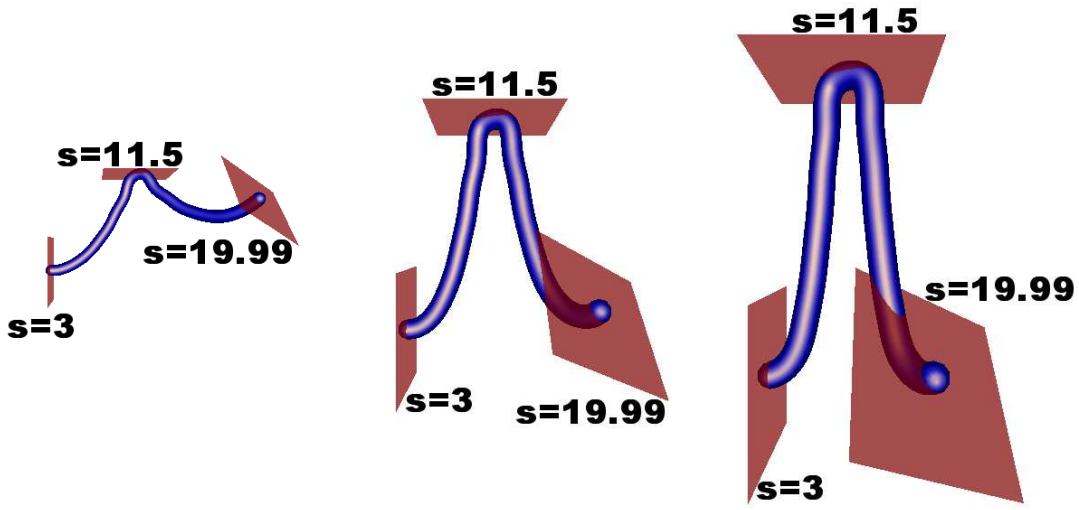
(a) Vue de face (b) Vue de profil

FIG. 3.11 – Trois contraintes de point dans un plan (trois plans orthogonaux)

Le dernier test de cette section met en évidence le fait qu'une contrainte de plan ne supprime qu'un seul degré de liberté, laissant au modèle la possibilité de se déplacer. Pour cela, on définit une spline de 22.3 cm contrainte de la même manière que le test précédent à savoir sur les extrémités et en son milieu. La première extrémité et le point milieu conservent la même contrainte, tandis que l'extrémité d'abscisse paramétrique 20^- (on nomme cette extrémité le point **A**) est contrainte par le plan de vecteur normal $\begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^T$ passant par **A**.

De cette manière, les trois points sont entraînés par leur poids et contraints dans leur plan respectif. Le point **A** tombe donc en vérifiant l'équation du plan, il se décale dans les x négatifs et également dans les z négatifs. Ainsi, il se rapproche de la caméra en entraînant la spline toute entière. Comme les plans de contrainte des deux autres points contraints leur permettent de se déplacer sur l'axe des z, ils n'empêchent pas la courbe de se déplacer vers la caméra.

Le temps de simulation de cette scène est de 1.06 ms. Différentes étapes de la simulation sont affichées sur la figure (3.12) pour montrer la dynamique de la spline et le mouvement vers la caméra.



(a) t=0.3s (b) t=1.5s (c) t=3.0s

FIG. 3.12 – Trois contraintes de plan créant un mouvement

3.4.1.4 Edition directe

Les contraintes mises en valeur jusqu'à présent sont définies de manière statique et restent fixes lors de la simulation. Pourtant, il peut être intéressant de modifier les caractéristiques des contraintes dynamiquement.

Nous avons mis en place un système permettant de modifier dynamiquement les données relatives aux contraintes. Par exemple, une contrainte de point fixe est définie par le point fixe de l'espace alors qu'une contrainte d'axe est définie par un point de l'axe et un vecteur directeur de cet axe. Le procédé mis en place permet de choisir la contrainte que l'on veut modifier et offre la possibilité de choisir également quelle donnée de cette contrainte on souhaite modifier. Ainsi, l'utilisateur a la possibilité de manipuler les contraintes par une technique d'édition directe. Pour le moment, cette technique utilise le clavier mais l'utilisation de la souris serait beaucoup plus intuitive et pratique. Cette amélioration est à l'étude.

3.4.2 Contraintes externes

La proposition de gestion des contraintes externes faites en section (3.2) permet d'articuler des modèles entre eux. Nous exposons ici quelques tests permettant de mettre en valeur cette proposition. On distingue les articulations d'objets faisant intervenir un seul type d'objet des articulations de plusieurs types d'objets.

3.4.2.1 Articulation de plusieurs objets de même type

Le premier test consiste à simuler plusieurs splines dynamiques ensembles et à définir des contraintes pour former trois anneaux déformables liés.

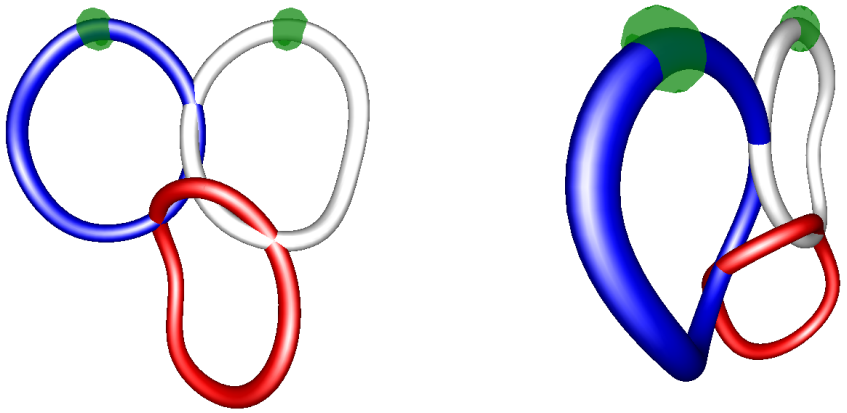
On définit donc trois splines, de type B-spline uniformes cubiques, de 10 points de contrôle chacune. Chaque spline pèse 50 g, posséde un coefficient d'amortissement de 0.05, mesure 40 cm et est alignée sur l'axe des x. Leurs déformations sont assurées par des énergies discrètes en élongation et en courbure. Les ressorts d'élongation ont une raideur de 100 et les ressorts de courbure ont une raideur de 300. La discrétisation du modèle, qui sert de support au ressort, définit trois points intermédiaires par segment spline.

Chacune des splines subit trois contraintes qui servent à fermer la courbe en assurant une continuité C^2 :

- une contrainte de point fixe entre les deux extrémités d'une même spline.
- une contrainte de tangente fixe sur les deux extrémités d'une même spline.
- une contrainte de courbure fixe sur les deux extrémités d'une même spline.

On définit également une contrainte de point fixe entre les couples de splines de manière à lier les anneaux ensembles (donc trois contraintes de point fixe). Enfin, on contraint deux splines à avoir un point fixe dans l'espace de manière à clouer les anneaux dans la scène. Le système possède donc 3×30 degrés de liberté et 42 contraintes atomiques accompagnées de leur multiplicateur de Lagrange respectif.

Le temps de calcul d'une itération de cette simulation est de 3.14 ms. L'état stable de cette scène est présenté sur la figure (3.13).



(a) Vue de face (b) Vue de profil

FIG. 3.13 – Trois anneaux déformables liés

Afin de mieux apréhender la définition d'une telle scène, voici une partie du script utilisé pour définir le système matériel :

```
ARTICULATION
 Name corpsArticule
[corps]
 BSplineUniformeCubique1D
 Name spline1
 ...
 NbPoints 10
 Extremite (-0.2 0.0 0.0) (0.2 0.0 0.0)
 Contrainte_point_fixe 0.000

[corps]
 BSplineUniformeCubique1D
 Name spline2
 ...
 Stretching ressorts_elongation 100 0
 Bending ressorts_courbure 300 0
 Contrainte_point_fixe 0.000

[corps]
 BSplineUniformeCubique1D
 Name spline3
 Masse 0.05
 AmortN{\oe}ud 0.05
```

```
...

[contraintes]
 Contrainte_point_fixe 1 spline2 1 spline2 0.000 6.999
 Contrainte_tangente_fixe 1 spline2 1 spline2 0.000 6.999
 Contrainte_courbure_fixe 1 spline2 1 spline2 0.000 6.999
 ...
 Contrainte_point_fixe 1 spline1 1 spline2 1.5 5.5
 Contrainte_point_fixe 1 spline1 1 spline3 3.0 5.9
 Contrainte_point_fixe 1 spline2 1 spline3 3.5 0.9
```

Il faut noter que les splines sont rectilignes au départ, les contraintes ne sont donc absolument pas vérifiées au début de la simulation. Le schéma de Baumgarte permet d'adoucir la vérification des contraintes et module donc les énergies mises en œuvre au démarrage d'une telle scène. Il peut être également noté que les deux contraintes qui fixent les anneaux dans la scène ne sont pas des contraintes définies par l'articulation mais par les corps eux-mêmes. Ainsi, il est tout à fait possible de mixer des contraintes internes et des contraintes externes.

Le second test met en valeur une articulation d'objets rigides formant un cycle fermé. Pour cela, on définit quatre pavés rigides de masse 1 g et d'amortissement 0.001. On définit alors cinq contraintes à l'aide d'une articulation. La première sert à fixer l'un des quatre pavés dans la scène. Les autres permettent de définir le cycle, chaque objet rigide étant lié à ses deux voisins directs, formant ainsi une chaîne fermée.

La simulation requiert un temps de calcul de 0.8 ms par itération. L'état stable de la simulation est présenté sur la figure (3.14).

Fig. 3.14 – Quatre objets rigides formant un cycle fermé

3.4.2.2 Articulation de différents modèles

Afin de montrer l'hétérogénéité d'une articulation, nous présentons maintenant un test mettant en œuvre quatre splines dynamiques, un tissu masses-ressorts et un objet rigide, le tout contraint pour former une balançoire.

Le corps rigide pèse 10 g et possède un coefficient d'amortissement de 0.01. Les quatre splines sont de type B-splines uniformes cubiques alignées sur l'axe des x avec 8 points de contrôle, une masse de 20 g, un coefficient d'amortissement de 0.015, une longueur de 28.5 cm et des ressorts d'élongation de raideur 200 (sur une discrétisation du modèle à l'aide de trois points intermédiaires par segment spline). Les quatre splines sont contraintes en interne à avoir une extrémité fixe dans la scène. Enfin, le tissu masses-ressorts pèse 5 g répartis sur 25 masses.

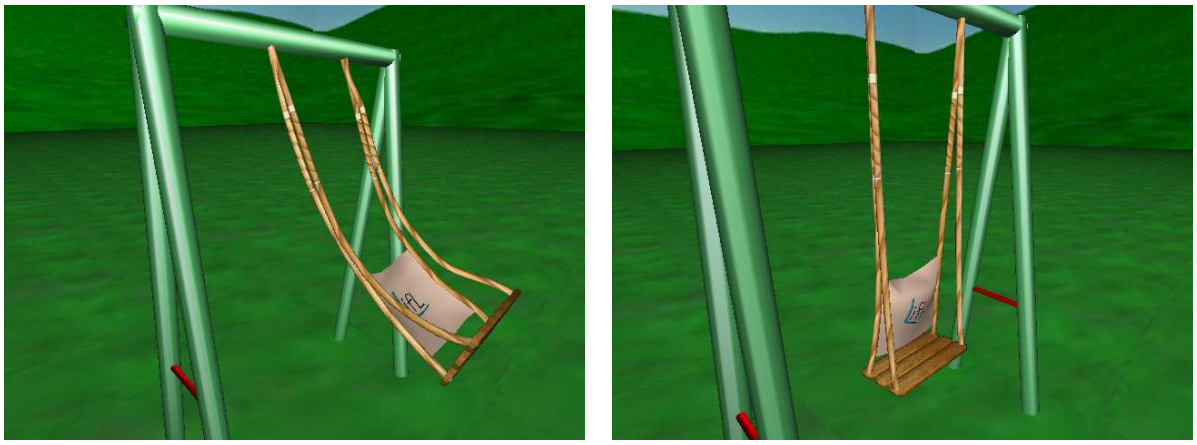
Ainsi, le système matériel est composé de 6 degrés de liberté pour le corps rigide, 96 degrés de liberté pour les splines et 75 degrés de liberté pour le tissu. Soit un total de 177 degrés de liberté.

Le système est complété par un ensemble de contraintes permettant de lier les objets entre eux pour former la balançoire :

- Chaque spline est fixée par une extrémité à la barre transversale de la balançoire (qui est un objet immobile de la scène). On obtient donc quatre contraintes de points fixes.
- Chaque spline est fixée par l'autre extrémité à l'un des quatre bords de le planche de bois (l'objet rigide) de la balançoire. On obtient donc quatre contraintes de point fixe supplémentaires.
- Les deux cordes du fond de la balançoire sont contraintes avec le tissu masses-ressorts qui forme le dossier. Sur chaque spline, on définit cinq contraintes de point fixe avec un bord du tissu. Ainsi, le tissu est fixé aux deux cordes arrières et, il est tendu entre ces cordes. Le placement du tissu demande donc 10 contraintes de point fixe supplémentaires.

Au final, cette scène requiert donc 177 degrés de liberté et 54 contraintes atomiques.

Le temps de calcul d'une itération de simulation pour cette scène complète est de 10.3 ms. La figure (3.15) montre un état de la simulation en balancement et l'état d'équilibre du système dynamique.



(a) En mouvement (b) Etat d'équilibre

FIG. 3.15 – Simulation d'une balançoire

On peut remarquer que la résolution du système demande de déterminer les valeurs des multiplicateurs de Lagrange. Ces valeurs se déterminent par la résolution d'un système d'équations de taille du nombre de contraintes atomiques. Cette résolution peut être faite par une méthode directe (le pivot de Gauss par exemple) ou une itérative (le bi-gradient conjugué par exemple). Nous proposons dans la section suivante une étude comparative de ces deux méthodes.

3.4.2.3 Optimisation

Ce petit test permet d'étudier les performances des méthodes de résolution de Gauss (implémenté par nos soins) et du bi-gradient conjugué (implémentation de la librairie MKL d'Intel[18]) lors de la résolution de l'équation matricielle pour le calcul des $\boldsymbol{\lambda}$. Ce système est de taille du nombre de contraintes atomiques présentes dans l'articulation. Nous testons donc les deux algorithmes différents sur une scène comportant deux splines de 50 points de contrôle alignés sur l'axe des x, de longueur 38.3 cm, de masse 10 g, et de coefficient d'amortissement 0.03. Des ressorts d'élongation de raideur 100 permettent de gérer les déformations des deux splines. Ces ressorts sont placés sur une discrétisation du modèle définie par les extrémités des segments splines et trois points intermédiaires sur chacun de ces segments.

L'une des deux splines est fixée à la scène par une extrémité à l'aide d'une contrainte définie par l'articulation entre cette spline et la scène. La simulation commence donc avec au moins trois contraintes atomiques. Puis, on pose une à une des contraintes de point fixe entre les deux splines pour mesurer les temps de calcul des deux méthodes. Sachant que la méthode itérative s'arrête sur un critère de précision,

[18] http ://www.intel.com/software/products/mkl/

on étudie cette méthode avec deux précisions différentes : 10^{-3} et 10^{-10}. Les résultats présentés dans le tableau suivant montrent les temps de calcul d'une itération en fonction du nombre (c) de contraintes :

c	Pivot de Gauss	Bi-gradient conjugué (précision 10^{-3})	Bi-gradient conjugué (précision 10^{-10})
3	5.73	5.85	5.86
6	6.39	6.55	6.58
9	6.93	7.01	7.05
15	8.03	8.07	8.04
18	8.65	8.58	8.62
21	9.03	8.88	8.89
27	9.81	9.36	9.51
33	10.87	10.01	10.22
45	13.02	10.92	11.25
60	16.76	12.17	12.56
90	28.95	14.39	15.14

Le graphique correspondant aux mesures est présenté sur la figure (3.16).

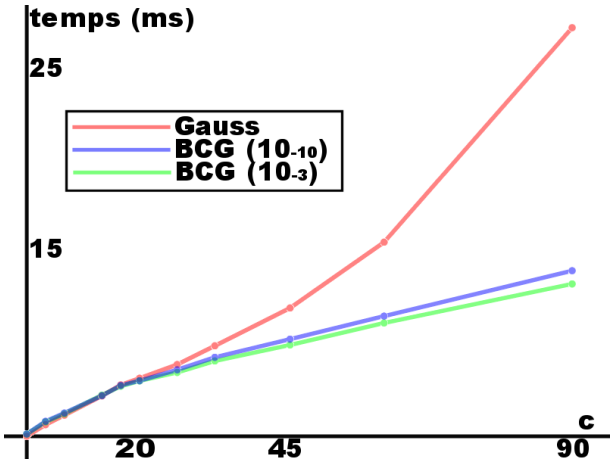


FIG. 3.16 – Graphique de comparaison des méthodes de résolution

En regardant les résultats obtenus, on peut affirmer que dans notre cas, la méthode du bi-gradient conjugué est plus adaptée que la méthode de Gauss pour les systèmes de plus de 15 contraintes atomiques. Cependant, comme les deux méthodes sont quasiment équivalentes en dessous de ce nombre, il est tout à fait envisageable de choisir définitivement la méthode itérative quelle que soient la taille du système à résoudre et la précision souhaitée.

3.4.3 Contraintes glissantes

Les contraintes de point glissant sont modélisées sur les figures par des sphères grises transparentes.

Le premier test met en évidence l'intérêt des points glissants pour la facilité de manipulation. Pour cela, on définit une B-spline uniforme cubique de 20 points de contrôle, de masse 50 g et de coefficient d'amortissement 0.03. Elle est initialisée sur l'axe des x avec une longueur 35.7 cm et des ressorts en élongation et flexion d'une raideur de 100 (les ressorts sont placés sur une discrétisation du modèle défini par les extrémités des segments splines et trois points intermédiaires par segment spline).

La spline est fixée dans l'espace au niveau de l'extrémité d'abscisse paramétrique 3. On pose également deux points glissants sur les points de la spline (au démarrage de la simulation) d'abscisse paramétrique 6 et 7.5. Le système matériel est donc défini par 60 degrés de liberté, 2 variables libres et

9 contraintes atomiques. L'utilisateur utilise alors le Phantom [19] pour manipuler une sonde et interagir avec les objets de la scène. Il a également la possibilité de saisir un objet en créant une liaison de type ressort entre la sonde et un point de l'objet.

Pour ce test, la machine employée est un Bi-xeon 3GHz avec 1024 Mo de mémoire vive et la méthode d'intégration numérique utilisée est Euler implicite[HMC01]. Le temps de calcul d'une itération peut donc varier si l'intégrateur a besoin de réduire le pas de temps d'intégration. Lorsque l'utilisateur n'interagit pas avec le modèle, le temps de calcul est de 4 ms. Lorsque l'utilisateur interagit fortement avec le modèle, le temps de calcul peut monter à 15 ms. En moyenne, ce temps de calcul est réduit à 5 ms.

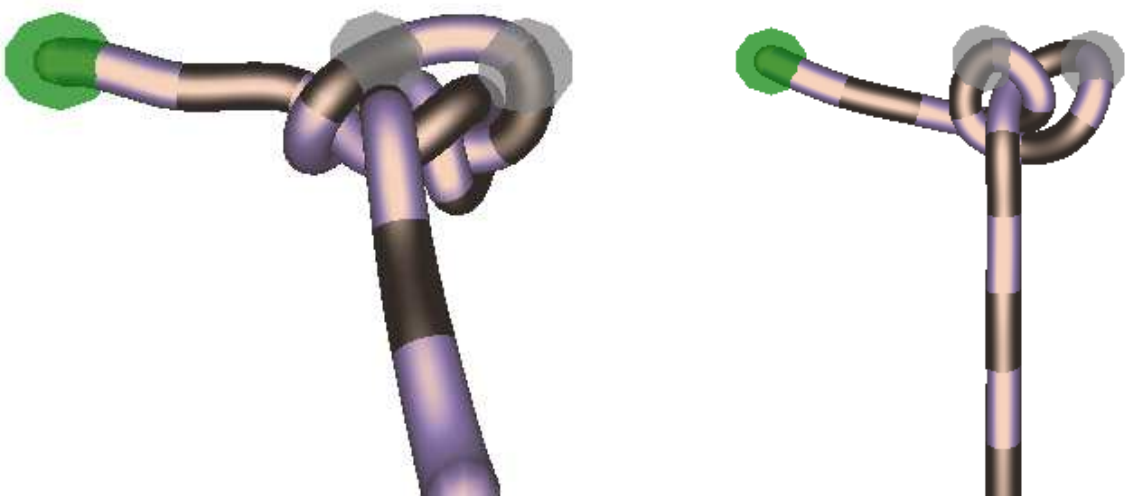


FIG. 3.17 – Manipulation d'une spline possédant des points glissants

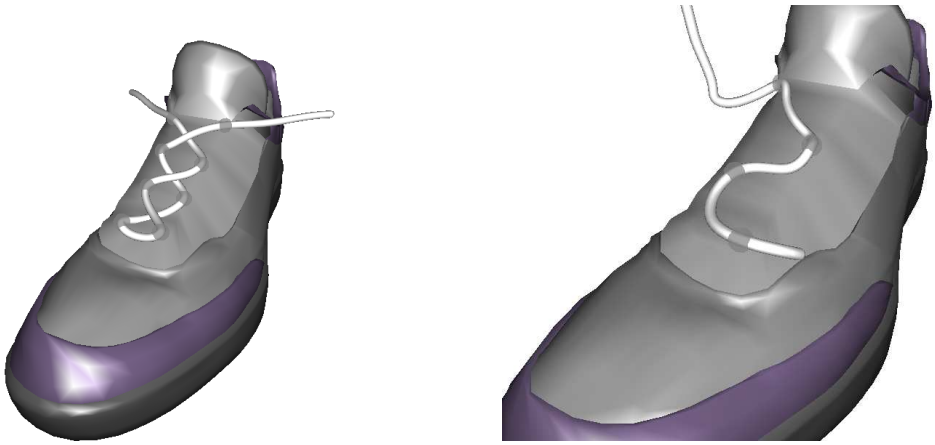
La figure (3.17) montre deux états de la simulation. On peut remarquer la présence d'un nœud plus ou moins complexe qui est le résultat de la manipulation.

Le second test met en valeur l'utilisation des points glissants dans une simulation de lacet de chaussure [LGM+04]. Pour cela, on définit une B-spline uniforme cubique à l'aide de 28 points de contrôle. Ces points de contrôle sont placés de manière à vérifier la configuration du lacet sur la chaussure. Les déformations en élongation du modèle sont gérées par des ressorts de raideur 300 (sur une discrétisation de la spline définie par les extrémités des segments splines et trois points intermédiaires sur chacun de ces segments). Si l'on place des ressorts de courbure, la configuration au repos du modèle déformable n'étant pas rectiligne, le modèle aurait tendance à se courber naturellement. Or, on suppose qu'un lacet ne possède pas d'énergie de flexion, mais uniquement des déformations très légères en élongation. La spline pèse 30 g, possède un coefficient d'amortissement de 0.03. Afin de gérer les interactions du modèle avec lui même, on définit une concentration en sphères de collision assez élevée puisque l'on choisit un facteur *densite* avec une valeur de 1 (généralement, ce nombre vaut 2 pour que deux sphères voisines soient tangentes) (cf. section (2.5.1)). Enfin, pour fixer le lacet dans des trous virtuels de la chaussure, on impose à la spline de passer par 8 points glissants. Sachant que si une variable libre sort du domaine de validité de la spline, on supprime simplement la contrainte du système. Ce qui correspond au fait que le lacet est sorti du trou. Ainsi, le système matériel complet possède 84 degrés de liberté, 8 variables libres et 24 contraintes atomiques.

Le temps de calcul de cette simulation est de 10.47 ms par itération en début de simulation (avec les 8 contraintes de point glissant), 6.1 ms lorsqu'il reste 5 contraintes de point glissant et de 4 ms lorsqu'il reste 3 contraintes de point glissant. L'utilisateur peut interagir avec le modèle simulé en manipulant une sonde qui entre en collision avec le modèle et qui permet de le saisir.

La figure (3.18) présente deux états de la simulation : le premier est celui au démarrage de la simulation (quand les 8 points glissants sont encore valides) et le second est un état où 4 points glissants ont

[19] http ://www.sensable.com/



(a) Etat de départ (8 points glissants) (b) Etat en cours de simulation (4 points glissants)

FIG. 3.18 – Simulation d'un lacet de chaussure

disparu à cause de la gravité du lacet et des manipulations de l'utilisateur. Une extrémité du lacet est hors champ simplement parce qu'elle est manipulée par l'utilisateur.

Un troisième test permet de mettre en évidence l'utilisation d'une contrainte glissante liée sous forme d'un nœud coulant. Pour cela, on prend une B-spline uniforme cubique de 20 points de contrôle, de masse 5 g et de coefficient d'amortissement 0.005. La spline est initialisée alignée sur l'axe des x, de longueur 35.7 cm. Elle est munie de ressorts d'élongation de raideur 100.

On fixe alors la spline en ses deux points d'abscisse paramétrique 6 et 8. On la laisse trouver son état d'équilibre. Puis, on introduit, en cours de simulation, une contrainte glissante liée spécifiant que le point d'abscisse paramétrique 10 est un point glissant qui doit être confondu avec le point d'abscisse paramétrique 1 de la même spline. Ensuite, l'utilisateur manipule le modèle comme il le souhaite à l'aide d'une sonde.

La figure (3.19) montre deux états de la simulation et met en avant le fait que le nœud formé est coulant. En effet, une partie de la spline glisse sur une autre partie de la spline.

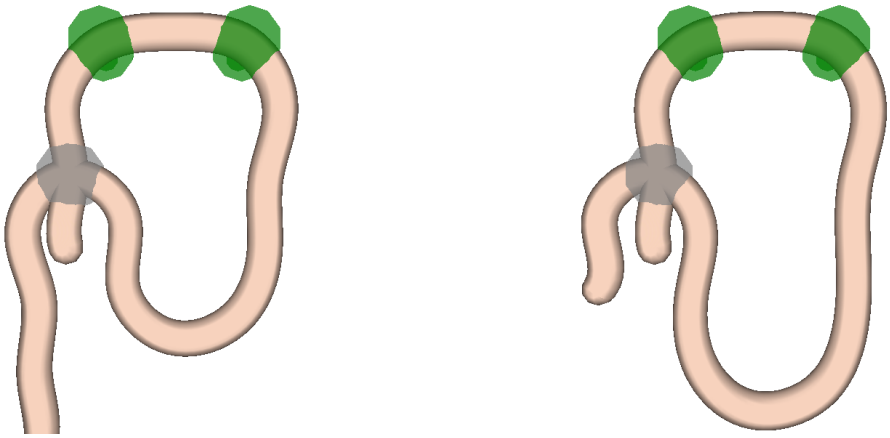


FIG. 3.19 – Simulation d'un nœud coulant

Cette simulation requiert un temps de calcul de 2.7 ms par itération.

3.5 Applications

Cette section a pour but d'exposer quelques applications possibles des propositions faites dans ce chapitre. On trouve par exemple, la découpe d'objets filiformes à des endroits pré-déterminés notamment pour la simulation chirurgicale, une technique d'interaction pour des objets articulés, la suture d'organe en simulation chirurgicale et enfin, des applications dans le domaine de la modélisation.

3.5.1 Prédécoupe [LF04]

L'une des applications les plus couramment menées en chirurgie est la découpe d'organes (lors d'une ablation) ou de fil. Malheureusement, les découpes requièrent des calculs à la volée qui ne sont pas négligeables puisqu'il faut changer la topologie du modèle. De plus, ces calculs ne sont pas prédéfinis puisque la zone de découpe n'est pas connue à l'avance. Pour éviter toute cette lourdeur, il est possible de fixer la zone de découpe pour mettre en place des précalculs. Cette technique est appelée une prédécoupe : l'objet n'est pas découpé mais les structures de données sont déjà en place pour permettre une découpe à un endroit pré-déterminé.

Les travaux de Wu et Heng[WH04] exposés en section (1.1.2.2.2) permettent de simuler une prédécoupe d'un organe en utilisant les travaux de Stéphane Cotin sur la méthode de condensation et le modèle hybride FEM quasi-statique/masses-tenseurs. La zone de découpe est connue avant la simulation, ce qui permet d'effectuer des précalculs offrant une simulation temps-réel de la pré-découpe. De plus, les auteurs proposent de résoudre le système d'équations dynamiques résultant par une méthode itérative, où chaque itération est effectuée sur le GPU [20] en codant les informations nécessaires dans des textures.

Le modèle hybride (cf. section (1.1.2.2.7)) de Cotin et al.[CDA00] est à la fois un modèle de découpe et de pré-découpe. Il autorise une découpe dynamique sur un réseau masses-tenseurs, mais il limite la zone de découpe à une partie de l'objet simulé. En effet, les découpes ne peuvent pas s'opérer sur la zone simulée en FEM quasi-statique.

Nous proposons une méthode générique pour simuler une prédécoupe d'objets filiformes (fil, intestin, trompe de fallope ou tout autre organe d'aspect filiforme) en imposant la zone de découpe. Pour cela, nous proposons de simuler une articulation de trois splines fixées ensembles dans le but de n'en paraître qu'une seule. Ainsi, la spline du milieu est contrainte en ses deux extrémités avec les deux splines voisines. Des contraintes de point fixe, tangente fixe et courbure fixe permettent d'assurer une continuité C^2 du modèle. Lors de la simulation, la plate-forme est capable de détecter une découpe du modèle par un outil de type ciseaux. Dès lors que la découpe se produit au niveau d'une des deux jonctions, on supprime les trois contraintes (position, tangente et courbure) (neuf contraintes atomiques) présentes à cette jonction. Ainsi, on simule une prédécoupe du modèle. En réitérant l'opération sur l'autre jonction, il est possible d'aboutir à plusieurs morceaux indépendants (ils font toujours partie du même système matériel mais plus aucune liaison ne les contraint ensemble).

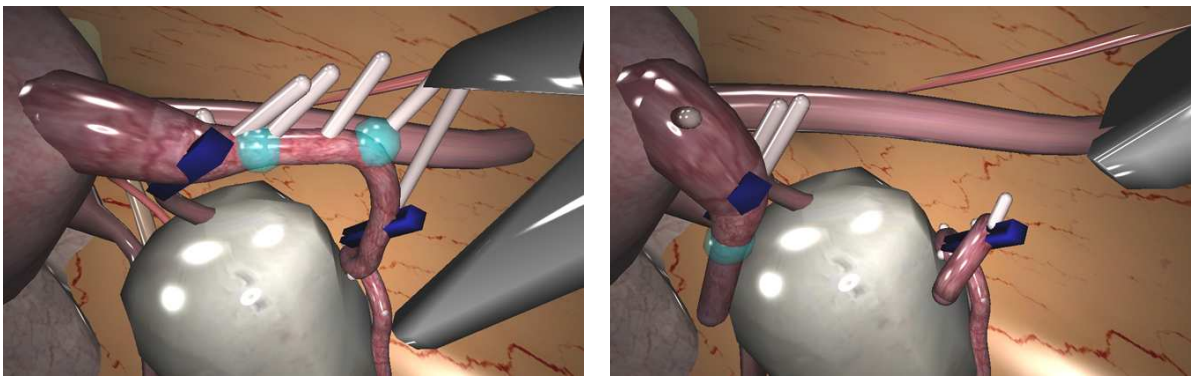
Ce procédé a été mis en place pour la simulation d'une salpingectomie[21][LF04] et permet donc de simuler une trompe de fallope à l'aide de trois splines articulées. Des contraintes supplémentaires sur les splines permettent de positionner et fixer la trompe de fallope dans la cavité abdominale.

Le système matériel possède 90 degrés de liberté (six points de contrôle pour la première spline, cinq pour la seconde et dix neuf pour la dernière) et dix huit équations de contraintes atomiques. De plus, deux splines possèdent chacune une contrainte de point fixe pour lier la trompe aux autres organes. Les trois splines sont également attachées par des ressorts fixés à la scène pour leur donner la forme d'une trompe de fallope.

Cette simulation demande un temps de calcul de 4.83 ms par itération. Sachant qu'à la fin de l'opération, les contraintes n'existent plus dans le système. Ainsi les temps de calcul chutent à 3.61 ms après suppression d'une jonction et 2.98 ms en fin d'opération. La simulation de l'opération est présentée sur la figure (3.20) où l'on peut observer l'état de la trompe de fallope avant l'opération et après la

[20] GPU : Graphics Processing Unit
[21] Salpingectomie : ablation de tout ou partie d'une trompe de fallope.

première suppression de jonction. Les sphères bleutées représentent les deux jonctions et les tiges blanches représentent les ressorts liant les points de contrôle à un point fixe de la scène. Les deux objets bleus rectangulaires, fixés sur la trompe de fallope, sont des agrafes qui ont été posées dans l'étape précédente de l'opération et qui permettent d'éviter les effusions de sang lors de la découpe.



(a) Avant découpe (b) Après la première découpe

FIG. 3.20 – Opération chirurgicale de salpingectomie

Cette application met en avant les possibilités d'une articulation à créer et supprimer des contraintes (potentiellement des liaisons entre divers objets) dynamiquement.

3.5.2 Technique d'interaction [MLFC04]

L'un des problèmes majeurs en simulation physique interactive est précisement l'interaction entre l'utilisateur et la simulation physique.

Dans ce contexte, la manière la plus intuitive de réaliser l'interaction est d'utiliser un périphérique à retour d'effort. De cette manière, l'utilisateur ressent les collisions et ne peut pas, ou presque pas, les violer. Cependant cette solution pose des problèmes numériques puisqu'il faut adapter la fréquence de simulation à la fréquence du périphérique à retour d'effort. L'autre problème de cette solution est le prix souvent élevé de ce type de périphérique.

Une autre solution consiste à utiliser des périphériques à positionnement relatif (comme la souris). Ainsi, les mouvements indésirables ou incompatibles avec la simulation sont simplement ignorés.

A première vue, la solution d'un périphérique absolu sans retour d'effort devrait être éliminée pour les environnements de simulation physique. Cependant, dans certaines applications spécifiques, l'utilisation de ce type de périphérique est incontournable. Par exemple, la simulation d'une procédure laparoscopique est un bon exemple. Celle-ci demande l'utilisation d'outils (forceps, pinces, etc...) dont les positions absolues sont mesurées et envoyées à la station de travail de la simulation. Dans ce contexte, on note également que les outils d'interaction ne peuvent pas être assimilés à de simples objets rigides. La pince de laparoscopie par exemple obéit à la physique d'un corps articulé alors que d'autres objets comme l'endobag exhibent des déformations.

Nous proposons donc une méthode générique pour gérer des outils d'interaction complexes dans des environnements de simulation physique.

Notre proposition repose sur le concept de *God Object* imaginé à l'origine par Dworkin et Zelter[DZ93]. Les auteurs remarquent que la discrétisation temporelle de la simulation et l'acquisition des données du périphérique provoquent une discontinuité et une décorrélation des positions, ce qui induit des mouvements quasi-aléatoires, comme contrôlés par la "main de Dieu". Ils proposent alors de considérer cette position comme le but à atteindre d'un objet virtuel dont la vitesse est limitée.

Ce concept est par la suite employé dans le contrôle du retour d'effort par Zilles et Salisbury[ZS95] pour être ensuite nommé *virtual proxy* par Ruspini et al.[RKK97]. Meyer et al. montrent la souplesse de cette technique pour la conception d'une architecture à un périphérique à retour d'effort. Tous ces travaux

s'appuient sur une approche statique du positionnement du *virtual proxy*. Les positions du périphériques sont analysées pour déterminer la meilleure position pour l'objet virtuel relativement à l'environnement. Ce calcul s'établit suivant une méthode d'optimisation. Cette méthode a été adaptée aux corps déformables par Mendoza et Laugier[ML01].

L'utilisation d'une méthode statique peut introduire une discontinuité des positions de l'objet virtuel dans le temps. Pour des objets manipulés ou des interactions plus complexes, des méthodes spécifiques doivent être mises en place pour déterminer les positions de l'objet virtuel.

D'une manière similaire au contrôle en animation physique[LCGG95], la dynamique est une approche convenable pour le contrôle de la position d'un objet en contact. McNeely et al.[MPT99] utilisent la loi du mouvement pour déterminer la position d'un objet virtuel rigide avec une géométrie complexe. Zhuang et Canny[ZC00] montrent que l'utilisation de la dynamique permet de mettre en place des modèles de collision plus précis qui sont nécessaires pour interagir avec des objets déformables. Une approche dynamique est également employée dans le cas d'objets articulés[RK98, CSC03] mais, dans ces travaux, l'interface ne contrôle que l'extrémité de l'objet articulé.

Nous proposons dans[MLFC04] de généraliser l'approche dynamique et de l'étendre pour permettre le contrôle de tout type d'objets quelle que soit sa complexité. Pour cela, nous considérons que le périphérique contrôle tout ou partie des degrés de liberté de l'outil virtuel laissant le soin à la dynamique de déterminer les autres.

Dans ce procédé, une erreur e est calculée entre les mesures et les données issues de la simulation. Des PD[22] contrôleurs permettent d'en déduire des forces pour la simulation et le retour d'effort. Ce principe est schématisé sur la figure (3.21). Les deux contrôleurs étant décorrélés, on peut déterminer des forces différentes entre l'environnement virtuel et le périphérique, ce qui offre une grande souplesse d'utilisation. De plus, un tel système peut être utilisé avec un périphérique à retour d'effort ou non sans influencer la partie propre à la simulation.

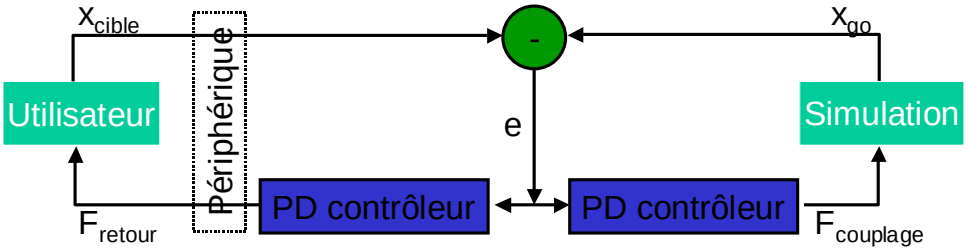


FIG. 3.21 – Schéma de différenciation entre l'objet virtuel et l'objet manipulé

Les forces calculées par les contrôleurs sont des forces proportionnelles aux déplacements et aux vitesses relatives. On peut donc les assimiler à des forces résultantes de ressorts amortis placés entre les parties de l'objet manipulé et les parties analogues de l'objet simulé. Ces liaisons sont schématisées sur la figure (3.22) pour une pince de chirurgie composée de plusieurs objets rigides articulés.

Dans cet exemple mettant en scène un objet articulé, la pince de chirurgie est décomposée en un trocart, un tube et les deux pinces qui permettent d'effectuer les actions (comme schématisé sur la figure (3.23)). L'articulation est définie de la manière suivante :

- Le trocart est fixé dans la scène par une contrainte de point fixe. Cette contrainte est sur l'axe central du trocart, elle définit donc une contrainte de pivot.
- Le tube est lié au trocart à l'aide d'une contrainte de glissière (cf. annexe (G)). Ainsi, le tube coulisse sur l'axe central du trocart et donc leurs mouvements en rotation sont liés. Si le trocart pivote, il entraîne le tube et réciproquement.
- Les deux pinces sont fixées au tube par deux contraintes de point fixe. Mais, les pinces gardent ainsi toute leurs libertés de mouvement en rotation. Or, en réalité les pinces ne possèdent qu'un seul degré de liberté en rotation, celui qui permet d'ouvrir et fermer la pince. On impose donc

[22] Proportional Derivative Controller

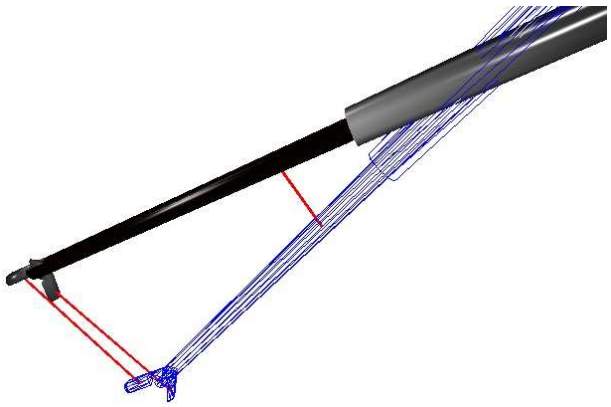


FIG. 3.22 – Schéma des ressorts reliant l'objet simulé de l'objet manipulé. Cas d'une pince de chirurgie composée d'objets articulés.

pour chacune des pinces deux contraintes supplémentaires de rotation (cf. annexe (G)) vis-à-vis du tube. Ceci ne laisse aux pinces qu'un seul degré de liberté, l'ouverture/fermeture des pinces pour saisir des objets.

Le temps de simulation de deux pinces de chirurgie (une droite et une gauche), dans une cavité abdominale figée, est de 5.63 ms.

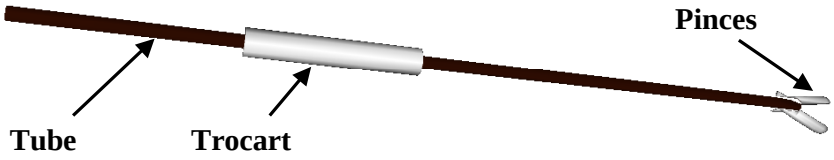


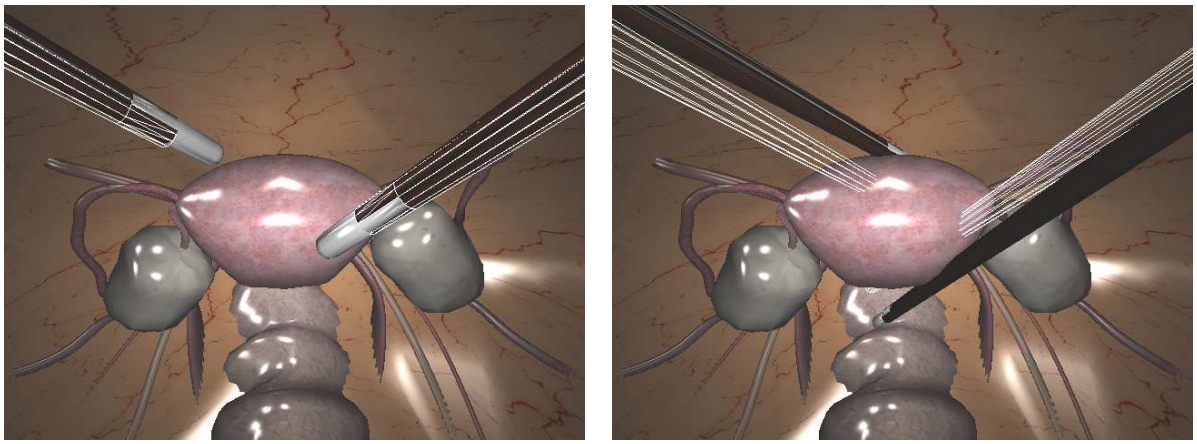
FIG. 3.23 – Décomposition d'une pince chirurgicale

Une telle simulation d'objets articulés couplés avec un périphérique absolu demande un temps de calcul de 9.7 ms pour deux pinces de chirurgie complétement articulées. Deux états de la simulation sont affichés sur la figure (3.24).

Les objets en fil de fer représentent les objets manipulés tandis que les objets pleins représentent l'objet simulé (le *GodObject*). Si l'objet manipulé ne rentre pas en interpénétration avec l'environnement, l'objet simulé n'est alors pas contraint par des collisions extérieures, il est confondu avec l'objet manipulé (schéma 3.24(a) de la figure 3.24). Cependant, il existe toujours un temps de latence, entre les deux objets, dû à l'utilisation d'un schéma de Baumgarte qui lisse la contrainte dans le temps. Si maintenant, l'objet manipulé rentre en interpénétration avec l'environnement, l'objet simulé subit les collisions et reste donc au contact de l'obstacle virtuel tout en étant attiré par l'objet manipulé à cause des ressorts qui les lient (schéma 3.24(b) de la figure 3.24). L'utilisateur peut donc manipuler à sa guise le périphérique, la simulation reste cohérente.

Ce principe est applicable sur des objets articulés[LF04] mais également sur des objets déformables. La figure (3.25) montre l'interaction d'un objet déformable avec un organe fixe dans la cavité abdominale.

La technique proposée offre une grande souplesse dans l'interaction et permet de pallier le problème visuel de l'interpénétration des objets virtuels, tout en prenant en considération la dynamique de l'objet manipulé. De plus, elle met en avant l'utilisation des contraintes inter-objets dans le cas d'objets articulés.



(a) GodObject confondu avec l'objet manipulé (b) GodObject disjoint de l'objet manipulé

FIG. 3.24 – Technique du GodObject sur les pinces de chirurgie

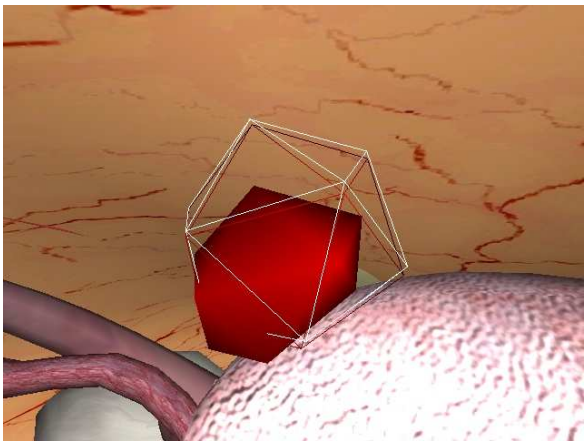


FIG. 3.25 – Application du God-Object pour un objet déformable

3.5.3 Suture [LMGC04]

Une application couramment menée en chirurgie est la suture d'organe ou de plaie. La simulation de cette opération chirurgicale demande la simulation du fil, manipulé par une aiguille et le tout en interaction avec un organe. Pour cette application, on a choisi un organe non simulé physiquement, il n'est donc qu'un objet visuel. Comme l'interaction aiguille-fil est très forte, notre première idée a été d'appliquer une contrainte de projection pour assurer la liaison de l'aiguille avec le fil. Mais une contrainte gérée par projection perturbe beaucoup trop le modèle spline dynamique. Nous avons donc choisi de simuler l'aiguille à l'aide de la technique d'interaction du *GodObject* (cf. page précédente). Ainsi, l'utilisateur manipule une aiguille invisible qui indique à l'aiguille simulée (par un objet rigide) la position qu'elle doit atteindre. On place alors l'aiguille et le fil dans un même système matériel à l'aide d'une articulation. Puis, on demande à l'articulation d'assurer une contrainte de point fixe entre l'extrémité arrière de l'aiguille et une extrémité du fil. Ainsi, la simulation reste stable et aucune projection ne vient perturber la dynamique du fil.

Ensuite, on simule la suture de l'organe en détectant la percée de l'aiguille sur la surface de l'organe. Dès lors que l'organe est percé, on pose dans le système dynamique deux contraintes supplémentaires :

- Une contrainte de point glissant, démarrant à l'extrémité du fil contraint sur l'aiguille, sur le point d'insertion de l'organe.

- En réalité, l'organe maintient localement le fil au point d'insertion et lui donne une direction en ce point quasiment constante. C'est pourquoi, on définit également une contrainte de tangente fixe glissante sur la même variable libre que la contrainte précédente. La tangente de référence étant la normale à la surface de l'organe à l'endroit de l'insertion.

Pour une spline de 20 points de contrôle (60 degrés de liberté) et deux points d'insertion (2 variables libres et 12 contraintes atomiques), la simulation demande un temps de calcul de 30 ms sur un Pentium IV 1.7 GHz. La figure (3.26) montre deux états d'une simulation de suture d'organe qui se déroule dans la cavité abdominale. Il est tout à fait envisageable d'ajouter à l'articulation (aiguille-fil) un organe

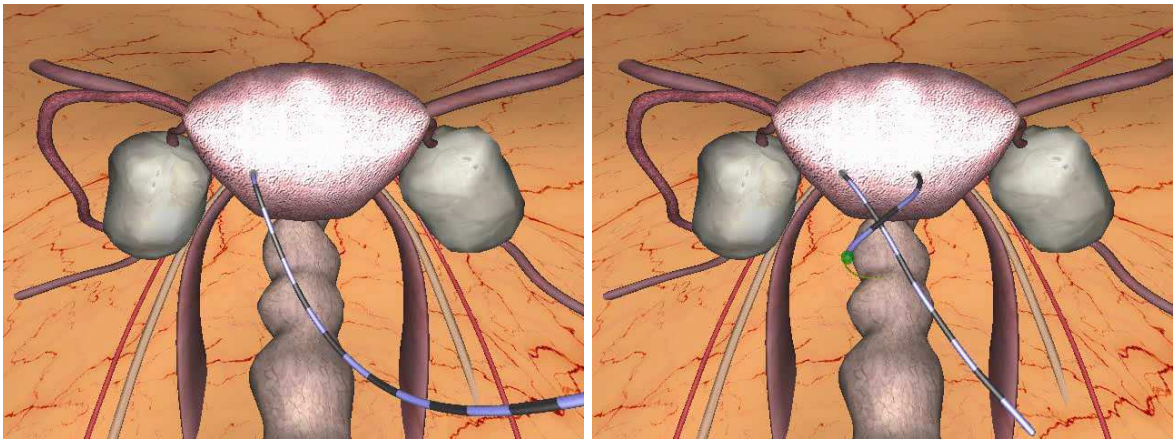


FIG. 3.26 – Suture d'un organe dans la cavité abdominale

déformable simulé physiquement pour intégrer dans la simulation dynamique l'interaction du fil avec l'organe. Pour cela, il peut être utile d'injecter un frottement local aux contraintes de point glissant. L'interaction fil-organe est alors définie par la force permettant de vérifier une contrainte de point glissant. Or cette force est donnée par les multiplicateurs de Lagrange de cette contrainte. Ainsi, on peut utiliser les multiplicateurs de Lagrange pour introduire une force extérieure dans le modèle déformable au point de surface contraint. Aucun exemple de contrainte glissante dans une articulation ne peut être actuellement montré parce que ce type de contraintes n'est pas encore géré par les articulations de notre moteur.

Pour le moment, l'interaction fil-organe est simulée en pseudo-statique. Dans la contrainte de point glissant, les données relatives à l'organe sont donc considérées comme constantes à chaque itération de la simulation. Ainsi, la contrainte de point glissant s'effectue sur un point de référence de l'organe considéré comme constant sachant qu'il peut varier d'une itération à l'autre.

Une simulation de suture de tissu a également été mise en place pour mettre en évidence l'interaction du fil avec des corps déformables. Un état de la simulation est présenté sur la figure (3.27).

Dans cette simulation, les deux tissus sont des masses-ressorts reliés à leurs extrémités par des ressorts fixés dans la scène.

3.5.4 Modélisation [LGM+04]

La modélisation variationnelle est un domaine traitant des méthodes et techniques utiles à la modélisation sous contraintes. Dans ce contexte, les contraintes aident le concepteur dans sa tâche de modélisation en fixant des invariants sur le modèle qu'il doit manipuler. Les contraintes les plus utilisées sont celles qui affectent un point particulier de l'objet (une contrainte de point fixe, tangente, normale...). Les techniques déployées s'efforcent de faire respecter des contraintes géométriques en minimisant une fonction d'énergie, propre à l'objet, sous contrainte. Pour cela, l'approche utilisée pour résoudre les contraintes peut être statique[WFB87] (minimisation d'énergie) ou dynamique[QT96] (utilisation des équations de Lagrange pour simuler un objet de type NURBS).

Welch et Witkin[WW92] proposent des contraintes de type géométrique de dimension finie (contrainte

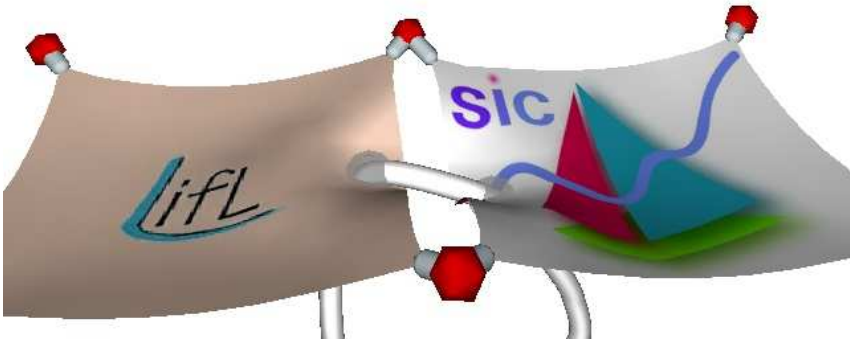


FIG. 3.27 – Suture de deux tissus

relative à un point particulier du modèle) ou transfinie (contrainte définie par une intégrale sur le modèle continu). Une contrainte de dimension finie est relative à un point particulier du modèle alors qu'en dimension transfinie, la contrainte prend en considération l'ensemble des points du modèle (formulation intégrale de l'équation de contrainte). Mais aucune des contraintes proposées ne permet de gérer une contrainte glissante telle que nous l'avons définie plus haut.

Witkin et al.[WFB87] proposent un ensemble de contraintes, sous forme énergétique, pour des modèles paramétriques. Pour cela, ils supposent qu'un modèle est capable de se définir sous trois formes particulières :

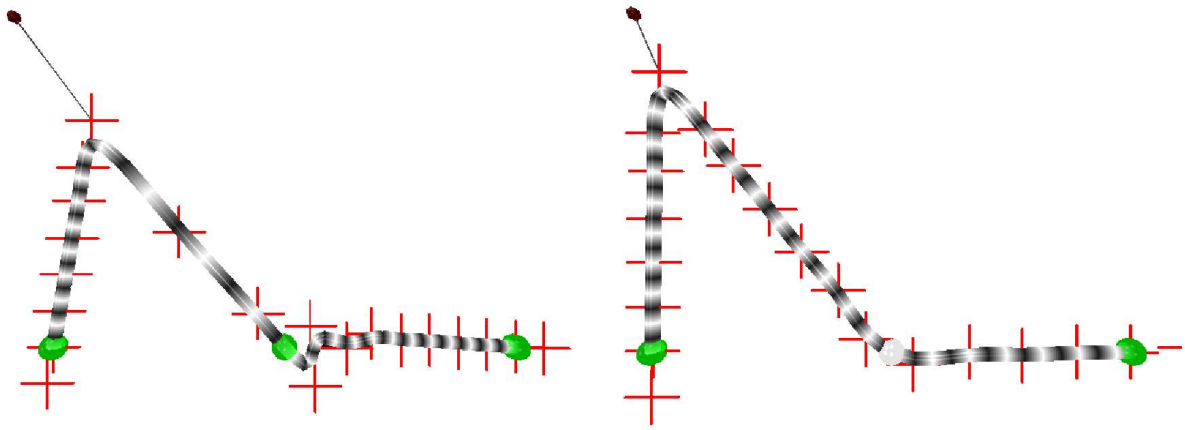
- une représentation paramétrique de la surface $\mathbf{P}(u, v)$ avec u et v les abscisses paramétriques.
- une représentation paramétrique des normales à la surface $\mathbf{N}(u, v)$ avec u et v les abscisses paramétriques.
- une représentation implicite de la surface $I(\mathbf{X})$ où $\mathbf{X}$ un vecteur de l'espace, la fonction retourne un potentiel (donc un scalaire). La surface est alors définie à l'aide d'une isovaleur e par $\{X|I(\mathbf{X}) = e\}$.

L'une des contraintes proposées par les auteurs est la contrainte d'attachement flottant (*Floating attachment*), elle permet de lier un point $\mathbf{P}(u, v)$ d'un objet à la surface $I(\mathbf{X})$ d'un autre objet. Ainsi, on a bien une contrainte liée à un point mobile d'un objet, donc une contrainte glissante. Cependant, la mise en place de la contrainte suppose l'existence d'une version implicite de l'objet, ce qui peut s'avérer inefficace pour la plupart des modèles déformables ou même lourd à définir pour des modèles 1D.

Nous proposons dans [LGM+04] d'utiliser la classe des contraintes glissantes dans le contexte de la modélisation variationelle. L'approche reprend les travaux de Qin et Terzopoulos[QT96] qui proposent une simulation dynamique d'une spline par les équations de Lagrange. Notre proposition consiste donc à étendre ce travail pour prendre en considération les contraintes glissantes. Ainsi, on offre au concepteur la possibilité de demander à une spline de passer par un point fixe de l'espace ou par un point lié à un autre objet. Un autre intérêt des contraintes glissantes en modélisation est l'homogénéisation du modèle paramétrique, autrement dit, que la tension du modèle soit uniformément répartie. Pour cela, l'utilisation d'une contrainte de point glissant au lieu d'une contrainte de point fixe permet d'automatiser cette tâche.

Le schéma (a) de la figure (3.28) montre une spline fixée à ses extrémités et également en son point milieu. Si l'utilisateur manipule le modèle, le point fixe empêche le modèle de répartir les déformations et donc des tensions apparaissent au niveau du point contraint. Ainsi, en ce point, le modèle spline possède

une tangente locale démesurée. Sur la figure, la zone située entre le point de contrôle manipulé et le point fixe possède une paramétrisation faible, ce qui supprime tout contrôle local. Alors que sur les autres zones du modèle, la paramétrisation n'a pas changée. Cette différence de paramétrisation induit au niveau du point fixe une déformation du modèle sur la partie libre (à droite du point fixe).



(a) Point fixe (b) Point glissant

FIG. 3.28 – Mise en évidence de la re-paramétrisation automatique (les croix rouges représentent les points de contrôle).

Si maintenant la contrainte de point fixe placée au milieu du modèle est remplacée par une contrainte de point glissant (cf. schéma (b) de la figure (3.28)), le modèle peut alors glisser sur ce point pour distribuer équitablement les énergies de déformation. De ce fait, le modèle corrige sa paramétrisation automatiquement de manière à diffuser les tensions du modèle. On remarque donc sur la figure l'uniformité de la répartition spatiale des segments paramétriques.

Tous les tests présentés en section (3.4.3) sont également valables dans le contexte de la modélisation variationnelle, notamment la manipulation d'un lacet qui peut être une tâche fastidieuse dans un outil de CAO. On peut également imaginer utiliser des contraintes glissantes liées pour modéliser un nœud de pendu. Il peut être utile dans le cadre de la modélisation d'annuler l'effet de la pesanteur en définissant un vecteur gravité nul ou en supprimant son apport dans les équations. Ainsi, dès que le modèle trouve une position non déformée, il reste figé jusqu'à la prochaine intervention de l'utilisateur. Ceci permet donc de donner plus de contrôle au concepteur dans la manipulation du modèle.

Enfin, l'édition directe des contraintes glissantes offre des possibilités supplémentaires au concepteur. Par exemple, une contrainte de point glissant peut être manipulée à loisir pour trouver la position idéale dans la scène par laquelle la spline doit passer. La figure (3.29) montre trois états différents d'une scène où une contrainte de point glissant est manipulée au clavier.

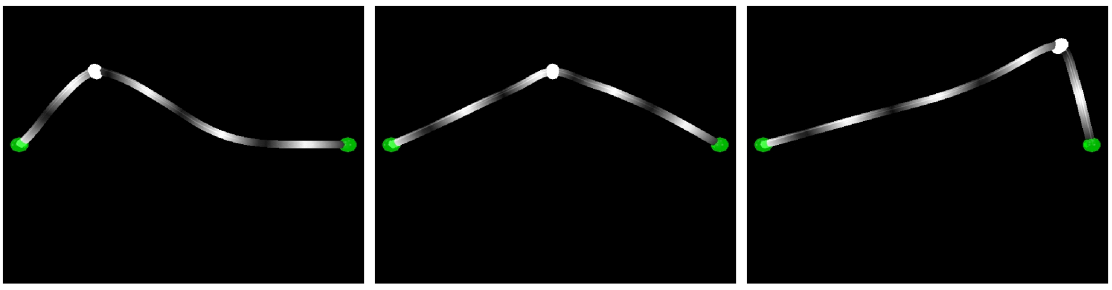


FIG. 3.29 – Edition directe d'une contrainte de point glissant

L'utilisation des contraintes glissantes en modélisation variationelle est donc enrichissante puisque ces contraintes apportent une homogénéisation automatique du modèle paramétrique. Cette tâche manuelle étant difficile à réaliser et prenant un temps considérable. Ces contraintes apportent également le fait d'imposer à un modèle de passer par un point connu de l'espace ou d'un autre objet. De plus, l'édition directe de ces contraintes permet de modifier à volonté la position par laquelle la courbe doit passer.

3.6 Extension aux splines dynamiques 2D et 3D

L'ensemble des travaux présentés dans ce chapitre portemajoritairement sur le modèle de spline dynamique 1D. Or, ce modèle peut être étendu aux dimensions supérieures (cf. section (2.6))[RNN00], il est donc naturel d'étudier l'ensemble des contraintes vues dans ce chapitre pour les splines dynamiques de dimensions 2 et 3. Pour cela, nous reprenons l'ensemble des notations employées auparavant.

Nous proposons donc de reprendre le plan de ce chapitre en le détaillant pour les splines 2D et 3D.

3.6.1 Contraintes internes

La méthode des multiplicateurs de Lagrange est également employée dans le cas de spline 2D ou 3D pour fixer des contraintes sur la surface spline ou le volume spline. Ainsi, le système d'équations dynamiques $\mathcal{M}.\mathbf{A} = \mathbf{B}$ des splines 2D/3D (déterminé en sections (2.6.1) et (2.6.2)) est étendu par :

$$\begin{pmatrix} \mathcal{M} & L^T \\ L & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ -\boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{E} \end{pmatrix} \tag{3.29}$$

Ainsi, une contrainte atomique est liée à un multiplicateur de Lagrange, une ligne de la matrice L et une entrée dans le vecteur $\mathbf{B}$. La matrice L définit la relation entre la contrainte et les degrés de liberté, le vecteur $\mathbf{B}$ mesure l'erreur courante commise sur la contrainte et le multiplicateur de Lagrange informe de l'intensité de l'effort à produire pour satisfaire la contrainte.

Il est à noter que la matrice des masses généralisées possède une structure particulière dans les cas 2D et 3D, puisque chaque ligne de cette matrice possède autant d'éléments que de points de contrôle. Il existe donc une transformation (cf équations (2.29) et (2.34)) qui permet de déterminer le degré de liberté (point de contrôle ainsi que l'axe) qui correspond à l'indice de la colonne. Ainsi, la matrice des contraintes L possède la même structure que la matrice des masses généralisées des splines 2D/3D. Autrement dit, pour chaque ligne de contrainte atomique, un élément de la ligne traite un couple ou triplet d'indices définissant un point du réseau 2D/3D de points de contrôle. Il faut donc renseigner la matrice des contraintes avec prudence en utilisant la transformation inverse (bijection).

En 2D par exemple, la colonne d'indice i de l'axe des x de la matrice L est relative à l'axe x du point de contrôle q_{i_1,i_2} avec :

$$i_1 = (i-1)/n_2 + 1 \quad \texttt{et} \quad i_2 = (i-1)\%n_2 + 1$$

la transformation inverse est donc déterminée par la relation qu'à tout couple d'indice (i_1, i_2), on fait correspondre l'indice unique :

$$i = i_1 n_2 + i_2$$

En 3D, la transformation inverse est donnée par

$$i = i_1 n_2 + i_2 + i_3 n_1 n_2$$

Nous détaillons à présent les exemples de contraintes étudiés en section (3.1) pour les dimensions suppérieures.

3.6.1.1 Contrainte de point fixe

La contrainte de point fixe désire simplement qu'un point du modèle (défini par ses abscisses paramétriques) soit confondu avec un point fixe $\mathbf{A}$ de l'espace 3D.

En 2D, cela s'exprime par :

$$\mathbf{C}(\mathbf{q}, s_1, s_2, t) = \mathbf{P}(s_1, s_2, t) - \mathbf{A}$$

Comme cette contrainte est de type holonome, son intégration dans les équations dynamiques passe par le schéma de Baumgarte (cf. équation (3.2)). Le développement de ce schéma sur cette contrainte de point fixe aboutit à l'équation finale :

$$\begin{aligned}\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\ddot{\mathbf{q}}_{\mathbf{i_1},\mathbf{i_2}}(t)b_{i_1}(s_1)b_{i_2}(s_2) &= -\frac{2}{\delta t}\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\dot{\mathbf{q}}_{\mathbf{i_1},\mathbf{i_2}}(t)b_{i_1}(s_1)b_{i_2}(s_2)\\ &- \frac{1}{\delta t^2}\left(\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\mathbf{q}_{\mathbf{i_1},\mathbf{i_2}}(t)b_{i_1}(s_1)b_{i_2}(s_2) - A\right)\end{aligned}$$

Cette équation de contrainte étant vectorielle, elle s'insére sur trois lignes de la matrice L, trois entrées dans le vecteur $\mathbf{E}$ et introduit trois multiplicateurs de Lagrange. La partie droite de cette équation permet de renseigner les entrées du vecteur $\mathbf{E}$ correspondant à la contrainte. Ces valeurs déterminent l'intensité de la violation de la contrainte qui permet de calculer les multiplicateurs de Lagrange associés à la contrainte. Quant à la partie gauche de l'équation, elle permet de remplir les coefficients des trois lignes de L correspondant à la contrainte. Les coefficients sont les mêmes sur les trois axes à savoir, des produits de fonctions de base. Donc, la matrice est constante au cours du temps.

En 3D, le principe est le même avec un point défini à l'aide de trois abscisses paramétriques :

$$\begin{aligned}\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\sum_{i_3=1}^{n_3}\ddot{\mathbf{q}}_{\mathbf{i_1},\mathbf{i_2},\mathbf{i_3}}(t)b_{i_1}(s_1)b_{i_2}(s_2)b_{i_3}(s_3) &= -\frac{2}{\delta t}\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\sum_{i_3=1}^{n_3}\dot{\mathbf{q}}_{\mathbf{i_1},\mathbf{i_2},\mathbf{i_3}}(t)b_{i_1}(s_1)b_{i_2}(s_2)b_{i_3}(s_3)\\ &- \frac{1}{\delta t^2}\left(\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\sum_{i_3=1}^{n_3}\mathbf{q}_{\mathbf{i_1},\mathbf{i_2},\mathbf{i_3}}(t)b_{i_1}(s_1)b_{i_2}(s_2)b_{i_3}(s_3) - A\right)\end{aligned}$$

Cette équation s'insére dans le système de la même manière que dans le cas 2D, avec la partie droite dans le vecteur $\mathbf{E}$ et la partie gauche dans la matrice L.

Cette contrainte permet de fixer les trois degrés de liberté d'un point quelconque d'un modèle spline 2D ou 3D. Mais, il peut être interessant de supprimer un seul ou deux degrés de liberté d'un point, de manière à lui laisser une liberté de mouvement dans une direction ou sur un plan choisi.

3.6.1.2 Contraindre un point dans un plan ou sur un axe

La contrainte la plus simple consiste à supprimer un degré de liberté pour un point du modèle. Il suffit donc d'empêcher un point de se déplacer dans une direction, il est donc contraint à rester dans le plan orthogonal à cette direction. Soit $\mathbf{n}$ le vecteur normal à ce plan et $\mathbf{A}$ un point du plan. Le plan est donc défini par l'ensemble des points $\mathbf{P}$ vérifiant l'équation :

$$\mathbf{AP}.\mathbf{n} = 0$$

Pour contraindre le point $\mathbf{P}(s_1, s_2, t)$ d'une spline 2D à rester dans le plan, il suffit que ce point vérifie l'équation du plan. Ainsi, l'équation de contrainte s'écrit :

$$C(\mathbf{q}, s_1, s_2, t) = \mathbf{n}.\mathbf{AP}(s_1, s_2, t)$$

Comme cette contrainte est également de type holonome, on utilise le schéma de Baumgarte (cf. équation 3.2) pour former l'équation de contrainte qui est insérée dans le système dynamique :

$$\begin{aligned}\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}b_{i_1}(s_1)b_{i_2}(s_2)\ddot{\mathbf{q}}_{\mathbf{i_1},\mathbf{i_2}}(t).\mathbf{n} &= -\frac{2}{\delta t}\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}b_{i_1}(s_1)b_{i_2}(s_2)\dot{\mathbf{q}}_{\mathbf{i_1},\mathbf{i_2}}(t).\mathbf{n}\\ &- \frac{1}{\delta t^2}\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}b_{i_1}(s_1)b_{i_2}(s_2)\mathbf{q}_{\mathbf{i_1},\mathbf{i_2}}(t).\mathbf{n}\end{aligned}$$

Dans le cas d'une spline 3D, le processus est le même mais avec une abscisse paramétrique supplémentaire :

$$\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\sum_{i_3=1}^{n_3} b_{i_1}(s_1)b_{i_2}(s_2)b_{i_3}(s_3)\ddot{\mathbf{q}}_{\mathbf{i_1},\mathbf{i_2},\mathbf{i_3}}(t).\mathbf{n} = -\frac{2}{\delta t}\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\sum_{i_3=1}^{n_3} b_{i_1}(s_1)b_{i_2}(s_2)b_{i_3}(s_3)\dot{\mathbf{q}}_{\mathbf{i_1},\mathbf{i_2},\mathbf{i_3}}(t).\mathbf{n} - \frac{1}{\delta t^2}\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\sum_{i_3=1}^{n_3} b_{i_1}(s_1)b_{i_2}(s_2)b_{i_3}(s_3)\mathbf{q}_{\mathbf{i_1},\mathbf{i_2},\mathbf{i_3}}(t).\mathbf{n}$$

En combinant deux contraintes de ce type sur un même point du modèle, on contraint ce point à être à l'intersection de deux plans, donc sur une droite. On emploie donc cette contrainte en double pour fixer deux degrés de liberté d'un point et ainsi le contraindre à évoluer le long d'une droite choisie.

3.6.2 Contraintes externes

La proposition faite en section (3.2) pour gérer les contraintes entre corps s'appuie sur une interface commune aux objets articulés. Si un modèle de spline 2D ou 3D souhaite être articulé avec d'autres objets, il suffit qu'il hérite de cette interface et qu'il implémente les méthodes qui l'intéressent. Par exemple, la méthode du point fixe, du point contraint sur un axe ou un plan...

3.6.3 Contraintes glissantes

La proposition de contrainte glissante de la section (3.3) n'est pas spécifique au modèle spline 1D. Elle est donc tout à fait utilisable dans les dimensions supérieures. Le système dynamique d'une spline 2D/3D est donc étendu à l'aide d'un vecteur de variables libres **s** soit par la méthode qui considère une contrainte parfaite ne produisant aucun travail (cf. section (3.3.1.1)), dans ce cas on obtient le système suivant :

$$\begin{pmatrix} \mathcal{M} & 0 & L^T & L_g^T \\ 0 & 0 & 0 & L_{gs}^T \\ L & 0 & 0 & 0 \\ L_g & L_{gs} & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ \ddot{\mathbf{s}} \\ -\boldsymbol{\lambda} \\ -\boldsymbol{\lambda}_\mathbf{g} \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{0} \\ \mathbf{E} \\ \mathbf{E_g} \end{pmatrix} \tag{3.30}$$

soit par la méthode qui considère une contrainte produisant un travail (cf. section (3.3.1.2)) appuyant la contrainte ou, au contraire, luttant contre elle. Dans ce cas, on obtient le système suivant :

$$\begin{pmatrix} \mathcal{M} & 0 & L^T & L_g^T \\ 0 & 0 & \epsilon & L_{gs}^T \\ L & 0 & 0 & 0 \\ L_g & L_{gs} & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ \ddot{\mathbf{s}} \\ -\boldsymbol{\lambda} \\ -\boldsymbol{\lambda}_\mathbf{g} \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{0} \\ \mathbf{E} \\ \mathbf{E_g} \end{pmatrix} \tag{3.31}$$

cette seconde technique permet de résoudre le système plus rapidement (cf. section (3.3.1.2)).

Quelle que soit la méthode choisie, l'intégration des contraintes glissantes dans le système (équation (3.30) ou (3.31)) s'effectue de la même manière. Prenons l'exemple d'une contrainte de point glissant, sachant que les autres contraintes étudiées en section (3.3.2) s'étendent aux cas 2D et 3D en suivant le même raisonnement.

Soit la contrainte de point glissant qui demande à un point de la spline, d'abscisses paramétriques mobiles $(s_{1k}(t), s_{2k}(t))$, d'être à la position spatiale définie par le point **A** :

$$\mathbf{C}(\mathbf{q}(t), s_{1k}(t), s_{2k}(t), t) = \mathbf{P}(s_{1k}(t), s_{2k}(t), t) - \mathbf{A}$$

cette contrainte ne dépend que des degrés de liberté et en aucun cas de leur vitesse, elle est donc de type holonome. Le schéma de Baumgarte (équation 3.2) permet donc d'établir l'équation de contrainte qui

doit être insérée dans le système :

$$\begin{aligned}
&\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2} \ddot{\mathbf{q}}_{\mathbf{i_1},\mathbf{i_2}}(t) b_{i_1}(s_{1k}(t)) b_{i_2}(s_{2k}(t)) + \mathbf{q}_{\mathbf{i_1},\mathbf{i_2}}(t)\left[b'_{i_1}(s_{1k}(t)).\ddot{s_{1k}}(t).b_{i_2}(s_{2k}(t)) + b_{i_1}(s_{1k}(t)).b'_{i_2}(s_{2k}(t)).\ddot{s_{2k}}(t)\right] \\
&= -\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2} \left[2\dot{\mathbf{q}}_{\mathbf{i_1},\mathbf{i_2}}(t)\left(b'_{i_1}(s_{1k}(t)).\dot{s_{1k}}(t).b_{i_2}(s_{2k}(t)) + b_{i_1}(s_{1k}(t)).b'_{i_2}(s_{2k}(t)).\dot{s_{2k}}(t)\right)\right. \\
&\quad \left. + \mathbf{q}_{\mathbf{i_1},\mathbf{i_2}}(t)\left(b''_{i_1}(s_{1k}(t)).\dot{s_{1k}}^2(t).b_{i_2}(s_{2k}(t)) + b_{i_1}(s_{1k}(t)).b''_{i_2}(s_{2k}(t)).\dot{s_{2k}}^2(t)\right)\right] \\
&\quad + 2\mathbf{q}_{\mathbf{i_1},\mathbf{i_2}}(t) b'_{i_1}(s_{1k}(t)).\dot{s_{1k}}(t).b'_{i_2}(s_{2k}(t)).\dot{s_{2k}}(t)\Big] \\
&- \frac{2}{\delta t}\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2} \left[\dot{\mathbf{q}}_{\mathbf{i_1},\mathbf{i_2}}(t) b_{i_1}(s_{1k}(t)) b_{i_2}(s_{2k}(t))\right. \\
&\quad \left. + \mathbf{q}_{\mathbf{i_1},\mathbf{i_2}}(t)\left(b'_{i_1}(s_{1k}(t)).\dot{s_{1k}}(t).b_{i_2}(s_{2k}(t)) + b_{i_1}(s_{1k}(t)).b'_{i_2}(s_{2k}(t)).\dot{s_{2k}}(t)\right)\right] \\
&- \frac{1}{\delta t^2}\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2} \left[\mathbf{q}_{\mathbf{i_1},\mathbf{i_2}}(t) b_{i_1}(s_{1k}(t)) b_{i_2}(s_{2k}(t)) - \mathbf{A}\right]
\end{aligned}$$

Ainsi, la partie gauche de cette équation permet de renseigner les trois lignes des matrices L_g (avec les coefficients des accélérations des degrés de liberté) et L_{gs} (avec les coefficients des accélérations des variables libres). La partie droite de cette équation permet de compléter les trois entrées du vecteur $\mathbf{E_g}$ relatives à la contrainte de point glissant.

Le cas du point glissant pour une spline de dimension trois s'écrit de la même manière. Cependant, contraindre un volume spline à avoir un point à une position spatiale n'est pas très utile si l'on considère qu'une fois le domaine de validité passé la contrainte n'existe plus. En effet, tant que le point contraint appartient au volume, le modèle se déplace sans aucune contrainte puisqu'il est défini dans toutes les directions, la contrainte glisse donc en toute liberté n'imposant aucune direction particulière. Et, dès qu'il arrive au bord, la contrainte est supprimée. Cette contrainte devient plus intéressante si on lui demande de respecter le domaine de validité des abscisses paramétriques. Là, on assure à tout moment qu'un point du volume spline vérifie la contrainte, quitte à appliquer une projection après intégration pour vérifier le domaine de validité.

On peut remarquer que les contraintes glissantes en dimensions deux et trois ne portent pas forcément sur tous les abscisses paramétriques. On peut donc définir une contrainte glissante sur une seule dimension d'une surface spline ou sur une ou deux dimensions d'un volume spline. Par exemple contraindre une surface spline à avoir un point, dont l'une des deux abscisses paramétriques est fixe, à être au point $\mathbf{A}$ de l'espace :

$$\mathbf{C}(\mathbf{q}(t), s_{1k}(t), t) = \mathbf{P}(s_{1k}(t), s_2, t) - \mathbf{A}$$

Comme dans le cas de la spline 1D, il peut être envisagé de définir des contraintes glissantes doubles permettant d'imposer une égalité entre deux points mobiles d'un même modèle spline 2D ou 3D. Cependant, l'utilisation d'une telle contrainte dans un volume spline ne serait pas très utile puisque les deux points mobiles convergeraient immédiatement vers le même point du volume. L'utilisation de contraintes glissantes liées est également envisageable en dimensions deux et trois.

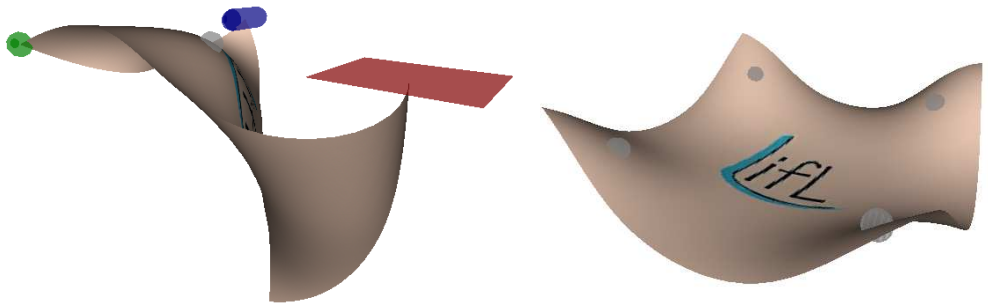
3.6.4 Tests

Dans toutes les simulations présentées dans cette section, la méthode d'intégration numérique employée est Runge Kutta 4.

Le premier test montre quelques contraintes possibles sur une spline 2D. On définit une B-spline uniforme cubique par 6×6 points de contrôle, une masse de 5 g et un coefficient d'amortissement de

0.003. Elle est initialisée dans le plan (x, z) par un carré d'arrête 38.3 cm. Les déformations sont gérées par des ressorts d'élongation et de courbure de raideur 1.

Dans un premier temps, on positionne sur le modèle 2D une contrainte de point fixe à une extrémité (d'abscisses paramétriques $(3.0, 3.0)$), une contrainte de point sur axe à une seconde extrémité (d'abscisses paramétriques $(3.0, 6.0^-)$), une contrainte de point dans un plan à une troisième extrémité (d'abscisses paramétriques $(6.0^-, 3.0)$) et enfin une contrainte de point glissant sur le point de la surface d'abscisses paramétriques $(4.0, 4.0)$.
On obtient donc un système matériel de 126 degrés de liberté, 2 variables libres (pour les 2 abscisses paramétriques du point contraint) et 9 contraintes atomiques. Le temps de calcul de cette simulation est de 9.48 ms et le schéma ((a)) de la figure (3.30) montre l'état d'équilibre du système.



(a) Quatre contraintes différentes (b) Quatre points glissants

FIG. 3.30 – Simulation d'une spline 2D contrainte

Dans un second temps, on reprend la même spline 2D de base en lui imposant 4 contraintes de points glissants initialisées aux points d'abscisses paramétriques $(3.5, 3.5)$, $(5.5, 3.5)$, $(3.5, 5.5)$ e $(5.5, 5.5)$. On obtient donc un système matériel de 126 degrés de liberté, 8 variables libres (4 contraintes glissantes à 2 variables libres chacune) et 12 contraintes atomiques. Le temps de calcul de cette simulation est de 10.9 ms et le schéma ((b)) de la figure (3.30) montre un état de la simulation qui ne trouve pas d'équilibre puisque la spline 2D glisse sur les 4 points imposés et finit par les repousser sur les bords. Là, les contraintes sont éliminées du système et donc la spline tombe sous l'effet de son poids.

Le second test sert à montrer l'application des contraintes sur le modèle spline 3D. On définit une spline de Catmull-Rom de $4\times4\times4$ points de contrôle, d'une masse de 5 kg et de coefficient d'amortissement 0.05. Les déformations sont assurées par un réseau de ressorts en élongation (raideur 10), flexion (raideur 8) et torsion (raideur 6).

Ensuite, on place sur ce modèle de spline volumique deux contraintes de point fixe (à deux extrémités opposées), une contrainte de point sur un axe pour une troisième extrémité et une contrainte de point dans un plan pour une quatrième extrémité. De plus, on définit une contrainte de point glissant dans le volume spline. Le système matériel est donc composé de 192 degrés de liberté, 3 variables libres (une contrainte de point glissant dans toutes les directions a besoin de trois variables libres) et 12 contraintes atomiques.

Le temps de calcul d'une itération de cette simulation est de 36.7 ms. La figure (3.31) montre un état de la simulation.

Le volume spline est affiché en transparence pour montrer la contrainte de point glissant qui se trouve dans le volume. Cette contrainte n'influence absolument pas la dynamique du modèle, mais elle peut être utile pour connaître, à tout moment, le point du volume qui se trouve à un endroit précis. On pourrait par exemple introduire un frottement visqueux sur ce point.

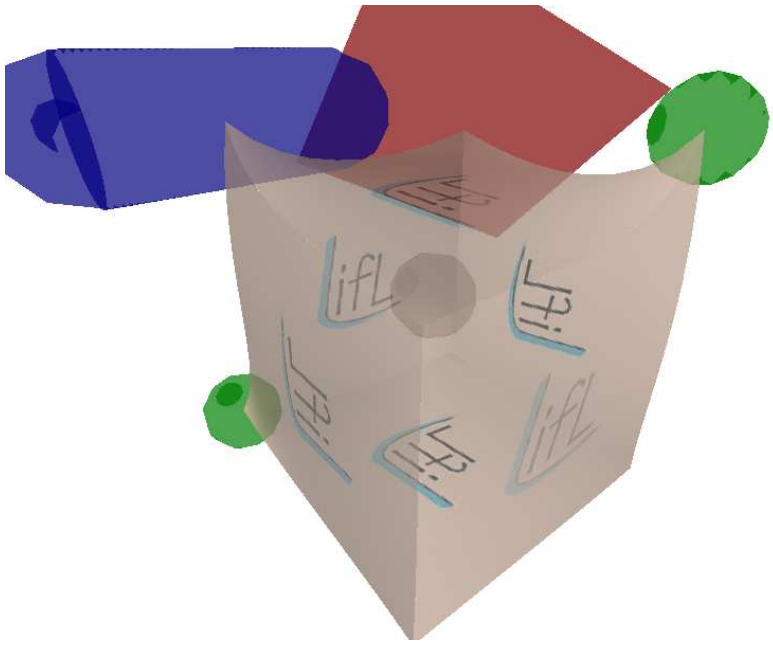


FIG. 3.31 – Simulation d'une spline 3D contrainte

3.7 Conclusion

Trois propositions ont été faites dans ce chapitre, la première consiste à introduire des contraintes dans le modèle spline dynamique. La seconde proposition concerne une architecture logicielle de gestion de contrainte entre divers objets, pour la gestion de système articulé ou lié. La troisième proposition de ce chapitre est l'extension du modèle spline dynamique à une nouvelle classe de contraintes. Ces contraintes, dites *glissantes*, permettent de définir des contraintes qui peuvent se déplacer sur le modèle suivant la dynamique d'un paramètre.

Les contributions apportées par ce travail de thèse résident dans les proposition deux et trois, avec l'architecture logicielle de simulation d'articulation et la nouvelle classe de contraintes dites *glissantes*.

Ces propositions ont fait également l'objet d'une étude pour leur utilisation avec les splines 2D et 3D. Enfin, quelques tests ont permis de démontrer l'efficacité du modèle et des propositions. De plus, plusieurs applications montrent leur souplesse d'utilisation dans divers domaines aussi variés que la simulation physique, la modélisation variationnelle, la simulation chirurgicale ou encore les techniques d'interaction.

L'extension du modèle spline dynamique aux contraintes donne un contrôle supplémentaire et une plus grande souplesse d'utilisation, ouvrant des possibilités à de nombreux domaines d'application.

Les propositions complètent le modèle spline dynamique par une large gamme de contraintes et de méthode pour lier le modèle à d'autres objets. Cependant, l'utilisation d'un schéma de Baumgarte pour gérer les contraintes ne permet pas d'assurer leur réalisation exacte. Il pourrait être intéressant d'étudier les méthodes de type post-stabilisation pour le modèle spline dynamique. Notamment, l'intégration des contraintes glissantes avec ce type de méthode. L'utilisation de ces méthodes permettrait d'alléger la taille des systèmes d'équations dynamiques à résoudre puisque qu'aucune contrainte n'est prise en compte dans la résolution. En effet, la réalisation des contraintes est reléguée après l'intégration numérique. Cette réalisation peut s'effectuer plus ou moins correctement (et inversement, plus ou moins rapidement) suivant que des approximations sont faites ou non.

Il pourrait être également intéressant d'essayer de gérer les auto-collisions du modèle par des mé-

thodes de type multiplicateurs de Lagrange (ou autre méthode plus robuste que les pénalités) pour en assurer l'intégrité. Ainsi, le nœud géré par les méthodes à pénalités de la plateforme de simulation pourrait être réalisé à l'aide de contraintes unilatérale. L'utilisation d'une telle méthode pour les auto-collisions apporterait une robustesse supplémentaire au modèle.

Pour certaines applications, le modèle spline doit pouvoir prendre des configurations géométriques complexes sur une portion limitée de la courbe. Par exemple, la suture d'organes avec terminaison de l'opération par l'établissement d'un nœud, requiert une souplesse du modèle de fil. Pour cela, il faut définir la spline avec beaucoup de points de contrôle pour lui donner assez de liberté afin qu'elle puisse prendre la configuration spatiale adéquate. Ce principe n'est pas très efficace puisque le modèle simulé possède ainsi beaucoup de degrés de liberté dont la plupart sont inutiles. En effet, seule la zone de la courbe où le nœud est serré a besoin d'une grande liberté de mouvement. Il apparaît nettement l'intérêt d'un modèle multi-résolution capable d'adapter automatiquement sa propre résolution en fonction de certains critères.

Nous proposons dans le chapitre suivant une extension du modèle spline dynamique permettant d'ajuster le placement des degrés de liberté sur le modèle spline pour lui donner automatiquement la bonne résolution vis-à-vis du mouvement désiré.

Chapitre 4

Multi-résolution

Sommaire

La multi-résolution permet d'adapter la forme ou les possibilités d'un objet pour une application particulière. Elle peut s'appliquer de différentes manières, c'est pourquoi nous proposons de commencer par étudier les divers travaux existant dans ce vaste domaine. Puis, nous exposons notre proposition de multi-résolution sur le modèle spline dynamique. Enfin, nous présentons un test d'adaptation automatique de la résolution suivi de l'application de la technique multi-résolution pour la découpe avant de conclure.

4.1 Travaux antérieurs

La multi-résolution trouve des applications dans des domaines variés comme la modélisation ou la simulation physique. Dans le premier cas, la multi-résolution est purement géométrique et permet d'adapter la forme d'un objet à une utilisation particulière. Ce changement peut être demandé par l'utilisateur (comme dans les outils de CAO) ou effectué de manière automatique. Dans le second cas, la multi-résolution s'applique au niveau mécanique et permet donc de simuler un objet à des résolutions différentes. L'intérêt majeur est de cibler le calcul dans les zones en interaction (zones actives, manipulées) ou en action soutenue (zones passives, simulées). Une multi-résolution mécanique peut provenir d'une multi-résolution du modèle géométrique sous-jacent ou proposer sa propre méthode de changement de résolution.

Afin de mieux appréhender les possibilités de la multi-résolution et les domaines couverts par ce terme, nous présentons quelques travaux qui ont trait à la multi-résolution géométrique, puis certains travaux se rapportant à une multi-résolution mécanique. Enfin, nous terminons cette section par une discussion sur les travaux présentés afin de dégager les avantages et les inconvénients de chacun. Ceci nous permet alors de mieux cerner les motivations de notre proposition.

4.1.1 Multi-résolution géométrique

Etant donné le large spectre des travaux existant, nous concentrons cette étude sur le cas particulier qui nous intéresse, celui des modèles 1D.

Il est d'abord important de clarifier le terme de multi-résolution géométrique. Dans cette section, un modèle géométrique multi-résolution est un modèle capable d'adapter sa géométrie soit en la modifiant directement, soit en la réorganisant. Les algorithmes de type niveaux de détails basé sur une multi-représentation sont donc exclus de cette définition puisqu'ils proposent plusieurs versions fondamentalement différentes d'un même modèle. Ainsi, il n'est pas possible de passer d'un niveau de détails à l'autre en modifiant quelques données ou en les réorganisant puisque les structures sont différentes. Un exemple de technique de niveaux de détails basée sur une multi-représentation des objets est celle des imposteurs qui propose de substituer un objet géométrique (une structure basée sur un ensemble de points spatiaux) par son image (un ensemble ordonné de couleurs).

Les méthodes incluses par cette définition de multi-résolution géométrique doivent donc proposer de réorganiser les données de manière à donner accès à différentes résolutions du modèle géométrique. Elles doivent sinon modifier directement les données géométriques sans établir plusieurs versions du modèle à des niveaux de détails différents.

Il existe différentes méthodes permettant de modifier la résolution d'un modèle géométrique. L'un des travaux précurseurs sur la multi-résolution est celui de Forsey et Bartels[FB88] qui introduit la notion de spline hiérarchique. Cependant, une théorie issue des mathématiques pures fait son apparition au début des années 90, c'est la théorie des ondelettes. Il existe également des algorithmes de subdivision pour des courbes (et des surfaces). Nous détaillons ici l'analyse multi-résolution par ondelettes et la méthode de subdivision.

4.1.1.1 Les ondelettes

Cette section n'a pour but que de donner un apercu très large de cet outil. Pour un descriptif plus approfondi, voir [Gri99].

L'analyse par ondelette est un outil mathématique puissant permettant d'analyser un signal quelconque (i.e. une fonction) afin de séparer les détails des données relativement grossières par le biais de filtres. Les ondelettes offrent la possibilité de stocker sur le même espace mémoire des données filtrées en plusieurs résolutions. Ainsi, on obtient une résolution extrêmement grossière associée à toutes les informations nécessaires et suffisantes pour reconstruire toutes les résolutions intermédiaires jusqu'à la résolution la plus fine, l'originale.

L'utilisation d'une décomposition en ondelette n'est soumise qu'à l'existence d'une imbrication d'espaces vectoriels V^i sur lesquels la fonction est définie :

$$... \subset V^{-1} \subset V^0 \subset V^1 \subset ... \subset V^n \subset ...$$

Cet outil est mathématiquement rigoureux, ce qui permet notamment de démontrer des propriétés intéressantes du modèle (ces propriétés sont présentées dans [Gri99]).

Une décomposition en ondelette peut être développée sur les courbes B-splines pour dégager une hiérarchie de courbes B-splines liées les unes aux autres. On obtient donc une suite de courbes B-splines de plus en plus fines. Les ondelettes offrent donc un moyen de déterminer une résolution plus fine d'une B-spline donnée. On voit donc apparaître un lien entre une décomposition en ondelette et la technique de subdivision qui est présentée dans la section suivante.

4.1.1.2 La subdivision

Soit une courbe mathématique et sa discrétisation informatique définie par un ensemble fini de points. Le principe de la subdivision est d'approcher d'aussi près que l'on souhaite la courbe mathématique en densifiant le modèle discret par un processus itératif. Pour cela, le processus raffine un maillage grossier en un maillage plus fin en adoucissant les contours et les courbures. Suivant l'algorithme itératif employé, la courbe obtenue peut interpoler les points initiaux ou les approximer.

L'algorithme de subdivision naïf est l'insertion d'un point au milieu d'un segment (point splitting) accompagnée de son opération inverse permettant de supprimer un point (edge collapse). Les techniques cité sont généralement employées sur des maillages triangulaires, donc 2D, mais ils sont transposable à la 1D. Le critère de subdivision est basé sur la longueur d'un segment. Si un segment rectiligne est trop long, un point milieu est défini. Au contraire, si un segment est de petite taille, alors l'un des deux points extrémités est supprimé, le choix peut s'effectuer en inspectant la longueur des segments voisins et en choisissant de supprimer le point commun avec le segment voisin le plus court.

D'autres algorithmes de subdivision existent, comme par exemple l'algorithme de Chaikin[Cha74] qui permet, à l'infini, d'approximer des points ordonnés par une courbe B-spline quadratique[Rie75] (cf. figure (4.1)).

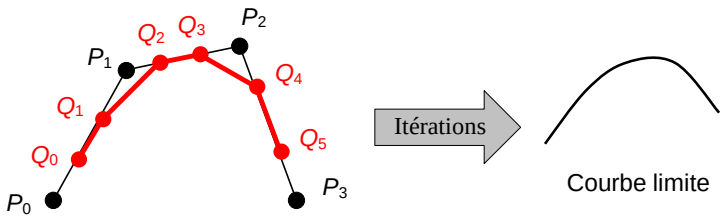


FIG. 4.1 – Subdivision de Chaikin

L'algorithme utilisé est le suivant, soit $\mathbf{P_i}$ les points du maillage grossier, les points $\mathbf{Q_i}$ du maillage plus fin sont donnés par les relations :

$$\begin{array}{lcl} \mathbf{Q_{2i}} & = & \frac{1}{4}\mathbf{P_i} + \frac{3}{4}\mathbf{P_{i+1}} \\ \mathbf{Q_{2i+1}} & = & \frac{3}{4}\mathbf{P_i} + \frac{1}{4}\mathbf{P_{i+1}} \end{array}$$

Cet algorithme fonctionne pour les maillages 1D. On trouve également des algorithmes plus spécifiques aux maillages surfaciques, comme la subdivision de Loop[Loo87, Sta98a, PX04] ou de Catmull-Clark[CC78, Sta98b]. Certains travaux proposent même des implémentations utilisant le matériel graphique[BKS00, RBAB02]. Il existe de plus des algorithmes locaux comme par exemple les *PN Triangles* qui sont définis à partir des patches de Bézier triangulaires. Enfin, on trouve des propositions de subdivisions adaptatives suivant différents critères comme des zones de l'espace, la courbure, la résolution écran, le point de vue...

La propriété de subdivision peut très souvent se présenter sous forme de masques. Un point d'une résolution est défini par une combinaison linéaire des points de la résolution plus grossière en utilisant des poids bien précis. Ces poids forment le masque de la subdivision. Ainsi, la subdivision d'une courbe peut se mettre sous forme d'un produit matrice-vecteur où le résultat est le vecteur des nouveaux points caractéristiques. L'ensemble des splines possédant cette propriété sont dites raffinables. Peter Schröder expose dans[ZSD+00] les principes d'une subdivision pour les B-splines uniformes. En partant de la formulation en convolution des B-splines, il aboutit à la propriété (1.23) :

$$N_0^k(t) = N_0^{k-1}(t) \otimes N_0^1(t)$$

et ainsi détermine la relation de raffinabilité des B-splines uniformes :

$$N_i^k(t) = \frac{1}{2^l} \sum_{j=0}^{l+1} C_j^{l+1} N_i^k(2t - j) \tag{4.1}$$

En utilisant cette équation et en posant les vecteurs

$$\mathbf{N}(t) = \begin{pmatrix} \vdots \\ N_0^k(t+1) \\ N_0^k(t) \\ N_0^k(t-1) \\ \vdots \end{pmatrix} \quad \text{et} \quad \mathbf{N}(2t) = \begin{pmatrix} \vdots \\ N_0^k(2t+1) \\ N_0^k(2t) \\ N_0^k(2t-1) \\ \vdots \end{pmatrix}$$

Schröder définit la relation matricielle :

$$\mathbf{N}(t) = \mathbf{N}(2t)S$$

où S est une matrice dont les coefficients sont donnés par l'équation (4.1).

Un point de la courbe spline est alors défini par :

$$\mathbf{P}(t) = \mathbf{N}(t)\mathbf{p^0} = \mathbf{N}(2t)\mathbf{p^1} = \mathbf{N}(2t)S\mathbf{p^0} = ... = \mathbf{N}(2^j t)\mathbf{p^j} = \mathbf{N}(2^j t)S^j\mathbf{p^0}$$

d'où l'on peut tirer la relation $\mathbf{p^1} = S\mathbf{p^0}$. Ainsi, la matrice S permet de calculer les points de contrôle d'une résolution par rapport à la résolution inférieure. Et les puissances de cette matrice permettent de déterminer les points de contrôle des résolutions suivantes.

Par exemple, la matrice S pour une B-spline uniforme cubique est constituée alternativement des coefficients $(\frac{1}{2}, \frac{1}{2})$ (pour les points d'indices impairs) et $(\frac{1}{8}, \frac{6}{8}, \frac{1}{8})$ (pour les points d'indice pair).

Cet algorithme de subdivision permet de définir une courbe B-spline uniforme à des résolutions différentes. Cependant, le fait d'avoir un vecteur de nœuds uniforme ne permet pas de changer la résolution localement et impose de recalculer l'ensemble des points de contrôle pour générer la nouvelle résolution. La résolution obtenue n'est donc pas locale mais bien globale. Pour obtenir une subdivision locale, il faut considérer une B-spline dont le vecteur de nœuds est non-uniforme. Ainsi, il est possible d'insérer un nouveau nœud à n'importe quel endroit du vecteur de nœud associé à un nouveau point de contrôle. De cette manière, un seul point de contrôle est ajouté produisant une nouvelle résolution locale de la courbe.

Soit une B-spline non uniforme d'ordre $d+1$ de vecteur de nœuds $\mathbf{U} = (u_0, ..., u_n)$ et de fonctions de base $N_{j,d}$. On considère que cette spline est également définie par un vecteur de nœuds $\mathbf{W} = (w_0, ..., w_{n+k})$ créé à partir de $\mathbf{U}$ en insérant k nœuds associés aux fonctions de base $\tilde{N}_{j,d}$. Donc, on suppose qu'il existe une injection σ qui envoie $(0, ..., n)$ sur $(0, ..., n+k)$ avec :

$$\forall i \in \{0, ..., n\}, \begin{cases} i \leq \sigma(i) \\ u_i = w_{\sigma(i)} \end{cases}$$

L'insertion est possible s'il existe des coefficients $\beta_{i,j}^d$ tels que :

$$\forall j \in \{0, ..., n-d\}, \forall s, N_{j,d}(s) = \sum_{i=0}^{n+k-d} \beta_{i,j}^d \tilde{N}_{i,d}(s)$$

Il existe par exemple l'algorithme d'*Oslo* (ou encore, l'algorithme de Boehm[Boe80]) qui permet de calculer ces coefficients à l'aide de la formule récursive suivante :

$$\begin{aligned} \beta_{i,j}^0 &= \begin{cases} 1 & \text{si } t_i^* \leq t_j \leq t_{i+1}^* \\ 0 & \text{sinon} \end{cases} \\ \forall k > 1 \quad \beta_{i,j}^k &= \frac{t_{j+k} - t_i^*}{t_{i+k-1}^* - t_i^*} \beta_{i,j}^{k-1} + \frac{t_{i+k+1}^* - t_{i+k}}{t_{i+k+1}^* - t_{i-1}^*} \beta_{i+1,j}^{k-1} \end{aligned} \tag{4.2}$$

Ainsi, on peut mettre le processus d'insertion sous forme matricielle :

$$\mathbf{N} = \beta^T \tilde{\mathbf{N}}$$

où $\mathbf{N}$ et $\tilde{\mathbf{N}}$ sont les vecteurs respectifs des fonctions de base originales et subdivisées. β^T est la matrice transposée des coefficients $\beta_{i,j}^d$.

A l'aide de cette expression matricielle et de l'équation de la courbe B-spline, on en déduit :

$$\begin{aligned} &\mathbf{p}^T \mathbf{N} = \tilde{\mathbf{p}}^T \tilde{\mathbf{N}} = \mathbf{p}^T \beta^T \tilde{\mathbf{N}} \\ &\Leftrightarrow \tilde{\mathbf{p}} = \beta \mathbf{p} \end{aligned}$$

où $\tilde{\mathbf{p}}$ et $\mathbf{p}$ sont les vecteurs des points de contrôle originaux et subdivisés. Grâce à cette relation, on peut déterminer les points de contrôle de la version raffinée de la B-spline.

L'insertion d'un nœud et de son point de contrôle associé permet de mettre en place une multi-résolution locale contraire à la subdivision d'une B-spline uniforme qui ne peut changer que globalement.

La suppression d'un nœud (et donc d'un point de contrôle) est bien plus délicate puisque si l'insertion peut se produire à tout moment, la suppression ne peut avoir lieu que lorsque la courbe est dans une configuration bien déterminée. Nous ne rentrons pas dans les détails, le lecteur pourra trouver toutes les informations dans les travaux suivants[EH95, LM87, Han87, Til92].

Détaillons maintenant la multi-résolution au sens mécanique du terme au travers de quelques travaux existants.

4.1.2 Multi-résolution mécanique

Un modèle est basé sur un nombre fini de degrés de liberté, or la plupart des objets réels sont constitués d'une infinité de degrés de liberté. Ce décalage entre l'objet réel et le modèle pose des limites comportementales. Par exemple, un tissu masses-ressorts composé de quatre points simulés ne permet pas d'obtenir le même comportement qu'un tissu composé de cent points simulés. De la résolution du modèle dépend le comportement émergent. Or, pour certaines applications, un modèle possède une résolution donnée pour des raisons de rapidité de calcul mais, il doit cependant être capable de simuler un comportement physique attendu lors d'une action interne ou externe. Le comportement qui doit être le résultat de la simulation peut être global à tout l'objet ou seulement local. Dans le premier cas, un changement de résolution en affinant globalement le modèle introduit inexorablement une lourdeur non négligeable dans les calculs mécaniques. Une autre solution est d'adopter un autre modèle mécanique. Dans le second cas, l'émergence d'un comportement local peut être mis en valeur en changeant localement la résolution du modèle physique. Cette adaptation du modèle densifie localement le modèle mais relativement peu par rapport à sa discrétisation totale. Donc les temps de calcul mécanique sont légèrement plus longs mais restent accessibles. Ce dernier point pouvant être compensé si le modèle est également capable du processus inverse, à savoir, réduire localement sa résolution si le comportement le permet.

Le but de la multirésolution mécanique est de donner à un modèle physique, possédant forcément un nombre fini de degrés de liberté, la possibilité d'adapter sa propre résolution pour obtenir un comportement plus réaliste et lui apporter plus de souplesse.

La multirésolution mécanique peut être décomposée en deux catégories, la première catégorie regroupe les algorithmes adaptatifs et la seconde recense les algorithmes multirésolutions. Les algorithmes adaptatifs se basent sur plusieurs représentations de l'objet et activent certaines parties de certaines représentations. Alors que les algorithmes multirésolutions ne possède qu'un seul exemplaire de l'objet, celle de la résolution courante, qui évolue à chaque changement de résolution. Les algorithmes adaptatifs se basent donc sur des représentations pré-calculées alors que les algorithmes multirésolutions modifient dynamiquement la résolution courante.

La multirésolution mécanique est étudiée dans beaucoup de domaines différents, comme l'animation humaine, les environnements de simulation adaptative, l'adaptation des modèles déformables...

Dans le contexte de l'animation humaine, Giang et al. [GMPO00] proposent un environnement adaptatif de niveaux de détails pour l'animation anthropomorphe nommé ALOHA (*Adaptative LOD for Human Animation*). Cet environnement est basé sur une décomposition en quatre couches : les os, les muscles, la graisse et la peau. Le moteur d'animation possède un module nommé *LOD Resolver* qui détermine à chaque itération les valeurs des paramètres de niveau de détails. Ces paramètres s'appliquent à la fois à l'animation physique ou cinématique et au niveau de détails géométrique. Chacune des quatre couches possède sa propre résolution qui peut être changée par décision du *LOD Resolver*. Les quatre couches étant liées entre elles par des contraintes, des ressorts...Le contrôle des quatre couches peut donc se faire au travers d'un seul paramètre.

Toujours dans un domaine lié à l'homme, Florence Bertaille et al.[BKCN03] proposent de simuler une chevelure à l'aide d'un algorithme multirésolution capable de changer la résolution des mèches en utilisant deux algorithmes, l'un pour la séparation et l'autre pour la fusion. En utilisant les critères adéquates, la chevelure est ainsi simulée par des mèches grossières où les mouvements sont faibles et par des mèches plus précises dans les zones où le mouvement est plus rapide.

L'un des travaux pionniers dans le domaine de la multi-résolution mécanique est la proposition de Hutchinson et al.[HPH96] qui décrit une méthode de raffinement automatique pour les modèles 2D masses-ressorts. Le critère de changement de résolution est relatif à l'angle formé par deux ressorts normalement alignés. Au delà d'un seuil, cette zone du maillage est subdivisée en insérant des masses et des ressorts. Les nouveaux points sont interpolés linéairement, ils sont donc défini au centre des ressorts du niveau inférieur. Ils sont pourvus de la même masse que les points du niveau inférieur, ce qui change la masse de l'objet. Les nouveaux ressorts ont une raideur double par rapport à ceux du niveau inférieur, puisqu'ils sont exactement deux fois plus court.

Cette technique permet une subdivision effective du maillage discret en gardant la même élasticité, mais ne permet pas de conserver la masse de l'objet. L'insertion des nouvelles masses s'effectuent obligatoirement au milieu de deux points du niveau inférieur, ce qui laisse assez peu de liberté et laisse supposer d'autre niveau de subdivision par phénomène de dichotomie.

Grinspun et al. [GKS02] proposent un environnement de travail (nommé CHARMS : *Conforming, Hierarchical, Adaptive Refinement MethodS*) spécifique pour la simulation adaptative. Pour cela, les auteurs se basent sur les splines et une décomposition en ondelettes. Plutôt que de raffiner les éléments d'un maillage élément fini, CHARMS raffine les fonctions de base utilisées. Suivant le problème traité, le modèle élément fini discrétise l'objet (1D ou 2D) en un maillage et détermine des fonctions d'approximation pour évaluer les champs de forces et de déplacements en tout point de la courbe ou de la surface. Les auteurs proposent d'utiliser les fonctions d'approximation des B-splines, soit linéaires soit d'ordre supérieur.

La boucle de simulation se déroule de la manière suivante :

```
SIMULATION()
1  while t < t_fin
2        do
```

3 Prédire : mesurer l'erreur et construire les ensembles $\mathcal{B}^+$ et $\mathcal{B}^-$
4 Adapter :
5 $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{B}^+ \cap \mathcal{B}^-$
6 Maintenir l'indépendance de $\mathcal{B}$ (suppression des redondances)
7 Résoudre : les équations linéaires ou non à l'aide de $\mathcal{B}$
8 $t \leftarrow t + \delta_t$

Une itération se déroule donc en trois étapes, le choix du raffinement à l'aide d'un oracle qui prédit les régions de l'objet qui ont besoin d'une meilleure résolution ou au contraire qui peuvent se contenter d'une résolution plus grossière. L'algorithme construit alors les fonctions de base qui permettent de définir l'espace d'approximation. Ensuite, l'espace d'approximation est adapté en mettant à jour les fonctions de base et en maintenant une cohérence entre elles. Ainsi il ne doit pas y avoir de redondance. Enfin, la troisième et dernière étape consiste à résoudre le système mécanique linéaire ou non-linéaire. Pour cela, le système d'équations doit être écrit à l'aide des fonctions de base choisies.

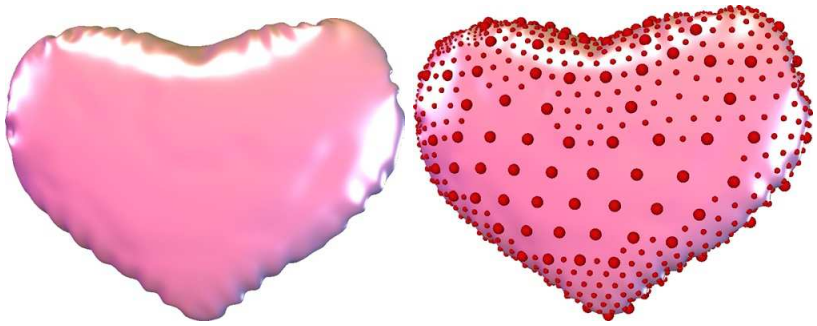


FIG. 4.2 – Mise en évidence du raffinement des fonctions de base (Extrait de [GKS02]). Les sphères rouges indiquent les fonctions de base actives.

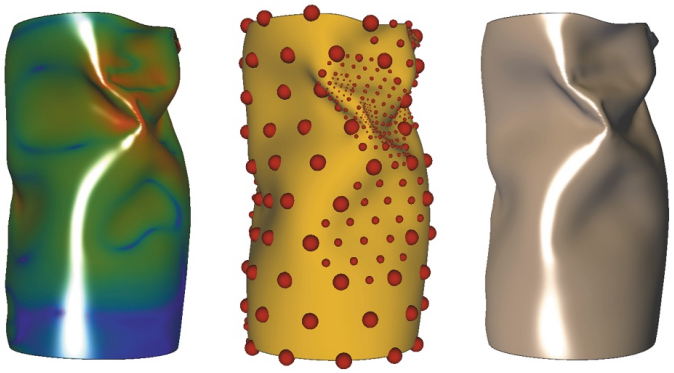


FIG. 4.3 – Mise en évidence du raffinement des fonctions de base (Extrait de [GKS02])

De cette manière, une simulation peut automatiquement affiner les fonctions de base dans les régions courbées (cf. figure (4.2)) ou aux endroits où la déformation est plus importante (cf. figure (4.3)). Sur cette dernière figure, le schéma de gauche indique en rouge les zones de fortes déformations en flexion, le schéma du milieu montre que ces zones possèdent une grande concentration de fonctions de base affinées et que, par conséquent, l'animation montre bien les courbures de l'objet et ses rides affinées (schéma de droite).

Dans le même esprit, Capelle et al.[CGC+02] proposent un environnement multirésolution de simulation dynamique d'objets déformables. Dans cette proposition, les objets sont simulés par une formulation éléments finis des équations physiques de Lagrange. De ce fait, les auteurs font apparaître un vecteur

déplacement et l'expriment à l'aide d'une combinaison linéaire de fonctions de base hiérarchiques. De plus, les objets sont englobés dans une grille non régulière dont chaque élément posséde une résolution active des fonctions de base. Ainsi, le modèle peut être simulé à l'aide de fonctions de base grossières dans un élément de la grille et à une résolution plus fine dans un autre élément. Lorsque le choix est fait de changer de résolution sur un élément de la grille, il suffit de mettre à jour les matrices de masse et de rigidité et, si l'élasticité employée est non linéaire, de mettre à jour également les termes concernant cette énergie. Le critère de décision se base sur deux seuils de déformation. Le premier est le seuil d'activation qui permet de prendre en compte les fonctions de base plus fines. Le second est le seuil de désactivation qui, au contraire, permet de reprendre en considération la fonction de base plus grossière. Il est conseillé de choisir un seuil de désactivation plus petit que celui d'activation pour éviter que le système ne change immédiatement de résolution après une activation ou n'oscille entre deux résolutions.

La figure (4.4) compare l'utilisation de grille régulière et non régulière pour l'animation d'un dragon déformable. Les schémas (a) et (d) montrent les positions au repos des objets déformables ainsi que les grilles utilisées au départ (respectivement régulière et non régulière). Les schémas (b) et (e) mettent en évidence l'animation du dragon lors de la prise en compte des contraintes de positions modélisées par les sphères noires. Sur cet exemple, le dragon est étiré et sa bouche ouverte. Les schémas (c) et (f) montrent les configurations déformées du dragon. On remarque bien que le résultat avec la grille non-régulière (schémas (d), (e) et (f)) est plus réaliste que celui où une grille régulière est utilisée (schémas (a), (b) et (c)).

La figure (4.5) met en évidence l'adaptation de la résolution lors de déformations. Le schéma (a) présente le dragon dans sa position au repos. Le schéma (b) indique le volume englobant formé par la grille non-régulière. Le schéma (c) montre la structure hiérarchique du volume englobant où les sphères rouges correspondent au niveau 0 des fonctions de base, les sphères vertes au niveau 1 et les sphères bleues au niveau 2. Le schéma (d) montre l'approximation en tétraèdre du dragon utilisée par la méthode des éléments finis. Le schéma (e) souligne l'utilisation d'une contrainte de positionnement indiquée par la présence de la sphère noire. Ainsi, cette contrainte impose d'utiliser localement des fonctions de base raffinées comme l'indique le schéma (f).

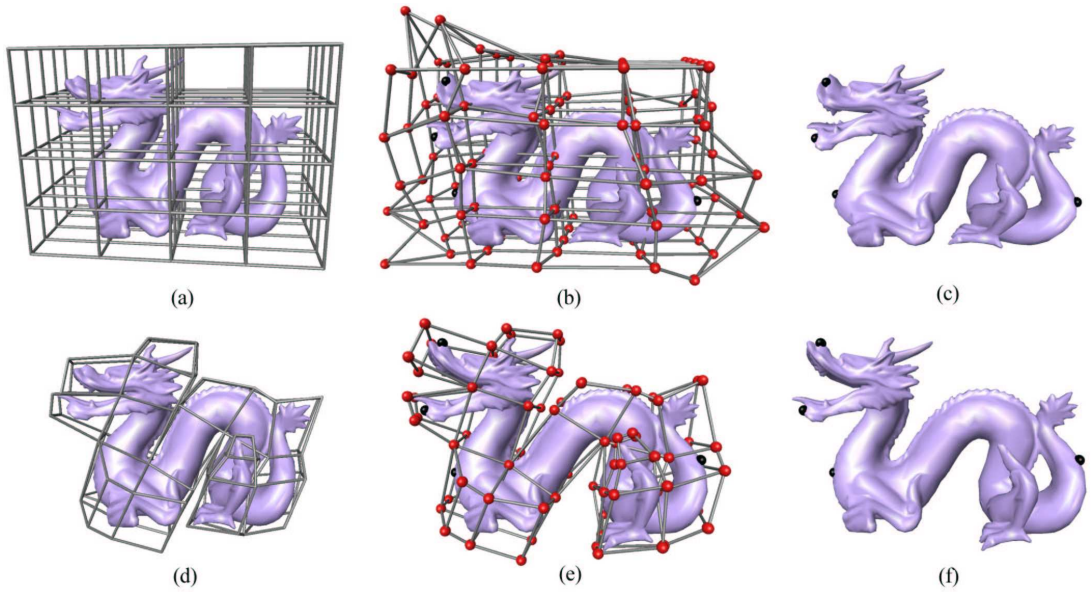


FIG. 4.4 – Comparaison des grilles régulières et non régulières lors de déformations (Extrait de [CGC+02])

En se basant également sur une grille englobante, Nocent et al.[NNR01] proposent de simuler un objet à l'aide d'un volume englobant constitué de deux splines surfaciques superposées. Seuls les deux splines sont simulées physiquement à l'aide des équations de Lagrange. Grâce à l'aspect continu des mo-

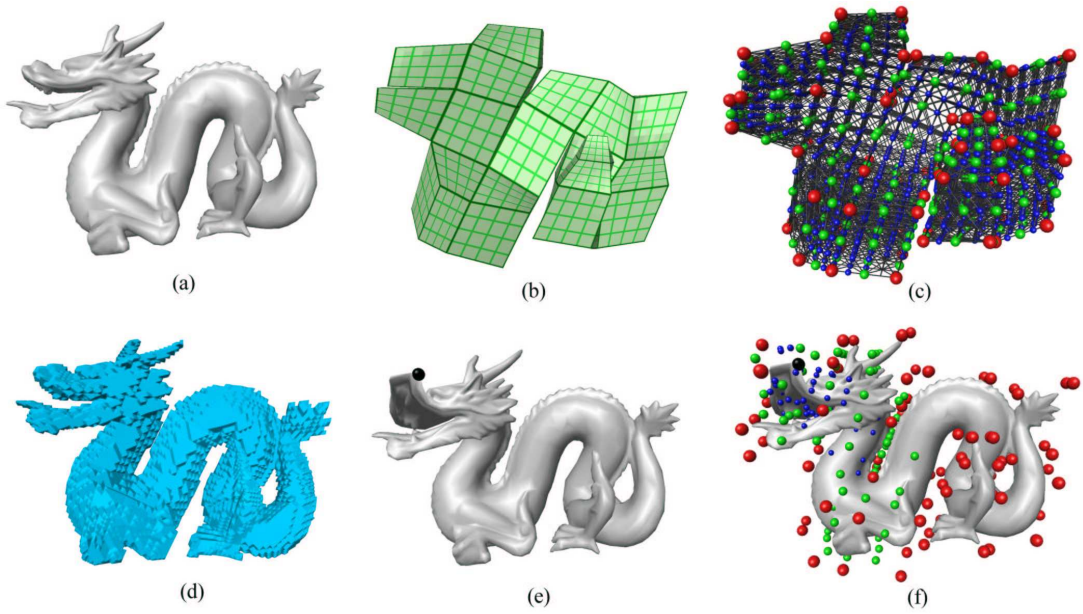


FIG. 4.5 – Mise en évidence de la multirésolution en présence de déformations (Extrait de [CGC+02])

dèles splines, la déformation du volume englobant se répercute en tout point de l'objet simulé. De part cette technique, il est possible de simuler physiquement des objets complexes, qui demandent un grand nombre de degrés de liberté, à l'aide uniquement des degrés de liberté des deux surfaces splines. Cette technique permet donc de décorréler le nombre de degrés de liberté simulé du nombre réel de degrés de liberté de l'objet. Autrement dit, elle permet de réduire le jeu de paramètres de la simulation.

Debunne et al.[DDCB00, DDCB01] proposent une simulation en temps réel d'objets déformables adaptatifs. Pour cela, l'objet déformable existe à différentes résolutions. A chaque itération, les différentes résolutions sont constituées de nœuds possédant la propriété *actif* ou au contraire *fantôme*. Si un nœud d'une résolution est actif, c'est que la résolution est celle choisie dans cette zone de l'objet. Si c'est une autre résolution qui est choisie, le nœud est invalidé, il devient alors fantôme. Les nœuds actifs sont simulés et intégrés alors que les nœuds fantômes nécessaires au calcul des déplacements de certains nœuds actifs sont juste interpolés sur la résolution active à cet endroit. De cette manière, il n'est pas nécessaire d'avoir des nœuds communs entre les différentes résolutions, les nœuds limites sont les nœuds fantômes et ils calculent leurs données par interpolation sur la résolution adéquate. Cette interface est schématisée sur la figure (4.6).

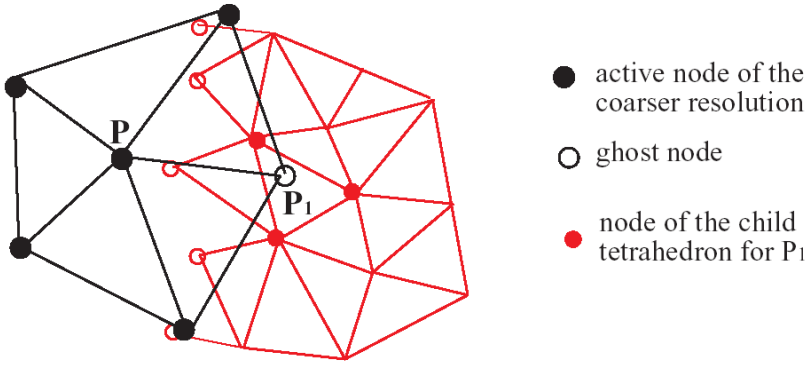


FIG. 4.6 – Nœuds actifs et nœuds fantômes (Extrait de [DDCB00])

Les auteurs font remarquer que le changement de résolution s'effectue sans aucune ambiguïté dans le choix des nœuds à activer puisqu'une résolution plus fine est déterminée par les nœuds présents dans la région de Voronoi du nœud à désactiver. Or, les régions de Voronoi d'un objet forment une partition de l'espace. Ainsi, un nœud de la résolution plus fine ne peut faire partie que d'une seule de ces régions et n'est donc lié qu'à un seul nœud père. De plus deux nœuds d'une résolution grossière ne peuvent pas partager le même nœud de la résolution plus fine (cf. figure (4.7)).

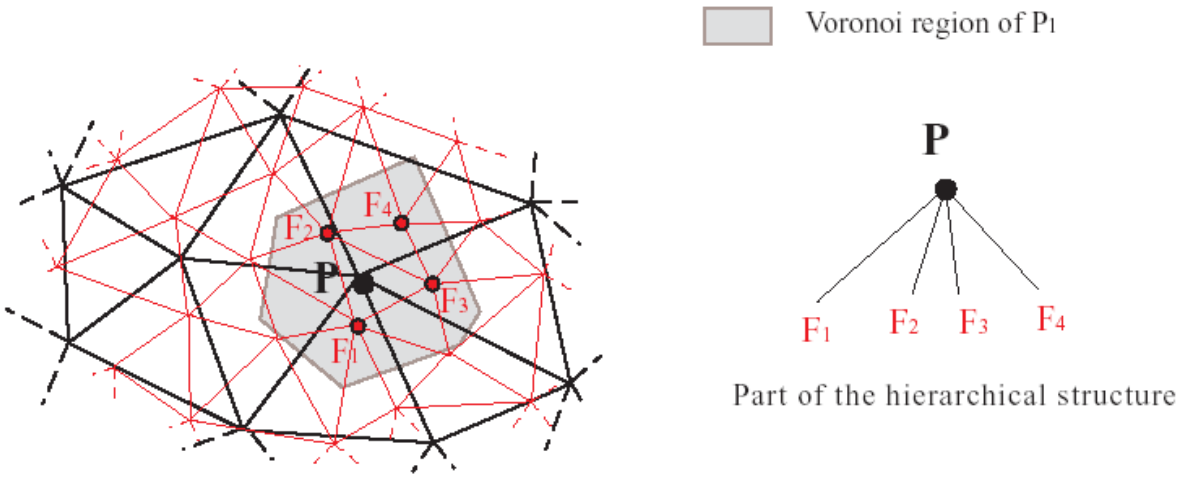


FIG. 4.7 – Relations entre les nœuds de deux niveaux de résolution consécutifs (Extrait de [DDCB00])

Les auteurs proposent d'adapter la résolution mécanique des modèles déformables à l'aide d'un critère de qualité. Ce critère se base sur le fait que le modèle différences finies utilisé par les auteurs est une approximation au premier ordre. Donc la mesure de l'erreur entre le second ordre et le premier ordre peut servir d'indicateur et donc de critère. Ils proposent également d'adapter le pas de temps de simulation en utilisant le critère de Courant pour déterminer le pas de temps maximum utilisable pour un matériau donné :

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

où h, ρ_0, λ et μ sont des grandeurs scalaires caractéristiques du matériau simulé.

Il faut noter également les travaux de Carlson et Hodgins[CH97] sur les niveaux de détails mécaniques. Ces travaux proposent plusieurs modèles d'animation pour simuler un objet et un ensemble de critères permettant de changer dynamiquement le modèle d'animation. Par exemple, suivant le critère réalisé, un objet (ou une partie d'un objet) peut être simulé par des lois physiques, animé par la cinématique inverse ou même être fixe.

4.1.3 Discussion

On peut classer ces travaux en deux catégories, ceux qui proposent une multi-résolution sur un maillage discret et ceux qui proposent une multi-résolution directement sur le modèle continu.

Multi-résolution mécanique d'un maillage discret :

Les modèles basés sur la méthode des éléments finis ou une méthode dérivé (masses-tenseurs) sont soumises à la discrétisation du maillage. Lorsqu'une résolution change, c'est un tout autre maillage qui est considéré localement ou globalement, avec une gestion entre différents niveau de résolution si le changement de résolution est local. La nouvelle résolution est formé d'éléments construits par rapport aux éléments du maillage supérieur. Par exemple, on peut définir la résolution plus fine d'un tétraèdre en définissant un nouveau point comme étant l'iso-barycentre de ses sommets. Ce

type de construction forment un nouveau maillage figé sur la discrétisation. Ceci apporte donc des décalages entre les résolutions, ce qui implique que le modèle simulé n'est plus le même.

Les travaux de Debunne[DDCB00, DDCB01] font également partis de cette catégorie puisque dans cette conception un objet est constitué de nœuds discrets à différentes résolutions. Lors d'un changement de résolution, le choix est fait d'utiliser certaines parties d'une résolution en faisant particulièrement attention aux nœuds situés à la limite de deux résolutions actives. La multi-résolution s'appuie donc sur plusieurs maillages discrets à différentes résolutions d'un même modèle.

Multi-résolution mécanique d'un modèle continu :

Ce type de multi-résolution s'appuie sur les propriétés continues du modèle physique, comme les travaux de Grinspun et al.[GKS02] ou encore ceux de Capelle et al.[CGC+02]. Le travail de Nocent et al.[NNR01] entre également dans cette catégorie puisqu'il permet de réduire le nombre de degrés de liberté pour simuler un modèle, tout en simulant toujours le même modèle, qu'il soit discret ou continu.

Lorsqu'une technique multi-résolution est basée sur le maillage discret d'un modèle, un soin tout particulier doit être apporté pour garder la continuité temporelle de la simulation. Et plus encore si plusieurs résolutions cohabitent en même temps. En effet, le fait de n'avoir qu'un maillage discret sans modèle continu pose le problème que tout changement du maillage, change le modèle.

Au contraire, avec une technique multi-résolution basée sur le modèle continu, il est plus facile d'assurer la continuité du modèle lors d'un changement de résolution. Puisqu'un changement de résolution ne fait que modifier le nombre de degrés de liberté permettant de simuler le modèle continu, mais ce modèle reste le même, et en particulier sa géométrie. Bien entendu, le fait de modifier les degrés de liberté va influencer le comportement du modèle qui réagit de façon moins grossière. Mais, le changement de résolution s'effectue sans discontinuité dans la simulation physique.

Nous proposons un modèle multi-résolution qui est, du point de vue mécanique, plus naturel. Le principe est de conserver le même modèle physique lors d'un changement de résolution et de permettre d'adapter la résolution en tout point du modèle. Notre proposition fait donc partie de la catégorie des modèles multi-résolutions basés sur un objet continu. Exposons maintenant ce modèle.

4.2 Multi-résolution sur la spline dynamique

Le but est ici d'introduire un cadre multi-résolution dans le modèle géométrique sous-jacent au modèle spline dynamique, puis, d'observer l'incidence de ce changement de résolution sur le modèle mécanique.

Pour cela, nous proposons d'adapter automatiquement la résolution locale de notre modèle spline dynamique en insérant/supprimant un seul point de contrôle à la fois. De ce fait, le modèle est capable d'adapter localement sa configuration géométrique et mécanique en fonction de critères géométriques et/ou physiques. L'étude se décompose en plusieurs étapes, la première détaille le mécanisme d'insertion, la seconde expose la suppression d'un point de contrôle et la troisième étudie la stabilité du modèle mécanique résultant. Une quatrième étape discute de divers travaux menés sur le thème de la découpe d'objets simulés physiquement et discute de ce problème de découpe sur notre modèle multi-résolution.

4.2.1 Technique d'insertion

L'insertion d'un nouveau point de contrôle va de pair avec l'insertion d'un nouveau nœud dans le vecteur de nœud. Ainsi, l'insertion n'est rendue possible que pour les splines raffinables, comme les B-splines non uniformes. Ce principe d'insertion sur une B-spline a été détaillé en section (4.1.1.2). L'ensemble de la technique multi-résolution est donc restreinte à la classe des splines raffinables, ce qui éliminent le choix des Catmull-Rom et des B-splines uniformes.

En appliquant cet algorithme d'insertion de point, on obtient une spline géométrique inchangée au niveau de la forme mais possédant un point de contrôle supplémentaire. Donc le modèle résultant est capable de prendre des configurations que le modèle de base ne pouvait pas prendre, notamment des

configurations plus courbées. Comme le modèle de spline dynamique se base entièrement sur le modèle géométrique sous-jacent, l'insertion d'un point va engendrer des modifications au niveau mécanique. Ces modifications affectent de façon continue les structures de données et les énergies de déformation.

4.2.1.1 Les structures de donnée

La dimension de l'espace des paramètres du système matériel passe de $3n$ à $3(n+1)$ puisqu'un nouveau point de contrôle introduit 3 degrés de liberté supplémentaires. Ceci étend l'ensemble des structures de données sachant qu'elles sont pré-allouées avec une taille maximum, aucune réallocation n'est donc nécessaire en cours de simulation. Certaines structures sont considérées comme variables : la matrice des contraintes L, les matrices intermédiaires pour la résolution $M^{-1}L^T$ et R, le vecteur intermédiaire $\mathbf{O}$ pour la résolution et le vecteur $\mathbf{B}$ regroupant le bilan des énergies potentielles. D'une itération à l'autre, pour ces structures, l'insertion d'un nouveau point de contrôle ne fait que changer leur taille mais ne requiert aucun recalcul au moment de l'insertion. Par contre, les structures considérées comme constantes au cours de la simulation requièrent une extension de la structure et un recalcul efficace des données. Détaillons maintenant le processus de mise à jour des différentes structures de donnée constantes au cours du temps :

La matrice des masses généralisées

La matrice des masses $\mathcal{M}_{(3n\times 3n)}$ est donc étendue à $\mathcal{M}_{((3n+3)\times(3n+3))}$. En rappelant qu'elle se décompose en trois blocs diagonaux identiques M (cf. équation (2.12)), on ne recalcule que la matrice $M_{(n\times n)}$ étendue en $M_{((n+1)\times(n+1))}$. La complexité d'un recalcul complet de la matrice M est en $\mathcal{O}(n^2)$ puisque l'on doit recalculer les n^2 coefficients et que le calcul d'un coefficient s'effectue à temps constant (cf. équation (2.13)).

Si l'on désire n'effectuer que le minimum d'opérations pour définir la nouvelle matrice des masses généralisées, il faut commencer par décaler les données. Le nombre de données à décaler dépend de l'indice du nouveau point de contrôle, si le point est au début de la spline, il oblige le décalage de quasiment tous les coefficients de la matrice (c'est le pire des cas). Si maintenant il est en fin de spline, seuls les coefficients des points de contrôle suivants dans la matrice doivent être décalés. Cette opération possède donc une complexité dans le pire des cas de $\mathcal{O}(n^2)$. On remarque que la complexité est la même que si l'on recalcule tous les coefficients. En pratique, un déplacement mémoire est beaucoup plus rapide que le recalcul d'un coefficient (27000 fois sur un pentium IV 2.4 GHz avec une architecture mémoire de type DDR).

L'algorithme d'insertion a été détaillé de manière générale dans la section (4.1.1.2). On considère ici le cas pratique d'une B-spline cubique sans perte de généralité. Soit une B-spline cubique ($d=3$) définie par un vecteur $\mathbf{q}^*$ de points de contrôle et un vecteur de nœuds $\mathbf{t}^*$. Soit une version raffinée de cette B-spline définie par un vecteur de points de contrôle $\mathbf{q}$ et un vecteur de nœuds $\mathbf{t}$ qui n'a de différent avec $\mathbf{t}^*$ qu'un nœud inséré entre $\mathbf{t_{i+d}}$ ($\mathbf{t_{i+3}}$) et $\mathbf{t_{i+d+1}}$ ($\mathbf{t_{i+4}}$), alors les points de contrôle $\mathbf{q}$ sont déterminés par les relations suivantes :

$$\begin{cases} \forall k < i \ , \ \mathbf{q_k} = \mathbf{q_k^*} \\ \mathbf{q_i} = \beta^3_{i,i}\mathbf{q_i^*} + (1-\beta^3_{i,i})\mathbf{q_{i-1}^*} \\ \mathbf{q_{i+1}} = \beta^3_{i+1,i+1}\mathbf{q_{i+1}^*} + (1-\beta^3_{i+1,i+1})\mathbf{q_i^*} \\ \mathbf{q_{i+2}} = \beta^3_{i+2,i+2}\mathbf{q_{i+2}^*} + (1-\beta^3_{i+2,i+2})\mathbf{q_{i+1}^*} \\ \forall k > i+2 \ , \ \mathbf{q_k} = \mathbf{q_{k-1}^*} \end{cases} \tag{4.3}$$

où $\beta^r_{i,j}$ est défini par l'équation (4.2).

Dans ce cas, les points modifiés sont les points d'indices i, $i+1$ et $i+2$. On remarque que les points d'indice supérieur ou égal à $i+d$ en général sont simplement décalés. Ainsi, la matrice des masses généralisées subit également ce décalage à partir de l'indice $i+d$ sur les deux dimensions :

$$M_{(n\times n)} = \begin{pmatrix} M_{11} & \dots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{n1} & \dots & M_{nn} \end{pmatrix}$$

et une fois étendue, $M_{(n\times n)}$ devient $M_{((n+1)\times(n+1))}$:

$$
\begin{array}{c}
\\
\\
\\
\\
\\
\\
\\
\\
\end{array}
\begin{array}{c}
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad i+d \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \downarrow \\
\left(\begin{array}{ccccccc}
M_{11} & \dots & M_{1(i+d-1)} & M_{1(i+d-1)} & M_{1(i+d)} & \dots & M_{1n} \\
\vdots & \ddots & & & & & \vdots \\
M_{(i+d-1)1} & \dots & M_{(i+d-1)(i+d-1)} & M_{(i+d-1)(i+d-1)} & M_{(i+d-1)(i+d)} & \dots & M_{(i+d-1)n} \\
M_{(i+d-1)1} & \dots & M_{(i+d-1)(i+d-1)} & M_{(i+d-1)(i+d-1)} & M_{(i+d-1)(i+d)} & \dots & M_{(i+d-1)n} \\
M_{(i+d)1} & \dots & M_{(i+d)(i+d-1)} & M_{(i+d)(i+d-1)} & M_{(i+d)(i+d)} & \dots & M_{(i+d)n} \\
\vdots & & & & & \ddots & \vdots \\
M_{n1} & \dots & M_{n(i+d-1)} & M_{n(i+d-1)} & M_{n(i+d)} & \dots & M_{nn}
\end{array}\right)
\end{array}
\begin{array}{l}
\\
\\
\\
\longleftarrow i+d \\
\\
\\
\\
\\
\end{array}
$$

Les coefficients rouges sont les coefficients décalés.

Après avoir décalé les coefficients, un recalcul de certains coefficients est effectué. Les coefficients en question sont ceux qui sont affectés par la modification du vecteur de nœuds (cf. figure (4.8)). Or, les coefficients sont des intégrales de produit de fonctions de base de spline (cf. équation (2.13)). Et, certaines fonctions de base sont directement affectées par la modification du vecteur de nœuds (cf. figure (4.9)).

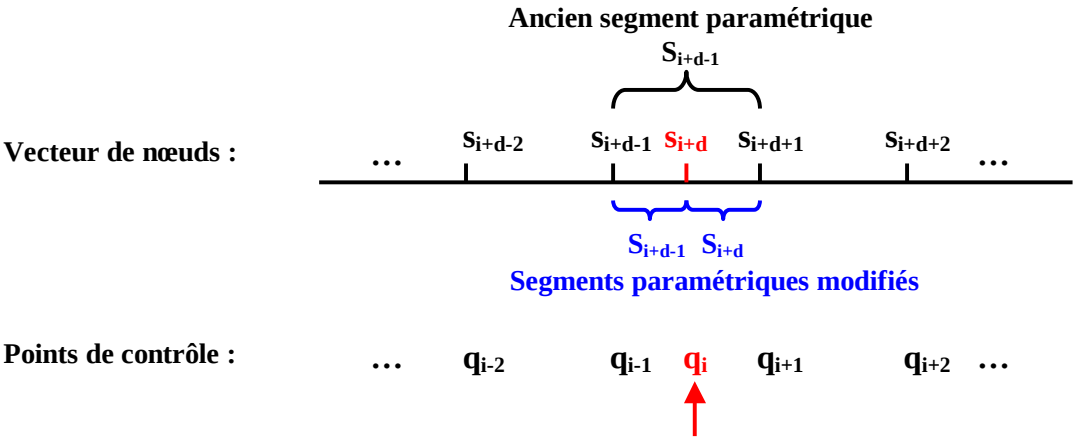


FIG. 4.8 – Conséquence d'une insertion sur le vecteur de nœuds

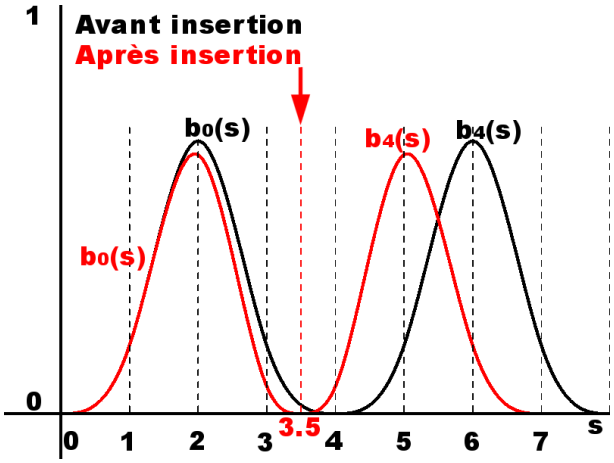


FIG. 4.9 – Conséquence d'une insertion sur les fonctions splines cubiques

Sur ce schéma, l'insertion a lieu à l'abscisse paramétrique 3.5, les fonctions de base affectées (i.e. dont la forme change, peu importe le support) par cette insertion sont celles d'indice entre 0 et 4 inclus. On trouve donc cinq fonctions de base modifiées.

Définition : *On définit un segment paramétrique S_j par l'intervalle* $[s_j, s_{j+1}[$.

L'insertion du nouveau point de contrôle (d'indice i) modifie le vecteur de nœuds en ajoutant une entrée supplémentaire à l'indice $i+d$ où d est le degré de la B-spline. Ce nouveau nœud altère la configuration des deux segments paramétriques adjacents : S_{i+d-1} et S_{i+d}. Ainsi, toutes les fonctions de base des splines qui ont un support en intersection non nulle avec l'un de ces deux intervalles subissent la modification. Or, le support d'une fonction de base de spline est déterminé à l'aide de la propriété de localité des B-splines par :

$$S_{b_j} = [s_j, s_{j+k}[= \bigcup_{l=0}^{k-1} S_{j+l}$$

où $k = d+1$ est l'ordre de la localité des B-splines utilisées.

De plus, les fonctions splines qui ont un support en intersection non nulle avec S_{i+d-1} ou S_{i+d} sont définies par :

$$\begin{array}{llll} S_{b_j} \bigcap S_{i+d-1} & \neq 0 & \text{pour} & i+d-k \leq j \leq i+d-1 \\ S_{b_j} \bigcap S_{i+d} & \neq 0 & \text{pour} & i+d-k+1 \leq j \leq i+d \end{array}$$

Ainsi, les fonctions de base dont la forme change après l'insertion du nouveau point sont celles dont l'indice j vérifie :

$$i+d-k \leq j \leq i+d$$

Pour ces fonctions de base, leur produit par une autre fonction de base (indice l) change également modifiant les coefficients correspondants (M_{jl} et M_{lj}) dans la matrice des masses généralisées.

Comme la matrice est symétrique, $k+1$ lignes et $k+1$ colonnes de la matrice sont à recalculer. Du fait de la symétrie, on ne recalcule que les colonnes pour en déduire les lignes symétriques. De plus, la matrice M est bande de largeur $2k-1$, cette propriété nous permet de ne recalculer que les éléments dans l'intersection entre les $k+1$ colonnes et la bande de largeur $2k-1$ (sachant que $k = d+1$, on a aussi $2k-1 = 2d+1$). Voici un exemple de mise à jour pour une B-spline cubique :

$$\begin{array}{r} \\ \\ \\ \\ i+d-k \longrightarrow \\ \\ \\ \\ \\ i+d \longrightarrow \\ \\ \\ \\ \\ \end{array}
\begin{array}{c}
\begin{array}{cccccccccccc} & & & & i+d-k & & & & i+d & & & \\ & & & & \downarrow & & & & \downarrow & & & \end{array} \\
\begin{pmatrix}
\ldots & & & & X & X & X & X & X & & & \ldots \\
\ldots & & & & X & X & X & X & X & & & \ldots \\
\ldots & & & & X & X & X & X & X & & & \ldots \\
\ldots & & & & X & X & X & X & X & & & \ldots \\
\ldots & X & X & X & X & X & X & X & X & & & \ldots \\
\ldots & & X & X & X & X & X & X & X & & & \ldots \\
\ldots & & & X & X & X & X & X & X & X & & \ldots \\
\ldots & & & & X & X & X & X & X & X & X & \ldots \\
\ldots & & & & X & X & X & X & X & X & X & \ldots \\
\ldots & & & & X & X & X & X & X & & & \ldots \\
\ldots & & & & X & X & X & X & X & & & \ldots \\
\ldots & & & & X & X & X & X & X & & & \ldots \\
\ldots & & & & X & X & X & X & X & & & \ldots
\end{pmatrix}
\end{array}$$

Les X symbolisent les coefficients des $k+1$ colonnes, la couleur rouge indique les coefficients dans l'intersection entre les $k+1$ colonnes et la bande. La couleur bleu est relative aux données dupliquées par symétrie, ces données sont donc à l'intersection de la bande et des $k+1$ lignes.

Pour effectuer cette mise à jour, à chaque colonne j traitée, on définit l'indice de début de ligne l_{debut} et l'indice de fin de ligne l_{fin} par :

$$l_{debut} = j-d \qquad l_{fin} = j+d$$

On calcule les coefficients en reportant également les coefficients symétriques (dans l'absolu, certains coefficients sont calculés deux fois, l'algorithme peut être optimisé pour prendre en compte cette redondance).

De cette manière, on recalcule effectivement $(k+1)*(2k-1) = 2k^2+k-1$ coefficients. Considérant l'ordre de localité k comme une constante du modèle B-spline, la mise à jour s'opère en temps constant.

La décomposition LU de la matrice des masses généralisées

Comme la résolution du système sans contrainte a besoin de la décomposition LU de la matrice des masses généralisées, cette décomposition est également recalculée. Ici, aucune accélération n'a été mise en place.

L'inverse de la matrice des masses généralisées

De plus, la résolution du système contraint a besoin de l'inverse de la matrice $\mathcal{M}$, il faut donc recalculer la matrice M^{-1} également. Or, cette matrice passe de la dimension $(n \times n)$ à la dimension $((n+1)\times(n+1))$.Il pourrait être intéressant d'appliquer un schéma de calcul introduisant uniquement les modifications dans la matrice inverse, comme le schéma de Shermann-Morisson-Woodbury. Cependant, dans sa version originale, ce schéma n'est pas utilisable tel quel pour des matrices de dimensions variables. Pour le moment, l'inverse est donc recalculée entièrement.

Le vecteur de gravité

Une partie de la gravité est précalculée (cf. équation (2.16)). Ainsi, l'insertion d'un nouveau point de contrôle demande une mise à jour de ce vecteur. Il est composé d'intégrales de fonctions de base de spline. Ainsi, seules les fonctions de base dont la forme a changé sont prises en compte. Comme les b_i modifiés sont ceux d'indice j vérifiant $i+d-k \leq j \leq i+d$, les coefficients recalculés du vecteur sont ceux d'indice j vérifiant :

$$i+d-k \leq j \leq i+d$$

On recalcule donc $k+1$ éléments. Comme k est considérée comme une constante du modèle spline, la mise à jour a donc une complexité en $\mathcal{O}(1)$.

Les points de contrôle

Comme le but est d'insérer un nouveau nœud dans le vecteur de nœuds, un point de contrôle associé à ce nouveau nœud est également introduit. Dans l'exemple d'une B-spline cubique, on utilise donc la formule (4.3) pour déterminer la position de ce point et les modifications sur les points de contrôle voisins.

4.2.1.2 Les énergies de déformations

Le comportement d'un objet déformable ne doit pas changer lorsqu'une insertion s'opére. Or, les énergies de déformation sont liées à la structure du modèle spline, autrement dit à son vecteur de nœuds. Comme l'insertion d'un point de contrôle est accompagné de l'insertion du nœud correspondant dans le vecteur de nœuds, la structure intrinsèque de la spline change. Donc, si l'on désire obtenir un comportement cohérent et continu dans le temps, les structures de données des énergies sont à adapter.

Evidemment, cette mise à jour dépend grandement du type de déformation utilisé. C'est pourquoi nous étudions dans un premier paragraphe les énergies discrètes puis, dans un second paragraphe, les énergies continues.

4.2.1.2.1 Energies discrètes Les énergies de déformation discrètes (cf. section (2.3.1)) sont modélisées par des ressorts placés sur une discrétisation de chaque segment paramétrique du modèle spline. Lors d'une insertion, la paramétrisation du modèle change, la discrétisation doit donc être recalculée et les ressorts redistribués sur cette nouvelle discrétisation. L'échantillonnage est effectué de manière paramétrique sur chaque segment spline de la même manière que dans la section (2.3.1). Ainsi, la discrétisation n'est

changée que sur les deux segments affectés par l'insertion. La configuration au repos de chaque ressort doit être déterminée à partir de la configuration au repos de la spline (généralement celle de départ). Donc, cette configuration au repos est conservée pour déterminer les longueurs (angles...) au repos des nouveaux ressorts lors d'une insertion.

Ainsi, la répartition des forces est différente lors d'un changement de résolution. Ceci résulte en une modification de la structure énergétique du modèle puisque la configuration des ressorts a localement changé, ne reproduisant plus le même comportement. Donc, lors d'une insertion, le modèle présente une certaine instabilité qui s'estompe au cours du temps pour finir par se stabiliser.

Une autre possibilité est de découper certains ressorts de la résolution courante à la manière de Hutchinson et al.[HPH96] en adaptant leur raideur. Les auteurs découpent un ressort en deux ressorts rectilignes produisant le même bilan de force. Dans notre cas, les extrémités du ressort sont des points de la spline d'abscisses paramétriques données. Lors d'une découpe, le point milieu est défini par la moyenne des abscisses paramétriques des extrémités et donc lui aussi est un point de la spline. Il n'est donc pas forcément aligné avec les points extrémités. Son insertion entraîne là aussi une discontinuité du comportement en flexion. Au final, cette technique donne la même distribution de ressort que la redistribution complète que nous avons choisi. Et étant donné la rapidité avec laquelle cette tâche peut être exécutée, il ne nous semble pas nécessaire de mettre en place ce processus de découpe de ressorts. Nous préférons nous attarder sur des modèles énergétiques continus qui offrent une meilleure stabilité du comportement.

Cette discontinuité temporelle est due à la discrétisation du modèle, elle peut donc être atténuée en augmentant la densité de l'échantillonnage ou mieux, supprimée en étudiant le cas des énergies de déformation continues (en élongation et flexion).

4.2.1.2.2 Energie continue d'élongation Cette énergie a été détaillée dans la section (2.3.2). Nous présentons ici le détail des mises à jour lors d'une insertion. La structure de donnée relative à l'énergie de déformation continue d'élongation met en place deux tableaux. Le premier stocke les termes B_{im} et le second stocke les termes B_{impq} (cf. équation (2.20)).

B_{im} :

$$B_{im} = \int_0^1 \frac{b_i'(s)b_m'(s)}{|\mathbf{T}(s,0)|}\,ds$$

D'abord, on voit apparaître le terme $\mathbf{T}(s,0)$ qui dépend de la configuration au repos de la courbe. Cette configuration au repos est ici considérée comme étant la configuration à $t = 0$. Ce terme est donc calculé à l'aide de la configuration de départ de la courbe qui est conservée tout au long de la simulation.

Ensuite, un terme B_{im} fait intervenir l'intégrale du produit de deux fonctions de base dérivées. Or une fonction et sa dérivée possèdent le même support et, le tableau B_{im} est symétrique. Il possède donc les mêmes propriétés que la matrice M. La mise à jour des termes B_{im} ressemble donc à la mise à jour de la matrice M : on commence par décaler les données sur les deux dimensions puis, on met à jour les termes qui en ont besoin comme avec la matrice M.

Comme beaucoup de ces termes sont nuls, la structure de données peut se contenter de n'allouer que l'espace mémoire nécessaire à stocker les données non nuls et non redondantes. Mais, la dimension du tableau n'étant que de deux, le choix s'est porté sur une structure classique de tableau pré-alloué (aucune réallocation dynamique) qui permet un accès direct aux données sans indirection. Dans ce contexte, la mise à jour des données B_{im} est donc optimale puisque l'on ne recalcule que le minimum de termes. La complexité de cette mise à jour est, comme pour la mise à jour de la matrice des masses généralisées, en $\mathcal{O}(1)$.

B_{impq} :

$$B_{impq} = \int_0^1 \frac{b_i'(s)b_m'(s)b_p'(s)b_q'(s)}{|\mathbf{T}(s,0)|^3}\,ds$$

Comme pour le tableau B_{im}, le terme $\mathbf{T}(s,0)$ est calculé à l'aide de la configuration au départ qui est conservée. Cependant, l'analogie s'arrête là puisque ce tableau est de dimension 4 et qu'il possède des symétries dans toutes les dimensions et beaucoup de termes sont nuls. On propose donc de mettre en place une structure de données particulière qui permet de ne stocker (et donc calculer) que les termes non nuls et sans redondance.

Comme cela a été indiqué dans la section (2.3.2), on fait le choix de vérifier la propriété $q \leq p \leq m \leq i$ pour éviter les symétries. On propose de stocker tous les termes dans un même tableau à une seule dimension, et d'utiliser quatre tables d'indirections pour accéder à une donnée déterminée par ses quatre indices.

Pour un indice i donné, on a la relation $m \leq i$ et la limite inférieure de l'indice m est donnée par le plus petit indice j vérifiant

$$S_{b_i} \bigcap S_{b_j} \neq 0$$

puisque si l'indice m est en dessous de cette limite, alors les fonctions b_i et b_m ont un support en intersection vide (il en est donc de même pour leurs dérivées) et donc le produit des deux fonctions est nul impliquant que les termes B_{impq} sont nuls quel que soit p et q.

La limite inférieure pour l'indice m est donc à nouveau donnée par la propriété de localité de la B-spline : les fonctions b_i et b_m s'étendent sur k segments paramétriques donc les fonctions b_m dont le support est en intersection non vide avec b_i sont celles d'indice m compris entre $i-(k-1)$ et $i+(k-1)$. L'indice m est donc borné par :

$$i-k+1 \leq m \leq i$$

Avec le même raisonnement, on détermine les bornes des indices p et q comme :

$$\begin{array}{c} i-k+1 \leq p \leq m \\ i-k+1 \leq q \leq p \end{array}$$

Ces bornes permettent donc de calculer les termes non nuls du tableau B_{impq} sans aucune redondance. Le stockage et l'accès aux données s'effectue par des tables d'indirections qui prennent en compte les symétries mais pas le support des fonctions de base. Il existe une table par indice : $offset_i$, $offset_m$, $offset_p$ et $offset_q$. Soumis à la relation $q \leq p \leq m \leq i$, on définit un ordre sur les indices : i est le premier indice, suivi de m, p et enfin q. A l'aide de cet ordre, on construit les tables en considérant pour chaque table d'indice i, m, p ou q les indices d'ordres inférieurs. Autrement dit, le décalage indiqué par la table $offset_i$ prend en compte toutes les données des indices plus petits pour toutes les dimensions.

$offset_q$:

L'indice q étant celui d'ordre minimum, il ne dépend que de lui seul et donc la table d'indirection est directe :

$$offset_q[k] = k$$

Autrement dit, pour i, m et p fixés, des indices q consécutifs indiquent des données consécutives dans le tableau mono-dimensionnel B_{impq}.

$offset_p$:

Vis à vis de l'ordre défini entre les quatre indices, p est d'ordre supérieur à q, ceci implique qu'il doit prendre en compte l'indice q dans le décalage. Pour un indice p fixé à k, on a $k+1$ possibilités pour l'indice q :

p	q
k	0
k	1
$\vdots$	$\vdots$
k	$k-1$
k	k

le décalage qui doit donc être effectué pour accéder à $p = k$ doit donc prendre en compte les données dont l'indice p est entre 0 et $k - 1$. On a donc

$$offset_p[k] = \sum_{a=1}^{k} a = \frac{k(k+1)}{2}$$

$offset_m$:

Pour un indice m fixé à k, les indices p et q peuvent varier entre 0 et k sachant que l'on a toujours la relation $q \leq p \leq m \leq i$. On a donc $k(k+1)/2$ possibilités (ce problème est équivalent au remplissage d'une matrice carré symétrique de dimension k). Pour accéder aux données d'indice m valant k, il faut prendre en considération les données d'indice m inférieur à k, ce qui procure la relation :

$$offset_m[k] = \sum_{j=0}^{k-1} \frac{(j+1)(j+2)}{2} = \sum_{j=1}^{k} \frac{j(j+1)}{2}$$

Or, il peut être noté la construction suivante :

$$\begin{array}{ccccccc} 1 & & & & & = & 1.2/2 \\ 1+ & 2 & & & & = & 2.3/2 \\ \ldots & \ldots & \ldots & \ldots & \ldots & = & \ldots \\ 1+ & 2+ & 3+ & \ldots & +k & = & k.(k+1)/2 \end{array}$$

qui fait apparaître, après l'addition des lignes ci-dessus, la formule :

$$\sum_{i=1}^{k} i(k-i+1) = \sum_{i=1}^{k} \frac{i(i+1)}{2}$$
$$\Leftrightarrow (k+1)\sum_{i=1}^{k} i - \sum_{i=1}^{k} i^2 = \sum_{i=1}^{k} \frac{i(i+1)}{2}$$

en utilisant la formule $\sum_{i=1}^{k} i^2 = k(k+1)(2k+1)/6$, on obtient :

$$(k+1)\sum_{i=1}^{k} i - \frac{k(k+1)(2k+1)}{6} = \sum_{i=1}^{k} \frac{i(i+1)}{2}$$
$$\Leftrightarrow \frac{k(k+1)^2}{2} - \frac{k(k+1)(2k+1)}{6} = \sum_{i=1}^{k} \frac{i(i+1)}{2}$$
$$\Leftrightarrow \frac{k^3+3k^2+2k}{6} = \sum_{i=1}^{k} \frac{i(i+1)}{2}$$

On en déduit donc le déplacement pour l'indice m :

$$offset_m[k] = \frac{k^3+3k^2+2k}{6}$$

$offset_i$:

Le décalage à apporter pour accéder aux données d'indice i valant k est le nombre total de données présentes aux indices inférieurs. Or, pour un indice i fixé à k, les indices m, p et q varient de 0 à k avec la relation d'ordre $q \leq p \leq m \leq i$. Donc, pour un m donné, le tableau

$offset_m$ nous donne le nombre de données présentes pour les indices inférieurs. Pour un indice i fixé à k, le nombre de données présentes aux indices i inférieurs est donc :

$$offset_i[k] = \sum_{i=0}^{k} offset_m[i]$$

Ainsi, on trouve :

$$offset_i[k] = \sum_{i=0}^{k} \frac{i^3 + 3i^2 + 2i}{6} = \sum_{i=0}^{k} \frac{i^3}{6} + \sum_{i=0}^{k} \frac{i^2}{2} + \sum_{i=0}^{k} \frac{i}{3}$$

Or, en utilisant la relation $\sum_{i=0}^{k} i^3 = \frac{k^2(k+1)^2}{4}$, on trouve finalement :

$$offset_i[k] = \frac{k^4 + 6k^3 + 11k^2 + 6k}{24}$$

L'accès à une donnée passe donc par ces quatre tables d'indirection de la manière suivante :

$$B_{impq}\left[offset_i[i] + offset_m[m] + offset_p[p] + offset_q[q]\right]$$

Lors d'une insertion, il est envisageable de décaler les données pour ne recalculer que les données qui ont été modifiées. Pour le moment, nous recalculons l'ensemble de la structure sachant que les tables d'indirection sont précalculées et inchangées au cours d'une insertion. Nous recalculons donc l'ensemble des données non nulles de cette table. En considérant la propriété de localité des B-splines, la complexité de ce recalcul est en $\mathcal{O}(n)$, puisqu'en faisant varier l'indice i de 0 à n (le nombre de points de contrôle), chaque indice m, p et q est borné par la propriété de localité et la relation d'ordre $q \leq p \leq m \leq i$. De plus, le calcul de l'intégrale se fait numériquement, par la méthode des trapèzes avec un pas fixe de 0.08.

4.2.1.2.3 Energie continue de flexion Cette énergie a été détaillée dans la section (2.3.3). Les termes d'énergie sont (cf. équation 2.24) :

$$-\frac{\partial E}{\partial q_i^\alpha}(t) = k_f \sum_{j=1}^{n} \int_0^1 b_j''(s) b_i''(s)\, ds \left(q_j^\alpha(t) - q_j^\alpha(0)\right)$$

Dans cette équation, les intégrales sont précalculées dans un tableau à deux dimensions. De plus, les éléments du tableau sont des intégrales de produit de fonctions de base dérivées deux fois. Ainsi, ces termes ont les mêmes propriétés que les termes de la matrice des masses généralisées : le même support (donc la même propriété de localité) et une symétrie. Donc, la mise à jour de ces données s'apparente à celle de la matrice M, à savoir un décalage des données dans les deux dimensions du tableau suivi d'un recalcul d'un nombre restreint de termes. Le nombre de termes effectivement recalculés est constant grâce à la propriété de localité, comme pour la matrice M. Le processus étant similaire à celui de la matrice des masses généralisées, nous ne nous étendons pas sur cette mise à jour.

4.2.2 Technique de suppression

La suppression d'un nœud et de son point de contrôle associé se déroule de façon inverse à l'insertion. Prenons le cas d'une B-spline cubique, on a vu que l'insertion se déroule suivant le schéma de calcul de l'équation (4.3). En observant ce schéma de calcul, on détermine aisément que la suppression du point $\mathbf{q_{i+1}}$ s'accompagne par les mises à jour des points de contrôle $\mathbf{q_i}$ et $\mathbf{q_{i+2}}$ en les remplaçant simplement par les points de contrôle $\mathbf{q_i^*}$ et $\mathbf{q_{i+1}^*}$:

$$\begin{cases} \mathbf{q_i^*} & = & \dfrac{\mathbf{q_i} - (1 - \beta_{i,i}^3)\mathbf{q_{i-1}^*}}{\beta_{i,i}^3} \\ \mathbf{q_{i+1}^*} & = & \dfrac{\mathbf{q_{i+2}} - \beta_{i+2,i+2}^3 \mathbf{q_{i+2}^*}}{1 - \beta_{i+2,i+2}^3} \end{cases} \tag{4.4}$$

4.2.2.1 Les structures de donnée

La mise à jour des structures de donnée est similaire à celle opérée pour le processus d'insertion :

Matrice des masses généralisées :

Les éléments de la matrice sont décalés pour combler l'espace laissé vide par le point supprimé. La matrice carré passe donc de la dimension $n+1$ à la dimension n. Ensuite, le processus demande un recalcul local des données. Cette étape de la mise à jour est locale puisqu'elle ne recalcule que les données affectées par la suppression. La complexité de cette mise à jour est donc en $\mathcal{O}(1)$.

Inverse et décomposition LU de la matrice M :

Ces deux étapes sont effectuées sans mise à jour mais par un recalcul complet des matrices.

Vecteur de gravité :

Le vecteur de gravité perd lui aussi une donnée, ce qui induit un décalage des données d'indice supérieur et un recalcul local des éléments affectés par la suppression. Cette mise à jour est en temps constant $\mathcal{O}(1)$.

Les points de contrôle :

La structure contenant les points de contrôle est bien évidemment modifiée puisqu'un point de contrôle est supprimé. Un décalage doit là aussi s'opérer, suivi d'une mise à jour suivant le schéma de calcul (4.4). Les points de contrôle affectés sont locaux et n'induisent donc qu'une complexité en $\mathcal{O}(1)$.

4.2.2.2 Energies de déformation

Les énergies de déformation sont affectées par la suppression d'un nœud de la même manière que lors d'une insertion. Les mises à jour des données (ressorts, tables...) s'effectuent donc sur un schéma similaire à l'insertion(cf. section (4.2.1.2)). Nous ne rentrons donc pas dans le détail des calculs.

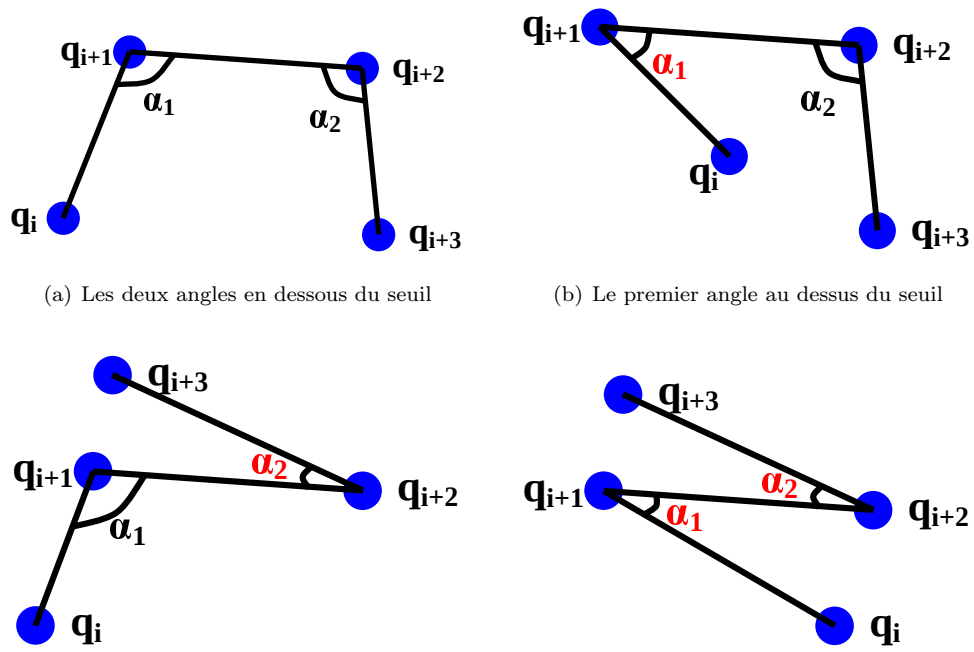
4.2.3 Simulation d'une spline multi-résolution

Si les techniques d'insertion et suppression de points de contrôle sont fixées, leur utilisation est liée à des critères d'insertion et suppression qui sont détaillés ci-après. De plus, l'utilisation de la multi-résolution n'est pas sans conséquence sur le modèle simulé. Il est donc important d'étudier la stabilité du modèle résultant afin de mieux appréhender ses avantages et ses inconvénients.

4.2.3.1 Critère d'insertion

L'insertion d'un point de contrôle permet une plus grande mobilité de la courbe dans une zone précise. Cela permet par exemple d'offrir à la courbe la possibilité de prendre une configuration très courbée qu'elle ne peut pas prendre dans sa résolution actuelle. C'est le cas par exemple lorsque l'on serre un nœud. La formation d'un nœud induit des courbures plus ou moins fortes qui vont s'accentuer pendant le serrage.

Au travers de cet exemple, on peut déterminer un critère géométrique lié à la courbure du modèle. Si la courbure locale de la courbe dépasse le seuil critique, on introduit un point de contrôle à cet endroit. Le fait d'introduire un point à cet endroit ne va pas réduire la courbure locale induisant donc une cascade d'insertions dans une même zone. Pour éviter cela, le critère d'insertion mesure une notion de courbure basée sur les points de contrôle locaux. Par exemple, pour une B-spline cubique, un segment paramétrique est déterminé par quatre points de contrôle. Le but de la manœuvre est donc de vérifier si un segment paramétrique a besoin d'être divisé ou non. Pour cela, on récupère les quatre points de contrôle et on teste les deux angles formés par les trois segments. S'ils sont tous les deux en dessous du seuil fixé, la configuration est considérée comme limite, on insère alors un point au centre du segment paramétrique. En pratique, on se contente d'approximer l'angle par son cosinus en calculant le produit scalaire des vecteurs normés. La figure (4.10) montre les quatre cas possibles pour un segment paramétrique d'une B-spline cubique.



(a) Les deux angles en dessous du seuil

(b) Le premier angle au dessus du seuil

(c) Le second angle au dessus du seuil

(d) Les deux angles au dessus du seuil

FIG. 4.10 – Schéma du critère d'insertion

Soit ϵ le seuil d'insertion, $\mathbf{q_i}$, $\mathbf{q_{i+1}}$, $\mathbf{q_{i+2}}$ et $\mathbf{q_{i+3}}$ les points de contrôle intervenant pour le calcul du i^{eme} segment spline. L'insertion a lieu au milieu du i^{eme} segment si les deux conditions suivantes sont vérifiées :

$$\frac{\mathbf{q_iq_{i+1}}}{|\mathbf{q_iq_{i+1}}|}.\frac{\mathbf{q_{i+1}q_{i+2}}}{|\mathbf{q_{i+1}q_{i+2}}|} < \epsilon \quad \text{et} \quad \frac{\mathbf{q_{i+1}q_{i+2}}}{|\mathbf{q_{i+1}q_{i+2}}|}.\frac{\mathbf{q_{i+2}q_{i+3}}}{|\mathbf{q_{i+2}q_{i+3}}|} < \epsilon$$

De ce fait, ϵ varie dans l'intervalle $[-1..1]$. Une valeur de 1 est un critère faible puisqu'il autorise toujours une insertion alors qu'une valeur de -1 impose un critère fort puisqu'il refuse toute insertion. En pratique, ϵ est négatif, proche de la valeur -0.4.

Ce critère permet d'augmenter la résolution de la courbe aux endroits où la courbure devient trop importante. Ainsi, les zones fortement courbées possèdent une résolution plus importante permettant de rendre plus précis le comportement local du modèle. On peut ainsi réaliser le serrage d'un nœud.

Le critère choisi est purement géométrique, il mesure des angles sur la configuration des points de contrôle. Il paraît légitime de chercher un critère basé sur les déformations du modèle, mais les énergies de déformation sont liées au modèle (élongation, courbure). Ainsi, tout critère basé sur la courbure (respectivement l'élongation) est indirectement lié à la déformation en flexion (respectivement en élongation).

4.2.3.2 Critère de suppression

La suppression d'un nœud dans le vecteur de nœuds d'une B-spline est soumis à des conditions mathématiques très strictes [LM87, Han87, Til92, EH95] puisqu'une fois le nœud supprimé, la courbe, amputée d'un point de contrôle, doit rester inchangée. Les conditions mathématiques étant difficilement vérifiables en temps réel, nous proposons de simplement opérer une suppression lorsque trois points de contrôle consécutifs sont alignés. En effet, en configuration rectiligne, une courbe B-spline peut se réduire au nombre minimum de points de contrôle tout en gardant la même configuration spatiale. Le but est donc de vérifier cette condition d'alignement à un niveau local, celui des points de contrôle. On vérifie donc sur chaque triplet de points de contrôle consécutifs $(\mathbf{q_i}, \mathbf{q_{i+1}}, \mathbf{q_{i+2}})$ de la courbe si leur alignement

vérifie la condition :

$$\frac{\mathbf{q_i q_{i+1}}}{|\mathbf{q_i q_{i+1}}|} \cdot \frac{\mathbf{q_{i+1} q_{i+2}}}{|\mathbf{q_{i+1} q_{i+2}}|} > \epsilon$$

où ϵ est ici le seuil de suppression. Comme celui d'insertion, il varie entre $[-1..1]$. Un seuil de -1 induit un critère de suppression lache puisqu'il autorise la suppression dans tous les cas de figure. Alors qu'un seuil à 1 refuse automatiquement la suppression. En pratique, trois points consécutifs alignés dans l'ordre donnent un produit scalaire égal à 1. Pour détecter les alignements ordonnés, il faut donc choisir un seuil proche de 1.

Etant donné que le critère théorique de suppression est rarement réalisé, la suppression induit généralement des modifications notables sur le comportement du modèle. Pour atténuer ces effets, il faut simplement se rapprocher le plus possible du critère de suppression théorique en choisissant un seuil de suppression plus proche de l'unité.

La suppression d'un point n'étant pas sans effet sur le comportement du modèle, il peut être intéressant de définir un critère qui prendrait en compte la stabilité du modèle. Si le modèle cherche sa position au repos, la suppression n'est pas autorisée alors que s'il commence à se stabiliser, on lui donne le droit de réduire sa résolution.

Il est également envisageable de mettre en place une transition douce de la suppression en interpolant deux simulations à des résolutions différentes à la manière de Perbet et al.[PC01]. Ceci revient à effectuer un *morphing*[23] entre deux résolutions différentes. Pendant quelques itérations de la simulation, le modèle est donc simulé à deux résolutions différentes et le modèle visualisé est le résultat d'une interpolation entre les deux résolutions. Cette technique aurait l'avantage d'adoucir les problèmes de saut comportementale dû à la suppression et donc permettrait de choisir un critère de suppression plus lâche. Cependant elle requiert la simulation du modèle à deux résolutions différentes et donc consomme beaucoup plus de temps. De plus, cette technique permet d'avoir un résultat visuel plus satisfaisant mais le modèle physique reste invalide au niveau physique.

4.2.3.3 Stabilité du modèle mécanique

Lorsque l'un des critères définis ci-dessus est réalisé, le modèle subit un changement de résolution qui induit des modification dans le comportement du modèle physique.

La stabilité du modèle multi-résolution est très étroitement liée au choix du modèle de déformation. Si ce modèle est discret, le ré-échantillonnage de la B-spline après une insertion ou une suppression induit des modifications qui se répercutent sur le comportement du modèle. Ces effets sont localisés dans le temps, ils apparaissent lors du changement de résolution. Ensuite, si la structure interne ne change plus pendant quelque pas de simulation, le modèle se stabilise. Si tout est continu (y compris les énergies de déformation) et surtout basé sur la forme, aucune discontinuité de comportement ne doit être observée lors d'une insertion car la forme ne se modifie pas.

Pour cela, il est plus intéressant de choisir les modèles de déformation continue pour utiliser la multi-résolution. Sachant que l'insertion d'un point est exacte, cette opération peut s'effectuer à tout moment et sur tout le modèle sans risquer de déstabiliser le modèle mécanique. Par contre, la suppression n'est, en général, pas un processus exact[LM87, EH95] et introduit donc une erreur. La courbe résultante après une suppression n'est plus la même. L'erreur commise est liée au seuil du critère de suppression. L'utilisateur peut donc minimiser cette erreur en définissant judicieusement la valeur du seuil.

4.2.3.4 Avantages et inconvénients

Le modèle multi-résolution proposé permet d'adapter le nombre de degrés de liberté du modèle pour lui permettre de simuler un comportement qu'il ne pourrait pas offrir sans un changement de résolution. Le principe est d'adapter les degrés de liberté, mais le modèle reste le même. L'adaptation du modèle est mathématiquement exact en insertion, procurant une stabilité physique du modèle dans le temps. Ainsi, le modèle mécanique est inchangé lors d'un changement de résolution d'un niveau grossier à un niveau

[23] Fondu, interpolation douce d'un état à l'autre.

plus fin. De plus, l'adaptation du modèle spline peut s'effectuer en tout point du modèle continu, offrant ainsi une grande liberté et une grande souplesse d'utilisation.

Un changement de résolution d'un niveau fin vers un niveau grossier n'est, quant à lui, pas mathématiquement exact puisque le critère proposé est une approximation du véritable critère. Ainsi, lors de la suppression de degrés de liberté, le modèle mécanique est modifié, relativement au seuil du critère de suppression.

4.3 Test : adaptation automatique de la résolution

Cette section expose différents résultats de simulation obtenus avec le modèle de spline dynamique multi-résolution.

L'un des intérêts de la multi-résolution sur le modèle de spline dynamique est d'offrir au modèle plus de degrés de liberté dans les zones courbées. Par exemple, en simulation chirurgicale, l'un des gestes de base est la manipulation d'un fil de chirurgie avec, en phase finale d'opération, la création d'un nœud. Lorsque le praticien crée le nœud, il forme des boucles qui impliquent des courbures plus ou moins prononcées sur la courbe. Puis, en serrant le nœud, les courbures s'accentuent. Sans multi-résolution, le modèle de spline arrive assez rapidement à sa configuration limite et ne peut pas prendre les courbures nécessaires au serrage du nœud. Lorsque l'on active l'insertion automatique de point, le modèle affine sa résolution dans les zones de forte courbure et donne la possibilité de serrer le nœud. Cependant, le nombre de points de contrôle augmente relativement au critère d'insertion. Si ce critère est lâche, le nombre de points de contrôle peut littéralement exploser alors que s'il est difficilement atteint, la simulation garde le contrôle du nombre de points de contrôle. Pour éviter une explosion du nombre de points de contrôle, on peut également activer la suppression automatique de point. Dans les zones de forte courbure, des points apparaissent alors que dans les zones rectilignes, des points sont supprimés. Ainsi, le nombre de points de contrôle peut s'équilibrer sur une simulation puisque l'action se déroule généralement sur une zone précise du modèle où la courbure va s'accentuer. Dans cette zone, la résolution risque d'augmenter alors qu'ailleurs, au contraire, elle risque de diminuer.

Le modèle utilisé pour cette simulation est une B-spline non uniforme cubique de 11 points de contrôle positionnée au départ pour préformer un nœud. La courbe pèse 10 g et possède un coefficient d'amortissement de 0.01. Afin que le modèle réagisse avec lui même, les auto-collisions sont activées et la réponse est calculée par pénalité avec des ressorts amortis de raideur 30 et de coefficient d'amortissement 0.1. L'énergie de déformation est continue et uniquement en élongation. Le module de Young utilisé est de 800000. Afin que le nœud puisse se former au cours de la simulation, on fixe les extrémités de la spline à l'aide de deux contraintes de points fixes. Puis, on laisse la courbe tomber sous son poids et serrer naturellement le nœud. Dans les simulations de serrage de nœud, la méthode d'intégration numérique employée est la méthode d'Euler implicite pour des raisons de robustesse et de fiabilité. Malheureusement, cette méthode n'assure pas un temps d'intégration constant, en conséquence de quoi les temps de calcul annoncés le sont à titre indicatif.

Sans multi-résolution, l'état d'équilibre est donné par la figure (4.11) et le temps de calcul d'une itération de la simulation est de 2.27 ms.

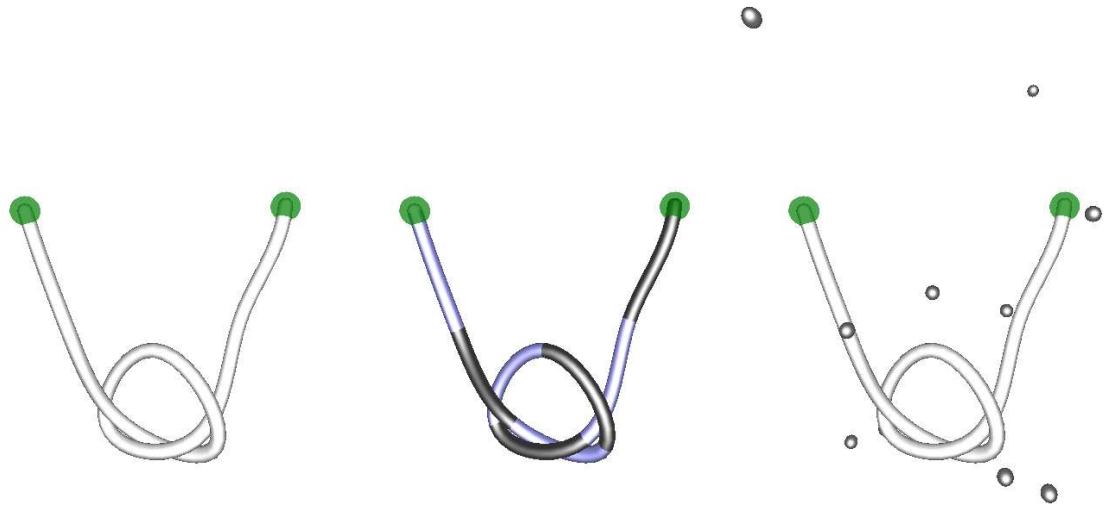
Avec l'insertion automatique et un seuil de -0.4, le temps de calcul d'une itération est de 12.01 ms lorsque la simulation est à l'équilibre. La spline possède 11 points de contrôle au départ et arrive à l'état stable avec 20 points de contrôle. L'état d'équilibre de la simulation est présenté sur la figure (4.13). Voici l'état du vecteur de nœud au départ :

$$s = \left\{ \begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \end{array} \right\}$$

et à l'état stable :

$$s = \left\{ \begin{array}{cccccccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 5.5 & 6 & 6.5 & 6.75 & 6.875 & 7 & 7.25 & 7.5 & 7.625 & 7.75 & 8 & 8.5 \\ 9 & 10 & 11 & 12 & 13 & 14 & \end{array} \right\}$$

Ces vecteurs de nœuds sont schématisés sur la figure (4.12).



(a) Courbe en rendu normal (b) Mise en évidence des segments paramétriques (c) Mise en évidence des points de contrôle

FIG. 4.11 – Serrage d'un nœud sans multirésolution

Etant donné que certaines structures de donnée sont entièrement recalculées lors d'une insertion, le temps de calcul d'une insertion augmente avec le nombre de points de contrôle. Au début, ce temps est de 0.014 ms (pour passer de 11 à 12 points de contrôle) et la dernière insertion, qui introduit le 20^{eme} point de contrôle, prend 0.020 ms. Afin de mieux percevoir la multi-résolution dans la zone de serrage du nœud, la figure (4.14) présente un zoom sur le nœud formé à l'état d'équilibre.

On remarque bien sûr les différentes figures que l'insertion de nombreux points de contrôle offre localement des courbures plus importantes et permet donc au nœud de se serrer.

Maintenant, si l'on active à la fois l'insertion et la suppression automatique, la B-spline affine sa résolution dans les zones de fortes courbures et au contraire la réduit dans les zones rectilignes. Le critère d'insertion est inchangé avec un seuil toujours à -0.4. Pour le critère de suppression, le seuil est fixé à 0.99975. Ce seuil est extrémement proche de 1 et montre la fragilité de l'algorithme inexact de suppression face à une simulation mécanique. Mise à part l'activation de la suppression automatique, le modèle mécanique conserve les mêmes propriétés.

La simulation commence donc avec 11 points de contrôle et trouve son état d'équilibre avec une B-spline de 16 points de contrôle. On trouve d'abord les étapes d'insertion puis, lorsque le modèle se stabilise, il prend une configuration plus ou moins étirée sous son propre poids, ce qui permet de passer en dessous du seuil de la suppression. Donc, la suppression a plutôt lieu à l'approche de l'état d'équilibre. Le temps de calcul d'une itération est, à l'état stable, de 9.89 ms. Les temps de calculs dus aux changements de résolution sont équivalents à ceux annoncés pour l'insertion. Cela s'explique par le fait que les mises à jour et les recalculs sont similaires dans les deux processus.

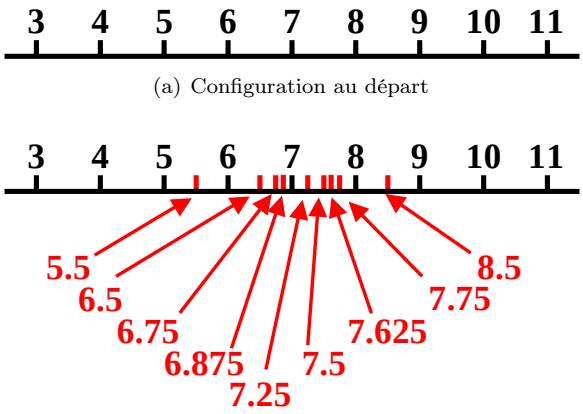
Voici l'état du vecteur de nœud au départ :

$$s = \left\{ \begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \end{array} \right\}$$

et à l'état d'équilibre :

$$s = \left\{ \begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 5 & 6 & 6.5 & 6.75 & 6.875 & 7 & 7.25 & 7.5 & 7.625 & 7.75 & 8 & 8.5 \\ 11 & 12 & 13 & 14 & \} & & & & & & & & & & & \end{array} \right.$$

Ces deux configurations sont schématisées sur la figure (4.15). Au total, la B-spline a subi 9 insertions (aux abscisses paramétriques 8.5, 5.5, 6.5, 7.5, 6.75, 7.75, 6.875, 7.25 et 7.625) suivies de 4 suppressions (aux abscisses paramétriques 4, 5.5, 9 et 10).



(a) Configuration au départ

(b) Configuration à l'équilibre

FIG. 4.12 – Evolution du vecteur de nœud lors d'une simulation multi-résolution (insertion uniquement) (la courbe est définie entre les abscisses paramétriques 3 et 11).

La figure (4.16) montre l'état d'équilibre de la simulation. On peut remarquer la forte concentration de points de contrôle au niveau du nœud et, au contraire, la faible concentration sur les morceaux rectilignes, aux abords des points fixes.

La figure (4.17) montre un zoom sur le nœud dans son état d'équilibre. En le comparant avec la figure (4.14) où la suppression automatique n'est pas activée, on remarque que les deux nœuds sont identiques avec des configurations paramétriques équivalentes et des points de contrôle localisés aux mêmes endroits. Ceci s'explique par le fait que le processus d'insertion entre naturellement en action lors du serrage du nœud alors que le processus de suppression attend que la courbe s'approche de sa position d'équilibre. Ainsi, le début de la simulation ressemble à celle où seule l'insertion est activée, laissant le nœud se former de la même façon dans les deux simulations. C'est seulement quand la simulation se stabilise que des points de contrôle sont supprimés. La courbe posséde cependant une configuration très courbée au niveau du nœud ne laissant aucune chance à l'algorithme de suppression de le modifier. C'est pourquoi les nœuds sont identiques dans les deux simulations.

La simulation d'une spline dynamique multi-résolution avec suppression montre des à-coups dans le comportement à chaque suppression d'un point de contrôle. Ceci rappelle simplement que la suppression doit être activée avec précaution et un choix judicieux pour le seuil doit être effectué.

4.4 Application de la multi-résolution à la découpe

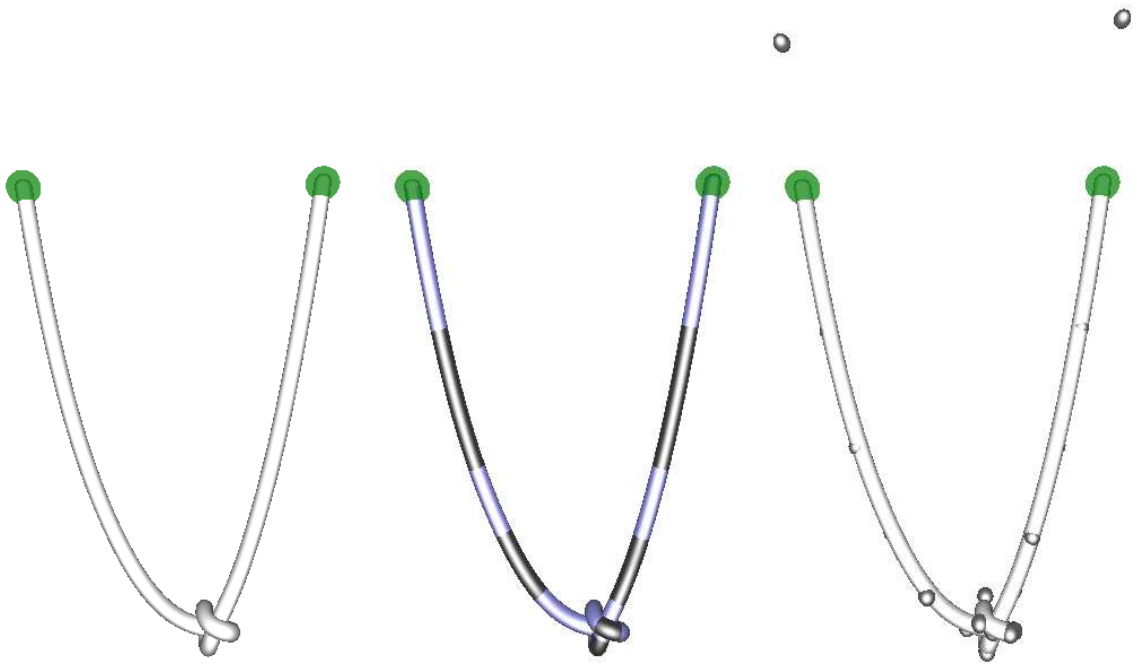
La technique de multi-résolution mécanique proposée peut être utilisée pour établir des découpes du modèle spline dynamique 1D. Mais, afin de mieux appréhender le domaine de la découpe d'objets simulés, nous commençons cette section par un survol des travaux existants sur le thème particulier des découpes en simulation physique.

4.4.1 Quelques travaux de simulation physique de découpes

Le problème des découpes en simulation physique peut être abordé de trois manières différentes comme l'annonce Boux-De-Casson[BdC00] :

Suppression d'élément :

Les éléments du maillage, présents sur la zone de découpe sont supprimés. Cette technique a l'inconvénient de ne pas conserver la masse de l'objet. En effet, chaque élément supprimé est une part de la matière de l'objet simulé et donc sa suppression entraîne une perte de matière. Cette technique fonctionne correctement pour des maillages fins permettant de retirer le minimum de matière lors



(a) Cylindre généralisé de couleur unie (b) Mise en évidence des segments paramétriques (c) Mise en évidence des points de contrôle

FIG. 4.13 – Serrage d'un nœud avec l'insertion automatique

d'une découpe [Pic01]. De plus, l'utilisation d'un maillage fin permet d'afficher le modèle découpé tout en conservant une précision dans le volume de l'objet alors qu'un maillage grossier ferait apparaître des artéfacts. Cependant, l'emploi d'un maillage fin n'est pas sans conséquence sur les temps de calculs de la simulation.

Découpe par modification topologique du maillage initial :

Lorsqu'une découpe est demandée, elle a lieu dans une zone bien délimitée qui intersecte des éléments du maillage.

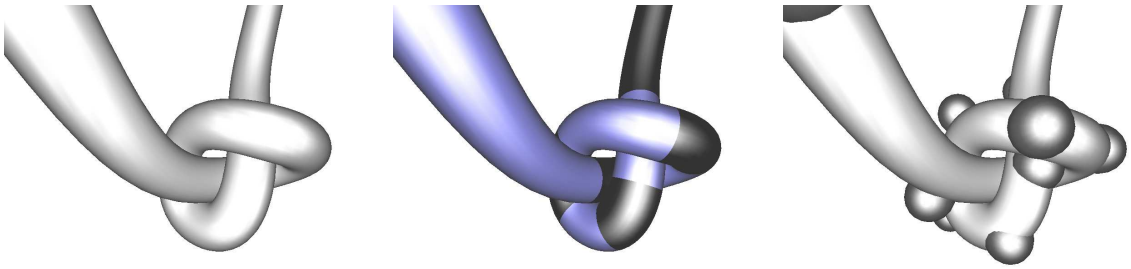
Cette technique modifie le maillage existant pour l'adapter à la découpe demandée. Ces modifications peuvent être de deux ordres : la duplication d'éléments (points, arrêtes...) pour permettre une séparation au niveau de la découpe et, au besoin, le déplacement de certains éléments pour les faire coïncider avec la zone de découpe.

Cette technique peut demander l'insertion de nouveaux éléments mécaniques (masses, ressorts, tétraèdres...) mais ils servent uniquement à séparer deux éléments (triangles, tétraèdres) voisins du modèle initiale. Ces nouveaux éléments ne permettent en aucun cas d'affiner la résolution de l'objet.

Découpe par raffinement :

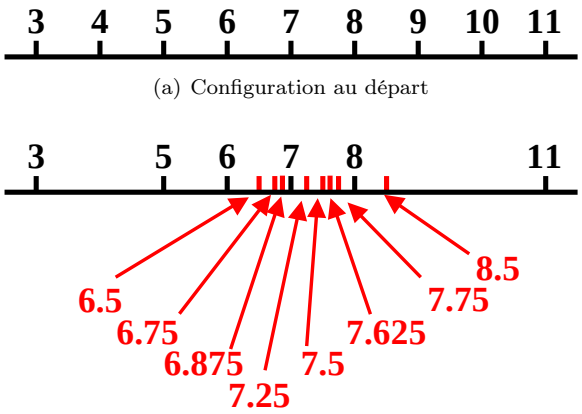
Une dernière technique permettant d'effectuer des découpes sur des modèles physiques est d'utiliser la multi-résolution pour définir des trous dans la topologie de l'objet ou encore d'adapter la résolution d'un maillage discret avant d'appliquer la découpe. Cette catégorie de méthodes est la plus aboutie puisqu'elle permet d'opérer une découpe qui allie la limitation des discontinuités de l'animation physique et une découpe effective aussi proche que souhaité de la zone de découpe.

La découpe peut être réalisée avec une approche par suppression de matière comme le propose par exemple Cotin et al.[Cot97, CDA00] ou encore Picinbono et al.[PLDA00]. Dans ces travaux, la découpe à lieu sur un modèle physique de type masses-tenseurs détaillés en section (1.1.2.2.6), or ce modèle physique offre des caractéristiques intéressantes dans le contexte des découpes.



(a) Cylindre généralisé de couleur unie (b) Mise en évidence des segments paramétriques (c) Mise en évidence des points de contrôle

FIG. 4.14 – Serrage d'un nœud avec l'insertion automatique (Zoom sur le nœud)



(a) Configuration au départ

(b) Configuration à l'équilibre

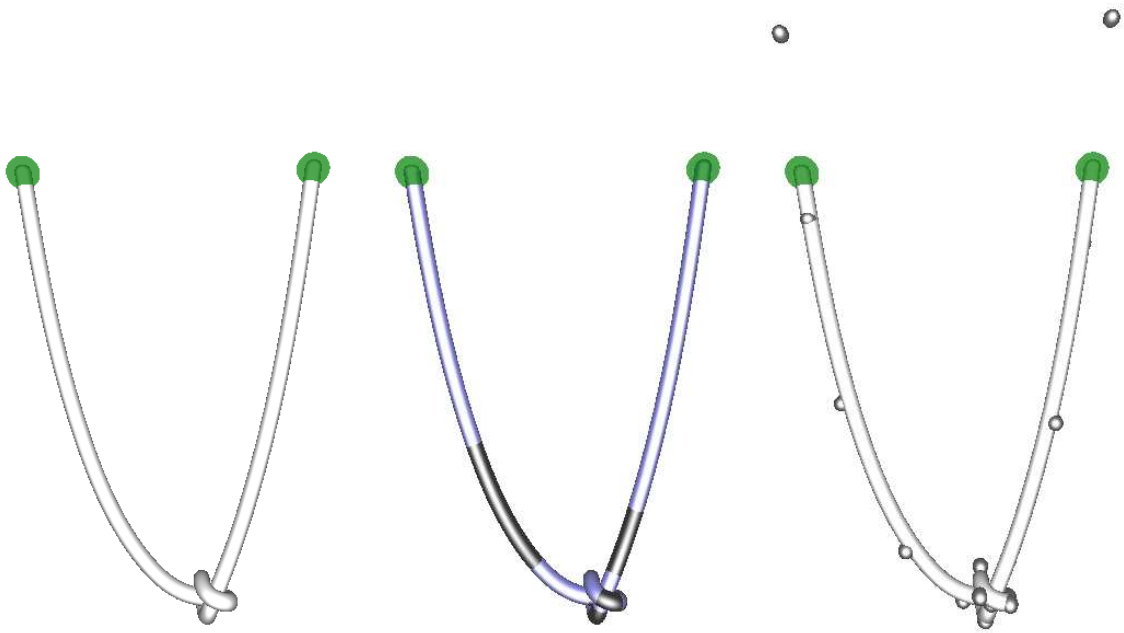
FIG. 4.15 – Evolution du vecteur de nœuds lors d'une simulation multirésolution (insertion et suppression) (la courbe est définie entre les abscisses paramétriques 3 et 11).

Cotin et al. [Cot97, CDA00] proposent d'insérer dans les équations des nœuds de surface une force de tension surfacique pour augmenter le réalisme de la simulation surtout lors d'une découpe. Les auteurs proposent de gérer la découpe par la technique de suppression d'éléments. Lorsqu'un utilisateur provoque une découpe, des tétraèdres du maillage sont purement et simplement supprimés et leurs contributions, au niveau des tenseurs du voisinage, sont soustraites. Les auteurs proposent ainsi de gérer la fracture de tissus mous (déchirement) comme un cas particulier de découpes engendrées, non plus par un utilisateur externe mais, par le modèle lui même au vu des contraintes internes.

Comme le modèle hybride (cf. section (1.1.2.2.7)) est un modèle dérivé du modèle masses-tenseurs, il hérite donc de cette propriété de découpe et de déchirement [CDA00]. Une découpe d'un modèle hybride est présentée sur la figure (4.18). Le schéma en fils de fer montre que la partie supérieure de l'organe est simulée par le modèle quasi-statique éléments finis alors que la partie inférieure de l'organe est simulée à l'aide des masses-tenseurs. Ainsi, les découpes et déchirements ne peuvent subvenir que dans la zone inférieure de l'organe. Cette méthode de découpe est cependant difficile à classer puisqu'elle limite la zone de découpe à l'endroit simulé par les masses-tenseurs. Cette zone étant prédéfinie, cette technique de découpe fait donc également partie intégrante des techniques de pré-découpe.

La découpe par suppression d'éléments d'un maillage masses-tenseurs est reprise par Picinbono et al.[PLDA00].

Une autre approche possible pour les découpes est celle de la modification topologique du maillage. On trouve dans la littérature plusieurs propositions relatives à cette approche, comme par exemple les tra-



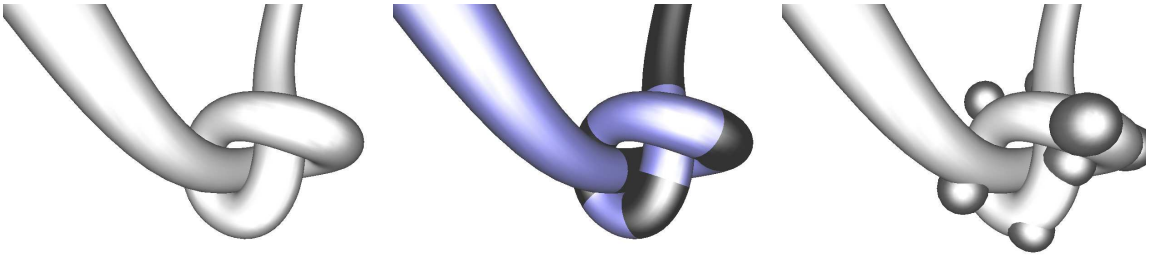
(a) Cylindre généralisé de couleur unie (b) Mise en évidence des segments paramétriques (c) Mise en évidence des points de contrôle

FIG. 4.16 – Serrage d'un nœud avec l'insertion et la suppression automatique

vaux de Nienhuys et van der Stappen[NvdS00, NvdS01], Boux-De-Casson et Laugier[BdCL00, MLBdC01] ou encore Mendoza et Laugier[ML03].

Nienhuys et van der Stappen[NvdS00, NvdS01] proposent une stratégie de découpe basée sur un modèle déformable éléments finis. L'objet est décomposé par un échantillonnage en tétraèdres qui sont la base du modèle éléments finis statique. Aucun précalcul n'est défini, ce qui permet de modifier la structure topologique en cours de simulation sans alourdir la simulation par d'éventuels recalculs. Les auteurs imposent alors que les découpes ne se produisent qu'aux arrêtes des tétraèdres. Ils proposent un algorithme de sélection des arrêtes définissant la zone de découpe, puis la découpe a lieu en modifiant les propriétés topologiques du maillage éléments finis.

Boux-De-Casson et Laugier[BdCL00, MLBdC01] proposent un modèle de déchirure en 2D. Le tissu est simulé physiquement à l'aide d'un modèle masses-ressorts muni de ressorts visco-élastiques. Le déchirement est simulé en séparant les facettes limitrophes. Autrement dit, des points du maillage sont



(a) Cylindre généralisé de couleur unie (b) Mise en évidence des segments paramétriques (c) Mise en évidence des points de contrôle

FIG. 4.17 – Serrage d'un nœud avec l'insertion et la suppression automatique (zoom sur le nœud)

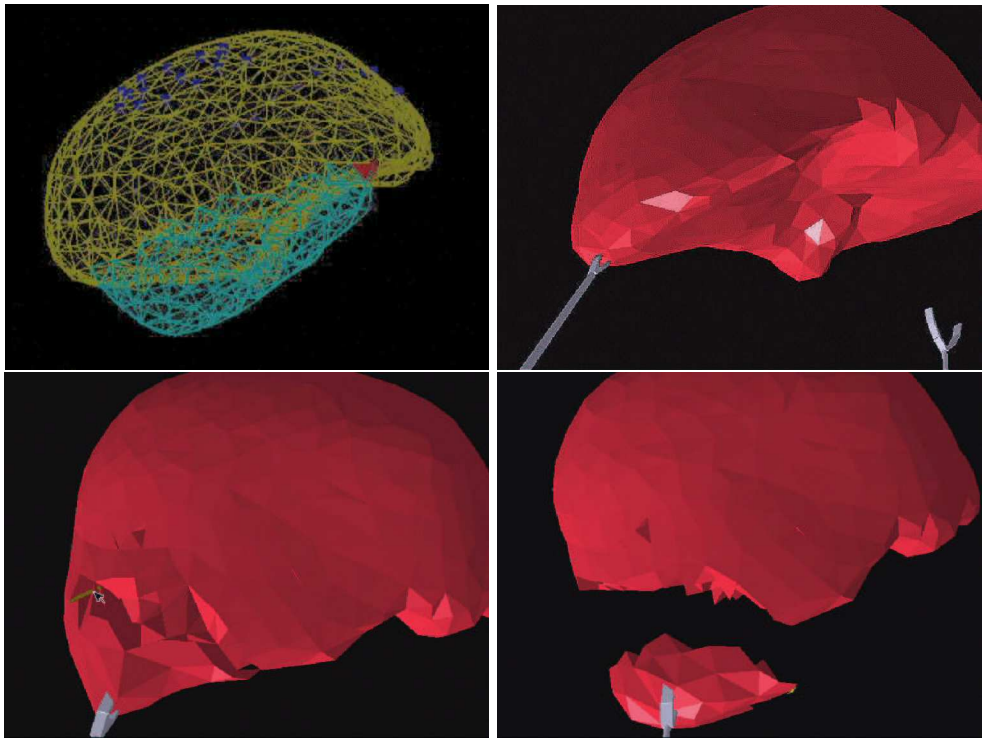


FIG. 4.18 – Découpe d'un modèle hybride par suppression de tétraèdres (Extrait de [CDA00])

dupliqués ainsi que des ressorts aux endroits où la déchirure à lieu. Afin de conserver une simulation physique stable, la configuration au repos, des ressorts mis en cause dans la déchirure, est mise à jour.

Cette proposition est étendue au cas 3D par Mendoza et Laugier[ML03] qui proposent de simuler un objet à l'aide d'un modèle élément fini basé sur une décomposition en tétraèdres. Lorsqu'une découpe intervient, elle est définie par un chemin sur la surface de l'objet. Les tétraèdres présents sur ce chemin sont simplement dispersés d'un côté ou de l'autre du chemin en déplaçant les nœuds des tétraèdres. Les propriétés physiques de ces tétraèdres sont également mis à jour pour rester cohérents avec la simulation de découpe.

Enfin, la dernière méthode permettant de mettre en œuvre des découpes est la méthode par raffinement. Là encore, il existe de nombreux travaux sur cette technique de découpe, comme par exemple les propositions de Shi et Yan[SY01], Bielser et Gross[BMG99, BG00], Vigneron et al.[VVW04] ou encore Picinbono[Pic01].

Shi et Yan[SY01] proposent d'adapter la topologie d'un maillage tétraèdrique pour que le chemin de découpe coïncide avec des arrêtes de tétraèdres. Pour cela, les tétraèdres présents sur le chemin de découpe sont scindés en plusieurs tétraèdres. Ainsi, le maillage est localement subdivisé au niveau de la zone de découpe.

Bielser et Gross[BMG99, BG00] proposent un cadre de simulation physique pour la découpe de modèles déformables. Leur proposition de découpe se base sur un algorithme de subdivision du maillage tétraèdrique. Lorsqu'un tétraèdre est découpé, cinq cas sont possibles pour découper l'élément, suivant la configuration de la découpe. Autrement dit, une zone de découpe affecte une partie du tétraèdre (arrête, face...) ou le découpe en deux parties distinctes. Cette technique itérative de subdivision des tétraèdres densifie localement le maillage et ne permet pas de l'utiliser en temps réel. Ganovelli et al.[GCMS00] reprennent cette idée d'un maillage tétraèdrique subdivisé pour simuler une découpe en proposant un algorithme de décomposition des tétraèdres plus efficace. Ainsi, ils obtiennent un maillage subdivisé sur la zone de découpe tout en limitant le nombre de tétraèdres générés.

Vigneron et al.[VVW04] proposent une technique permettant de changer la topologie d'un maillage élément fini en utilisant les éléments finis étendu. Cette technique introduit, au endroit souhaité du maillage, des fonctions supplémentaires qui viennent perturber les fonctions d'interpolations des FEM. En choisissant des fonctions discontinues, il est possible de simuler des craquelure dans le maillage et donc des découpes. Cette technique s'appuie sur le modèle continu de l'objet simulé et permet de répercuter l'effet de la découpe sur le maillage discret. Il est à noter que Vigneron montre l'application de cette technique au cas d'un maillage 2D. Son utilisation en 3D requiert la mise en place de nouvelles fonctions de perturbations.

Picinbono propose dans sa thèse[Pic01] une amélioration du modèle de découpe par suppression d'éléments en le substituant par un modèle de découpe du maillage. Il obtient ainsi une subdivision de son maillage tétraèdrique au niveau de la zone de découpe.

Nous proposons de découper notre modèle de spline dynamique 1D à l'aide d'une méthode par raffinement. Cette méthode permet de conserver la masse de l'objet, d'approcher d'aussi près que l'on désire la zone de découpe et enfin elle ne perturbe pas la simulation physique du modèle.

4.4.2 Découpe de notre modèle de spline

Une conséquence de la multi-résolution du modèle de spline dynamique est la possibilité de découper une courbe B-spline. En effet, l'insertion d'un point de contrôle est localisée sur le vecteur de nœuds. Si le nœud inséré est de multiplicité 1, la B-spline est inchangée, elle conserve sa continuité sur toute sa longueur. Si la multiplicité du nœud est strictement supérieure à 1, le modèle perd localement de sa continuité. Une B-spline cubique est normallement de continuité C^2. Si le nœud inséré possède une multiplicité de 2, la continuité diminue localement à 1. Pour une multiplicité de 3, la continuité est nulle. Et, si l'on insére à nouveau un point au même endroit sur le vecteur de nœuds, la multiplicité de ce nœud passe à 4, la continuité de la courbe devient C^{-1} à cet endroit. Autrement dit, la B-spline simulée est composée de deux morceaux de B-spline géométriquement et physiquement indépendants.

La découpe du modèle spline dynamique multi-résolution peut donc simplement être réalisée en insérant à l'endroit désiré assez de points de contrôle pour réduire la continuité à C^{-1}.

Pour mettre en évidence la découpe, une B-spline non uniforme cubique de 19 points de contrôle est simulée. Cette spline est contrainte par des points fixes à ses extrémités et en son milieu. Les propriétés physiques du modèle sont sa masse de 10 g, son coefficient d'amortissement de 0.005, son énergie continue d'élongation avec un module de Young de 5000 pour un rayon de 6 mm. La configuration au départ de la spline est rectiligne alignée sur l'axe des x avec une longueur de 44.4 cm.

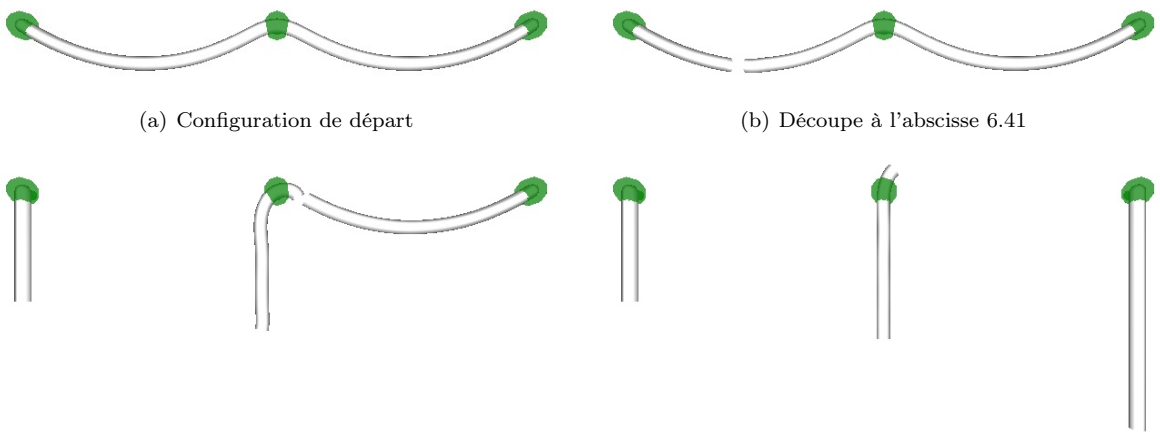
La sous-figure (a) de la figure (4.19) montre l'état d'équilibre de la simulation sans aucune découpe. Le temps de calcul de cette simulation est de 2 ms par itération.

La sous-figure (b) de la figure (4.19) montre l'état de la simulation après une première découpe occasionnée manuellement au niveau de l'abscisse paramétrique 6.41. Le nombre de points de contrôle augmente donc de 19 à 23 et le temps de calcul d'une itération de la simulation après cette découpe est de 4.5 ms.

La sous-figure (c) de la figure (4.19) montre l'état de la simulation après une seconde découpe occasionnée manuellement au niveau de l'abscisse paramétrique 11.75. Le nombre de points de contrôle augmente donc de 23 à 27 et le temps de calcul d'une itération de la simulation après cette seconde découpe est de 7.85 ms.

La sous-figure (d) de la figure (4.19) montre l'état d'équilibre de la simulation après les deux découpes. On peut aisément observer les trois morceaux de courbe qui appartiennent pourtant à la même spline dynamique.

Afin de mettre en évidence l'indépendance des morceaux d'une même spline dynamique, à la fin de cette simulation, l'utilisateur manipule une sonde qui entre en interaction avec le modèle. La figure (4.20) montre un état de la simulation où la sonde est en interaction avec l'un des morceaux de la spline découpée. On remarque que les deux autres morceaux de splines ne sont absolument pas affectés par cette interaction.



(a) Configuration de départ (b) Découpe à l'abscisse 6.41

(c) Découpe à l'abscisse 11.75 (d) Etat d'équilibre de la simulation après les deux découpes

FIG. 4.19 – Découpe d'une B-spline non uniforme cubique

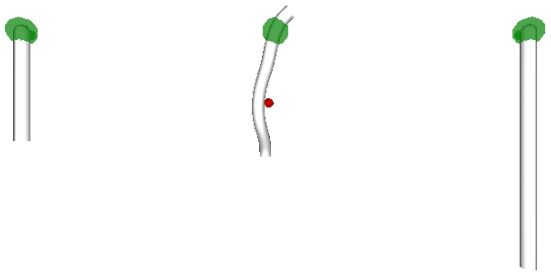


FIG. 4.20 – Indépendance des morceaux d'une même B-spline

4.5 Conclusion

Ce chapitre a permis d'exposer notre proposition de multi-résolution pour le modèle de spline dynamique. Elle permet d'adapter le nombre de degrés de liberté du modèle physique sans pour autant changer le modèle géométrique ou physique. L'avantage premier de notre proposition est d'offrir la possibilité d'affiner la résolution du modèle continu en tout endroit. De plus, ce changement de résolution s'effectue sans discontinuité temporelle dans le sens d'un raffinage du modèle et avec une certaine discontinuité dans l'autre sens. Un test d'adaptation automatique de la résolution du modèle permet de montrer son intérêt, notamment pour le serrage de nœud. Enfin, la section précédente a permis de montrer que notre modèle multi-résolution permet de simuler des découpes sur un modèle spline dynamique 1D.

L'algorithme d'insertion est mathématiquement exact, ce qui n'est pas le cas de l'algorithme de suppression. Le processus de suppression est, en théorie, soumis à des conditions d'exécution très strictes. Une des perspectives de ce modèle multi-résolution réside donc dans l'établissement de nouveaux critères de subdivision. Pour l'insertion, il peut être intéressant de limiter des insertions trop abusives dans une même zone ou, d'intégrer l'interaction de l'utilisateur dans le critère de subdivision. Pour la suppression, il peut être intéressant d'affiner le critère pour se rapprocher des conditions théoriques de la suppression. On peut également imaginer un critère de suppression basé sur la stabilité du modèle. En effet, si le modèle en action passe par un état rectiligne, l'algorithme supprime alors quelques points qui pourraient être nécessaires dans l'évolution des itérations suivantes. Un critère de stabilité permettrait d'activer la suppression uniquement lorsque le modèle a trouvé son état d'équilibre.

Au niveau des temps de calculs, la subdivision demande un temps, pour les recalculs et les mises à jour, relativement faible par rapport au temps de calcul de la simulation (entre 1% et 2%). La plus grande perte de temps se situe au niveau de l'évaluation, à chaque itération, de l'énergie continue d'élongation.

Enfin, l'extension de ces travaux aux dimensions supérieures est envisageable. Cependant, une telle multi-résolution sur une spline 2D ou 3D perd toute sa souplesse puisque l'insertion d'un nœud sur une dimension s'accompagne de l'ajout d'un ensemble de points de contrôle sur les autres dimensions. En 2D, on obtient donc une résolution plus fine sur un chemin défini par un paramètre s_1 ou s_2 de valeur constante. Cette adaptation n'est donc plus entièrement locale mais globale sur une dimension. Le problème s'étend sur deux dimensions pour le cas des splines 3D. Il n'est donc pas très intéressant d'appliquer une telle multi-résolution sur les splines de dimension supérieure ou égale à deux.

Conclusion

Ce travail de thèse a permis de mettre en place, dans un cadre de simulation physique temps réel, un modèle déformable 1D multi-résolution. Le modèle de base repose sur la mécanique de Lagrange et s'est appuyé sur les travaux de Yannick Rémion et son équipe du LERI. Il permet de simuler tout objet déformable à squelette curviligne. Ainsi, l'objet est doté d'un volume au niveau de l'affichage mais son comportement est réduit au squelette. La visualisation du modèle s'effectue au travers d'un vaste choix de méthodes d'habillages, comme par exemple les surfaces implicites ou les cylindres généralisés. Quant aux déformations du modèle, elles sont dictées par des énergies d'élongations et de flexions aussi bien discrètes que continues. Cet ensemble d'énergies permet de couvrir une large gamme de comportements rendant l'utilisation du modèle possible dans de nombreux domaines. Afin de rendre plus souple l'usage du modèle, nous proposons l'introduction de contraintes. Celle-ci est rendue possible par l'emploi de la méthode des multiplicateurs de Lagrange. Dans ce contexte, l'un des apports les plus importants de cette thèse concerne la définition d'une nouvelle classe de contraintes dites *contraintes glissantes* permettant de définir des conditions relatives à un point mobile du modèle 1D. Ce type de contraintes apporte une souplesse supplémentaire au modèle, par le fait que les conditions définies se déplace le long du modèle suivant la dynamique de celui-ci. De plus, l'un des apports majeurs de cette thèse est une architecture logicielle permettant de simuler des articulations d'objets dynamiques quelconques. Une autre technique permettant d'étendre les possibilités du modèle réside dans l'aspect multi-résolution. Cette proposition passe par la définition de critères qui permettent de déterminer les endroits du modèle nécessitant un changement de résolution. Des techniques de subdivisions introduisent, au niveau géométrique et mécanique, une adaptation locale du modèle. Une conséquence importante et originale de la multi-résolution est la découpe du modèle déformable 1D.

En terme de résultats, le modèle proposé a permis de mettre en place de nombreuses applications. Par exemple, dans le domaine de la simulation chirurgicale, le modèle de base a permis de simuler un intestin grêle. De plus, le modèle contraint permet, au travers des contraintes glissantes, de simuler une opération de suture. De même, la proposition d'architecture dédiée aux articulations d'objets offre la possibilité de mettre en place la pré-découpe d'un modèle 1D. Cette application a été mise en place pour une opération de salpingectomie qui consiste en la découpe d'une trompe de Fallope. Enfin, le modèle multi-résolution offre la possibilité de serrer un nœud notamment pour la simulation de fil chirurgical. Les travaux proposés dans cette thèse trouvent également des applications en modélisation variationnelle par l'utilisation des points glissants pour aider le concepteur dans sa démarche de modélisation. Dans ce contexte, des exemples de modélisation sont proposés tel un lacet de chaussure ou un nœud coulant. Dans un contexte lié aux techniques d'interactions, le modèle articulé permet de mettre en place l'une de ces techniques, celle dénommée *virtual proxy*. Cette technique permet à l'utilisateur d'interagir à l'aide d'un périphérique sur les objets virtuels sans créer d'incohérences dans la simulation. L'ensemble des résultats présentés illustrent l'utilisation du modèle en temps réel (voire temps interactif pour certaines applications) et les diverses applications du modèle démontrent sa grande souplesse d'utilisation (intestins, trompes de Fallope, fil de chirurgie, lacet de chaussure, cordes de balançoire...). Cependant, les travaux développés ne permettent pas encore d'utiliser le modèle dans un simulateur commercial de chirurgie. Le modèle proposé n'est pas encore suffisament robuste pour une application réelle et une étape d'intégration des méthodes proposées est encore nécessaire. Ce modèle peut être étendu de diverses façons, chacune d'elles apportant un peu plus de réalisme physique, de robustesse ou de souplesse.

Une première extension de ce travail consiste à appliquer le modèle aux dimensions supérieures. Nous avons déjà montré que les propositions sont applicables en dimension 2 et 3 moyennant des calculs plus importants. Seule la multi-résolution n'a pas été étudiée dans les dimensions supérieures pour la bonne raison que le produit tensoriel fourni une définition moins souple de la multi-résolution. Il faudrait plutôt utiliser des algorithmes de subdivision locaux comme l'algorithme de Loop, de Catmull-Clark ou les T-NURCCS[SZBN03]. Cependant, l'utilisation d'algorithmes de subdivision ne permet pas toujours de s'appuyer sur l'existence d'un modèle continu. De ce fait, la subdivision du modèle discret modifie celui-ci puisqu'il n'est pas basé sur une forme continue (telle qu'une spline). Ceci souligne l'un des intérêts de notre modèle multi-résolution, à savoir qu'il est basé sur un modèle géométrique continu permettant de choisir la résolution à laquelle nous souhaitons le simuler.

Une deuxième extension du modèle concerne les déformations de l'objet simulé. Notre modèle est

muni d'énergies de déformations en élongation et en flexion, mais aucune énergie de torsion n'est définie actuellement. Il pourrait être intéressant d'étudier l'intégration d'une telle énergie au sein de notre modèle, notamment au travers des travaux de Pai[Pai02] (cf. section (2.7)). Pour finaliser la prise en compte des déformations, l'énergie continue de flexion peut également être améliorée pour être totalement décorélée de l'énergie d'élongation.

Concernant l'aspect multi-résolution du modèle, les critères de changement de résolution pourraient être affinés pour améliorer l'utilisation de la multi-résolution. Nous pouvons imaginer par exemple un critère basé sur les zones d'interactions ou sur la stabilité du modèle. Etant donné l'incertitude numérique des simulations dynamiques (à cause du recours à une méthode d'intégration numérique qui peut diverger...), un critère pourrait être mixé avec la relative stabilité du modèle.

Ensuite, l'un des apports majeurs de ce travail de thèse réside dans la proposition de la nouvelle classe de contraintes appelées *contraintes glissantes*. Cette thèse a permis d'exploiter quelques unes de ces contraintes, mais cette nouvelle classe renferme des possibilités encore inexploitées. Nous pouvons citer par exemple l'utilisation d'une contrainte de point glissant sur une spline de dimension deux, où la contrainte serait elle-même contrainte sur un chemin prédéfini. De cette manière, un point sur un tissu déformable serait contraint de se déplacer suivant un chemin donné.

En outre, une autre proposition du modèle concerne les auto-collisions (pénalité), leur gestion est prise en charge par une méthode qui ne permet pas d'assurer leur intégrité. Par exemple, le serrage d'un nœud demande une gestion plus fine des auto-collisions. Il serait intéressant d'étudier l'utilisation de techniques plus robuste comme les multiplicateurs de Lagrange pour la gestion des nœuds.

Enfin, la méthode de gestion des contraintes employée (stabilisation à l'aide d'un schéma de Baumgarte) ne permet pas toujours d'assurer leur réalisation. Une méthode post-stabilisation pourrait être étudiée dans notre contexte. Il faudrait cependant généraliser cette prise en charge des contraintes au cas des contraintes glissantes que nous avons proposées.

Bibliographie

[ACR94] Uri M. Ascher, Hong Sheng Chin et Sebastian Reich. Stabilization of DAEs and invariant manifolds. *Numerische Mathematik*, 67(2) :131–149, 1994.

[Asc97] Uri M. Ascher. Stabilization of invariants of discretized differential systems. *Numerical Algorithms*, 14(1-3) :1–24, 1997.

[AT00] Amaury Aubel et Daniel Thalmann. Realistic deformation of human body shapes. In *Computer Animation and Simulation*, pages 125–135, Interlaken (Switzerland), 21-22 août 2000.

[Bar81] Brian A. Barsky. *The Beta-spline : A Local Representation Based on Shape Parameters and Fundamental Geometric Measures*. PhD thesis, Department of Computer Science, University of Utah, Salt Lake City, Utah (USA), 1981.

[Bar94] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Computer Graphics Proceedings*, Annual Conference Series, pages 23–34, Orlando, Florida (USA), 24-29 juillet 1994.

[Bar96] David Baraff. Linear-time dynamics using lagrange multipliers. *Computer Graphics (Proceedings of SIGGRAPH)*, pages 137–146, 4-9 août 1996.

[Bar97] Ronen Barzel. Faking dynamics of ropes and springs. *IEEE Computer Graphics and Applications*, 17(3) :31–39, mai-june 1997.

[Bau72] J.W. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 1(1) :1–16, juin 1972.

[BB88] Ronen Barzel et Alan H. Barr. A modeling system based on dynamic constraints. *Computer Graphics (Proceedings of SIGGRAPH)*, 22(4) :179–188, août 1988.

[BC00] David Bourguignon et Marie-Paule Cani. Controlling anisotropy in mass-spring systems. In *Proceedings of the 11th Eurographics Workshop, Computer Animation and Simulation '00*, pages 113–123, Interlaken (Switzerland), 21-22 août 2000.

[BdC00] François Boux de Casson. *Simulation dynamique de corps biologiques et changements de topologie interactifs*. PhD thesis, Université de Savoie, Chambéry (France), décembre 2000.

[BdCL00] François Boux de Casson et Christian Laugier. Simulating 2D tearing phenomena for interactive medical surgery simulators. In *Proc. of Computer Animation*, Philadelphia, Pennsylvania (USA), mai 2000.

[BG00] Daniel Bielser et Markus H. Gross. Interactive simulation of surgical cuts. In IEEE Computer Society Press, editor, *Pacific Conference on Computer Graphics and Applications (Pacific Graphics)*, pages 116–125, Hong Kong (China), 3-5 octobre 2000.

[BHG91] David E. Breen, Donald H. House et Phillip H. Getto. A particle-based computational model of cloth draping behavior. In *Scientific Visualization of Physical Phenomena (Proceedings of CGI '91)*, pages 113–134, Cambridge, MA (USA), juin 1991.

[BHG92] David E. Breen, Donald H. House et Phillip H. Getto. A physically-based particle model of woven cloth. *The Visual Computer*, 8(5-6) :264–277, juin 1992.

[BHW94a] David E. Breen, Donald H. House et Michael J. Wozny. A particle-based model for simulating the draping behavior of woven cloth. *Textile Research Journal*, 64(11) :663–685, novembre 1994.

[BHW94b] David E. Breen, Donald H. House et Michael J. Wozny. Predicting the drape of woven cloth using interacting particles. In *Proceedings of ACM SIGGRAPH*, volume 28, pages 365–372, Orlando, Florida (USA), juillet 1994.

[Bin87] Thomas O. Binford. Generalized cylinder representation. In Shapiro S.C., editor, *Encyclopedia of Artificial Intelligence*, pages 321–323. John Wiley & Sons, 1987.

[BKCN03] Florence Bertails, Tae-Yong Kim, Marie-Paule Cani et Ulrich Neumann. Adaptive wisp tree - a multiresolution control structure for simulating dynamic clustering in hair motion. *Symposium on Computer Animation*, juillet 2003.

[BKS00] Stephan Bischoff, Leif P. Kobbelt et Hans-Peter Seidel. Towards hardware implementation of loop subdivision. In *SIGGRAPH/EUROGRAPHICS Workshop On Graphics Hardware*, pages 41–50, Interlaken (Switzerland), 21-22 août 2000.

[Bla94] Carole Blanc. *Techniques de modélisation et de déformation de surfaces pour la synthèse d'images*. PhD thesis, LaBRI, Université de Bordeaux 1, décembre 1994.

[Bli82] Jim F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions On Graphics*, 1(3) :235–256, juillet 1982.

[Blo90] Jules Bloomenthal. *Graphics Gems*, volume 1, chapter Calculation of Reference Frames along a Space Curve, pages 567–571. Academic Press, 1990.

[Blo95] Jules Bloomenthal. Bulge elimination in implicit surface blends. In *Implicit Surfaces*, pages 7–20, Grenoble (France), avril 1995.

[Blo02] Jules Bloomenthal. Medial-based vertex deformation. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 147–152, San Antonio, Texas (USA), 21-22 juillet 2002.

[BMG99] Daniel Bielser, Volker A. Maiwald et Markus H. Gross. Interactive cuts through 3-dimensional soft tissue. *Computer Graphics Forum (Proceedings of Eurographics)*, 18(3), 7-11 septembre 1999.

[BNC96] Morten Bro-Nielsen et Stéphane Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In Jarek Rossignac et François X. Sillion, editors, *Computer Graphics Forum (Eurographics'96)*, volume 15 of *3*, pages 57–66, Futuroscope, Poitiers (France), septembre 1996. Blackwell Publishers.

[BO04] Gareth Bradshaw et Carol O'Sullivan. Adaptive medial-axis approximation for sphere-tree construction. *ACM Transactions on Graphics*, 23(1) :1–26, janvier 2004.

[Boe80] Wolfgang Boehm. Inserting new knots into b-spline curves. *Computer Aided Design*, 12(4) :199–201, juillet 1980.

[BS91] Jules Bloomenthal et Ken Shoemake. Convolution surfaces. *Computer Graphics (Proceedings of SIGGRAPH 91)*, 25(4) :251–256, 28 juillet - 2 août 1991.

[BS95] Carole Blanc et Christophe Schlick. X-splines : A spline model designed for the end-user. *Computer Graphics (Proceedings of SIGGRAPH'95)*, pages 377–386, 1995.

[BW97] David Baraff et Andrew Witkin. Partitioned dynamics. Technical Report CMU-RI-TR-97-33, Robotics Institute, Carnegie Mellon University, 1997.

[BW98] David Baraff et Andrew Witkin. Large steps in cloth simulation. In *Computer Graphics Proceedings, Annual Conference Series*, pages 43–54, Orlando, Florida (USA), 19-24 juillet 1998.

[BWK03] David Baraff, Andrew Witkin et Michael Kass. Untangling cloth. In *Proceedings of ACM SIGGRAPH (ACM Transactions on Graphics)*, volume 22 of *3*, pages 862–870, San Diego, California (USA), 27-31 juillet 2003.

[Béz66a] Pierre Bézier. Définition numérique de courbes et surfaces (partie 1). *Automatisme*, 11 :625–632, 1966.

[Béz66b] Pierre Bézier. Définition numérique de courbes et surfaces (partie 2). *Automatisme*, 12 :17–21, 1966.

[Béz77] Pierre Bézier. *Essai de définition numérique des courbes et surfaces expérimentales*. PhD thesis, Université de Paris VI, 1977.

[Béz86] Pierre Bézier. *Courbes et Surfaces*. Hermès france, 1986.

[CA99] Coello Coello Carlos Artemio. A survey of constraint handling techniques used with evolutionary algorithms. Technical Report Lania-RI-99-04, Laboratorio Nacional de Informtica Avanzada, 1999.

[Cas01] Géry Casiez. Modélisation et simulation du comportement dynamique d'un fil en flexion. Mémoire de DEA, Université des Sciences et Technologies de Lille 1, Lille (France), 26 juin 2001.

[CB01a] Ivan-Ferreira Costa et Remis Balaniuk. LEM - An approach for real time physically based soft tissue simulation. In *International Conference in Automation and Robotics (ICRA'2001)*, Seoul (South Korea), 2001.

[CB01b] Ivan-Ferreira Costa et Remis Balaniuk. Static solution for real time deformable objects with fluid inside. *European Research Consortium for Informatics and Mathematics (ERCIM) News No. 44*, janvier 2001.

[CC78] Edwin Catmull et James H. Clark. Recursively generated b- spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6) :350–355, novembre 1978.

[CDA96] Stéphane Cotin, Hervé Delingette et Nicholas Ayache. Real time volumetric deformable models for surgery simulation. In K.H. Höhne et R. Kikinis, editors, *Visualization in biomedical computing (VBC)*, pages 535–540, Hamburg (Germany), 22-25 septembre 1996. Berlin ; New York : Springer, ©1996.

[CDA99] Stéphane Cotin, Hervé Delingette et Nicholas Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics*, 5(1) :62–73, janvier-march 1999.

[CDA00] Stéphane Cotin, Hervé Delingette et Nicholas Ayache. A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. *The Visual Computer*, 16(8) :437–452, 2000.

[CGC+02] Steve Capell, Seth Green, Brian Curless, Tom Duchamp et Zoran Popović. A multiresolution framework for dynamic deformations. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 41–48, San Antonio, Texas (USA), 21-22 juillet 2002.

[CH97] Deborah A. Carlson et Jessica K. Hodgins. Simulation levels of detail for real-time animation. In Wayne A. Davis, Marilyn Mantei et R. Victor Klassen, editors, *Graphics Interface*, pages 1–8. Canadian Human-Computer Communications Society, mai 1997.

[CH01] Marie-Paule Cani et Samuel Hornus. Subdivision curve primitives : a new solution for interactive implicit modeling. In *Shape Modelling International*, pages 82–88, Genova (Italy), 7-11 mai 2001.

[Cha74] Georges .M. Chaikin. An algorithm for hight speed curve generation. *Computer Graphics and Image Processing*, 3(4) :346–349, décembre 1974.

[Chi95] Hong Sheng Chin. *Stabilization Methods For Simulations Of Constrained Multibody Dynamics*. Department of mathematics, British Columbia, mai 1995.

[Cli02] Michael B. Cline. Rigid body simulation with contact and constraints. Master's thesis, University of British Columbia, novembre 2002.

[Cot97] Stéphane Cotin. *Modèles anatomiques déformables en temps réel : Application à la simulation de chirurgie avec retour d'effort*. Thèse de sciences, Université de Nice Sophia-Antipolis, novembre 1997.

[Cox72] Maurice G. Cox. The numerical evaluation of b-splines. *Journal of the Institute of Mathematics and its Applications*, 10 :134–149, 1972.

[CP03] Michael B. Cline et Dinesh K. Pai. Post-stabilization for rigid body simulation with contact and constraints. In *in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[CR74] Edwin Catmull et Raphael Rom. A class of local interpolating splines. In R.E. Barnhill et R.F. Riesenfeld, editors, *Computer Aided Geometric Design*, volume 2, pages 317–326. Academic Press, New York, 1974.

[CSC03] Daniela Constantinescu, Septimiu E. Salcudean et Elizabeth A. Croft. Haptic feedback using local models of interaction. In *11th symposium on Haptic Interface for Virtual Environment and Teleoperator Systems*, pages 416–421, Los Angeles - California (USA), 22-23 mars 2003.

[dB72] Carl de Boor. On calculating with b-splines. *Journal of Approximation Theory*, 6(1) :50–62, juillet 1972.

[DC95] Mathieu Desbrun et Marie-Paule Cani. Animating soft substances with implicit surfaces. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings, Annual Conference Series*, pages 287–290. ACM SIGGRAPH, Addison Wesley, 1995. Los Angeles, California, published under the name Marie-Paule Gascuel.

[DC96] Mathieu Desbrun et Marie-Paule Cani. Smoothed particles : A new paradigm for animating highly deformable bodies. In R. Boulic et G. Hegron, editors, *Proceedings of EG Workshop on Animation and Simulation*, pages 61–76, Futuroscope, Poitiers (France), 26-30 août 1996. Springer-Verlag. Published under the name Marie-Paule Gascuel.

[DDBC99] Gilles Debunne, Mathieu Desbrun, Alan H. Barr et Marie-Paule Cani. Interactive multiresolution animation of deformable models. In *Eurographics Workshop on Computer Animation and Simulation*, pages 133–144, Milano (Italy), 7-11 septembre 1999.

[DDCB00] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani et Alan H. Barr. Adaptive simulation of soft bodies in real-time. In *Computer Animation*, pages 133–144, Philadelphia - Pennsylvania (USA), 3-5 mai 2000.

[DDCB01] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani et Alan H. Barr. Dynamic real-time deformations using space and time adaptive sampling. In *Computer Graphics (Proceedings of SIGGRAPH)*, Annual Conference Series, Los Angeles, California (USA), 12-17 août 2001. ACM Press / ACM SIGGRAPH. Proceedings of SIGGRAPH'01.

[DMB00] Mathieu Desbrun, Mark Meyer et Alan H. Barr. *Modeling and Animation of Clothes*, chapter 9 : *Interactive Animation of Cloth-like Objects for Virtual Reality*, pages 219–239. Natick, MA, AK Peters, 2000.

[DRG03] Jeremy Denise, David Reversat et André Gagalowicz. Modeling hysteretic behaviour of fabrics. In *Proceeding of Computer Vision / Computer Graphics Collaboration for Model-based Imaging, Rendering, image Analysis and Graphical special Effects (MIRAGE)*, INRIA Rocquencourt (France), 10-11 mars 2003.

[DSB99] Mathieu Desbrun, Peter Schröder et Alan H. Barr. Interactive animation of structured deformable objects. In *Graphics Interface*, pages 1–8, Kingston (Canada), juin 1999.

[DZ93] Paul Dworkin et David Zelter. A new model for efficient dynamic simulation. In Hubbold R.J. et Juan R., editors, *Eurographics Workshop on Computer Animation and Simulation*, pages 135–147, Barcelona (Spain), 4-5 septembre 1993.

[EH95] Matthias Eck et Jan Hadenfeld. Knot removal for b-spline curves. *Computer Aided Geometric Design*, 12(3) :259–282, mai 1995.

[EWW96] Bernhard Eberhardt, Andreas Weber et Strasser Wolfgang. A fast, flexible, particle-system model for cloth draping. In *IEEE Computer Graphics and Applications*, volume 16, pages 52–59, 1996.

[FAM+02] Laure France, Alexis Angelidis, Philippe Meseure, Marie-Paule Cani, Julien Lenoir, François Faure et Christophe Chaillou. Implicit representations of the human intestines for surgery simulations. In Marc Thiriet, editor, *Modelling and Simulation for Computer-aided Medicine and Surgery (MS4CMS)*, volume 12, pages 42–47, Rocquencourt (France), 12-15 novembre 2002. EDP Sciences.

[Far92] Gerald E. Farin. *Curves and Surfaces for Computer-Aided Geometric Design : A Practical Guide.* Academic Press, third edition, 1992.

[Fau98] François Faure. Interactive solid animation using linearized displacement constraints. In *Computer Animation and Simulation 1998*, pages 61–72, 1998. ISBN 3-211-83257-2.

[FB88] David R. Forsey et Richard H. Bartels. Hierarchical b-spline refinement. *Computer Graphics (Proceedings of SIGGRAPH)*, 22(4), 1988.

[FF01] Nick Foster et Ronald Fedkiw. Practical animations of liquids. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 23–30. ACM Press / ACM SIGGRAPH, 2001.

[FLA+04] Laure France, Julien Lenoir, Alexis Angelidis, Philippe Meseure, Marie-Paule Cani, François Faure et Christophe Chaillou. A layered model of a virtual human intestine for surgery simulation. *Medical Images Analysis, Elsevier Sciences*, 2004.

[FLMC02] Laure France, Julien Lenoir, Philippe Meseure et Christophe Chaillou. Simulation of a minimally invasive surgery of intestines. In *Virtual Reality International Conference (VRIC)*, Laval (France), 17-21 juin 2002.

[FM90] Anthony V. Fiacco et Garth P. McCormick. *Nonlinear Programming : Sequential Unconstrained Minimation Techniques.* SIAM, 1990.

[FM97a] Nick Foster et Dimitris N. Metaxas. Controlling fluid animation. In *Computer Graphics International*, pages 178–188, Hasselt, Diepenbeek (Belgium), 23-27 juin 1997.

[FM97b] Nick Foster et Dimitris N. Metaxas. Modeling the motion of hot, turbulent gas. In *Proceedings of SIGGRAPH'97*, Annual Conference Series, pages 181–188, Los Angeles, California (USA), 3-8 août 1997.

[FOA03] Bryan E. Feldman, James F. O'Brien et Okan Arikan. Animating suspended particle explosions. In *Proceedings of ACM SIGGRAPH 2003*, pages 708–715, San Diego, California (USA), 27-31 juillet 2003.

[FSJ01] Ronald Fedkiw, Jos Stam et Henrik Wann Jensen. Visual simulation of smoke. In *In SIGGRAPH 2001 Conference Proceedings*, Annual Conference Series, pages 15–22, Los Angeles, California (USA), 12-17 août 2001.

[Gar] Jean Garrigues. http ://esm2.imt-mrs.fr/gar/gdhtml/coursgdnode74.html.

[GBO04] Tolga G. Goktekin, Adam W. Bargteil et James F. O'Brien. A method for animating viscoelastic fluids. In *Proceedings of ACM SIGGRAPH 2004*. ACM Press, 8-12 août 2004.

[GBS99] Laurent Grisoni, Carole Blanc et Christophe Schlick. Hermitian b-splines. *Computer Graphics Forum (Proceedings of Eurographics)*, 18(4) :237–248, décembre 1999.

[GCG92] Jean-Dominique Gascuel et Marie-Paule Cani-Gascuel. Displacement constraints : a new method for interactive dynamic animation of articulated bodies. In *Third Eurographics Workshop on Animation and Simulation*, Cambridge (United Kingdom), septembre 1992.

[GCMS00] Fabio Ganovelli, Paolo Cignoni, Claudio Montani et Roberto Scopigno. Enabling cuts on multiresolution representation. In *Computer Graphics International*, pages 183–191, Geneva (Switzerland), 14-19 juin 2000. IEEE Computer Society.

[GKS02] Eitan Grinspun, Petr Krysl et Peter Schröder. Charms : a simple framework for adaptive simulation. *ACM Transaction on Graphics (and in Proceedings of SIGGRAPH 2002)*, 21(3) :281–290, 21-26 juillet 2002.

[GM97] Sarah F.F. Gibson et Brian Mirtich. A survey of deformable modeling in computer graphics. Technical Report TR-97-19, Mitsubishi Electric Research Laboratory, Cambridge, Massachusetts (USA), novembre 1997.

[GM03] Laurent Grisoni et Damien Marchal. High performance generalized cylinders visualization. In *Shape Modeling'03 (ACM Siggraph, Eurographics, and IEEE sponsored)*, pages 257–263, Aizu (Japan), juillet 2003.

[GMPO00] Thanh Giang, Robert Mooney, Christopher Peters et Carol O'Sullivan. Aloha : Adaptive level of detail for human animation : Towards a new framework. In *Eurographics*, pages 71–77, 20-25 août 2000. Short Papers.

[GMTT89] Jean-Paul Gourret, Nadia Magnenat-Thalmann et Daniel Thalmann. The use of finite element theory for simulating object and human body deformations and contacts. In *Proceedings of Eurographics*, pages 477–487, Hamburg (Holland), 4-8 septembre 1989.

[GPS01] Herbert Goldstein, Charles P. Poole et John L. Safko. *Classical Mechanics*. Addison Wesley, third edition, juillet 2001.

[Gri99] Laurent Grisoni. *Eléments de Multirésolution en Modélisation Géométrique*. Doctorat d'informatique, Université de Bordeaux I, 15 décembre 1999.

[GVL96] Gene H. Golub et Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore and London, third edition, 1996.

[Hab97] Arash Habibi. *Modèles Physiques Supports de la Relation Mouvement-Forme-Image*. Thèse de doctorat d'informatique, Institut National Polytechnique de Grenoble, janvier 1997.

[HAC03] Samuel Hornus, Alexis Angelidis et Marie-Paule Cani. Implicit modelling using subdivision-curves. *The Visual Computer*, 19(2-3) :94–104, mai 2003.

[Hah88] James K. Hahn. Realistic animation of rigid bodies. In *Proceedings of SIGGRAPH'88*, volume 22 of *4*, pages 299–308, Atlanta, Georgia (USA), 1-5 août 1988.

[Han87] David C. Handscomb. Knot elimination : reversal of the oslo algorithm. *ISNM (International Series of Numerical Mathematics)*, 81 :103–111, 1987.

[HB00] Donald H. House et David E. Breen. *Cloth Modeling and Animation*. A.K. Peters, Ltd, 24 juillet 2000.

[HMC01] Laurent Hilde, Philippe Meseure et Christophe Chaillou. A fast implicit integration method for solving dynamic equations of movement. In *Proceedings of Virtual Reality Software and Technology*, Banff (Canada), 15-17 novembre 2001.

[HPH96] Dave Hutchinson, Martin Preston et Terry Hewitt. Adaptive refinement for mass/spring simulations. In *Eurographics Workshop on Computer Animation and Simulation*, pages 31–45, Poitiers (France), 26-30 août 1996. Springer-Verlag.

[JP99] Doug L. James et Dinesh K. Pai. Artdefo, accurate real time deformable objects. *Computer Graphics (ACM SIGGRAPH 99 Conference Proceedings)*, pages 65–72, 8-13 août 1999.

[JP01] Doug L. James et Dinesh K. Pai. A unified treatment of elastostatic contact simulation for real time haptics. *Haptics-e, The electronic journal of haptics research*, 2(1), 27 septembre 2001.

[JP02] Doug L. James et Dinesh K. Pai. Dyrt : Dynamic response textures for real time deformation simulation with graphics hardware. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 21(3) :582–585, 21-26 juillet 2002.

[JV03] Johan Jansson et Joris S.M. Vergeest. Combining deformable and rigid-body mechanics simulation. *The Visual Computer*, 19(5) :280–290, février 2003.

[KIS] KISMET. http ://www-kismet.iai.fzk.de/.

[Lak95] Roderic S. Lakes. *Experimental methods for study of Cosserat elastic solids and other generalized continua*, chapter 1 of *Continuum models for materials with micro-structure*, pages 1–22. H.-B. Muhlhaus, John Wiley & Sons Ltd - New York, 1995.

[LCGG95] Alexis Lamouret, Marie-Paule Cani-Gascuel et Jean-Dominique Gascuel. Combining physically-based simulation of colliding objects with trajectory control. *Journal of Visualization and Computer Animation*, 6(2) :71–90, 1995.

[Lec04] Antoine Leclercq. *Etude des surfaces pour le jeu vidéo*. PhD thesis, Infogrames, 18 mars 2004. Présentée à l'Université Claude Bernard (Lyon 1).

[LF04] Julien Lenoir et Sylvère Fonteneau. Mixing deformable and rigid-body mechanics simulation. In *Computer Graphics International*, pages 327–334, Hersonissos, Crete (Greece), 16-19 juin 2004.

[LGM+04] Julien Lenoir, Laurent Grisoni, Philippe Meseure, Yannick Rémion et Christophe Chaillou. Smooth constraints for spline variational modeling. In *Graphite*, pages 58–64, Nanyang Technological University (Singapore), 15-18 juin 2004.

[LM87] Tom Lyche et Knut Morken. Knot removal for parametric b-spline curves and surfaces. *Computer Aided Geometric Design*, 4(3) :217–230, novembre 1987.

[LMC02] Julien Lenoir, Philippe Meseure et Christophe Chaillou. Simulation physique de fils. Groupe de travail animation et simulation, Bordeaux (France), 2002.

[LMGC02] Julien Lenoir, Philippe Meseure, Laurent Grisoni et Christophe Chaillou. Surgical thread simulation. In Marc Thiriet, editor, *Modelling and Simulation for Computer-aided Medecine and Surgery (MS4CMS)*, volume 12, pages 102–107, Rocquencourt (France), 12-15 novembre 2002. INRIA, EDP Sciences.

[LMGC04] Julien Lenoir, Philippe Meseure, Laurent Grisoni et Christophe Chaillou. A suture model for surgical simulation. In *2nd International Symposium on Medical Simulation (ISMS'04)*, pages 105–113, Cambridge, Massachusetts (USA), 17-18 juin 2004.

[Loo87] Charles T. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, department of Mathematics, 1987.

[LPC95] Jean Louchet, Xavier Provot et David Crochemore. Evolutionary identification of cloth animation models. In Dimitri Terzopoulos et Daniel Thalmann, editors, *Proceedings of the Eurographics Workshop on Computer Animation and Simulation '95*, pages 44–54, Maastricht (Netherlands), septembre 1995. Springer-Verlag.

[MCG03] Matthias Müller, David Charypar et Markus H. Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation*, pages 154–159, San Diego, California (USA), 26-27 juillet 2003.

[Mes02] Philippe Meseure. Animation basée sur la physique pour les environnements interactifs temps-réel. Habilitation à Diriger des Recherches, 17 décembre 2002.

[ML01] Cesar Mendoza et Christian Laugier. Realistic haptic rendering for highly deformable virtual objects. In *IEEE Virtual Reality Conference*, pages 264–270, Yokohama (Japan), mars 2001.

[ML03] Cesar Mendoza et Christian Laugier. Simulating cutting in surgery applications using haptics and finite element models. Poster Papers in IEEE Virtual Reality, 22-26 mars 2003.

[MLBdC01] Cesar Mendoza, Christian Laugier et François Boux de Casson. Virtual reality cutting phenomena using force feedback for surgery simulations. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Utrecht (The Netherlands), 14-17 octobre 2001.

[MLFC04] Philippe Meseure, Julien Lenoir, Sylvère Fonteneau et Christophe Chaillou. Generalized god-objects : a paradigm for interacting with physically-based virtual worlds. In *Computer Animation and Social Agents*, pages 215–222, Geneva - Switzerland, juillet 7-9 2004.

[MPT99] William A. McNeely, Kevin D. Puterbaugh et James J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. *Computer Graphics (Proceedings of SIGGRAPH)*, pages 401–408, août 1999.

[MS98] Jon McCormack et Andrei Sherstyuk. Creating and rendering convolution surfaces. *Computer Graphics Forum*, 17(2) :113–121, 1998.

[MST+04] Matthias Müller, Simon Schirm, Matthias Teschner, Bruno Heidelberger, et Markus H. Gross. Interaction of fluids with deformable solids. *Computer Animation & Virtual Worlds*, 15(3-4) :159–171, juillet 2004.

[MT92] Dimitris N. Metaxas et Demetri Terzopoulos. Dynamic deformation of solid primitives with constraints. *Computer Graphics Proceedings, Annual Conference Series (ACM SIGGRAPH)*, 26(2) :309–312, 26-31 juillet 1992.

[MT96] Tim McInerney et Demetri Terzopoulos. Deformable models in medical image analysis : A survey. *Medical Image Analysis*, 1(2) :91–108, juin 1996.

[MW88] Mathew Moore et Jane Wilhelms. Collision detection and response for computer animation. *Computer Graphics (Proceedings of SIGGRAPH)*, 22(4) :289–298, 1-5 août 1988.

[Nie03] Han Wen Nienhuys. *Cutting in deformable objects.* PhD thesis, Utrecht University (Netherlands), 2003.

[NN03] Cyril Ngo-Ngoc. *Modélisation Non Linéaire et Simulation des Matériaux Souples Textiles (Application aus essais Kawabata).* PhD thesis, Université de Lille 1, 27 janvier 2003.

[NNB04] Cyril Ngo-Ngoc et Samuel Boivin. Nonlinear cloth simulation. Rapport de Recherche n°5099, INRIA, janvier 2004.

[NNR01] Olivier Nocent, Jean-Michel Nourrit et Yannick Rémion. Towards mechanical level of detail for knitwear simulation. In V. Skala, editor, *Proceedings of Winter School of Computer Graphics (WSCG)*, Plzen (Czech Republic), 5-9 février 2001.

[Noc99] Olivier Nocent. Justification des équations de lagrange : Un éclaircissement sur les fondements de la mécanique du siècle des lumières. Rapport Interne, 1999.

[Nou99] Jean-Michel Nourrit. *Modélisation, animation et visualisation de textiles à base de mailles.* Thèse de doctorat en informatique, Université de Reims, 1999.

[NR01] Olivier Nocent et Yannick Rémion. Continuous deformation energy for dynamic material splines subject to finite displacements. In M.P. Cani, N. Magnenat-Thalmann et D. Thalmann, editors, *Eurographics Workshop on Computer Animation and Simulation*, pages 87–98, Manchester (United Kingdom), 2-3 septembre 2001. Springer Verlag.

[NT98] Luciana Porcher Nedel et Daniel Thalmann. Real time muscle deformations using mass-spring systems. In *Proceedings of CGI'98*, Hannover (Germany), 22-26 juin 1998. IEEE Computer Society Press.

[NvdS00] Han Wen Nienhuys et Frank van der Stappen. Combining finite element deformation with cutting for surgery simulations. In A. de Sousa et J.C. Torres, editors, *Eurographics (short presentation)*, pages 43–52, Interlaken (Switzerland), 20-25 août 2000.

[NvdS01] Han Wen Nienhuys et Frank van der Stappen. A surgery simulation supporting cuts and finite element deformation. In Wiro J. Niessen et Max A. Viergever, editors, *Medical Image Computing and Computer-Assisted Intervention*, volume 2208 of *Lecture Notes in Computer Science*, pages 145–152, Utrecht (Netherlands), 14-17 octobre 2001. Springer-Verlag.

[Nvi] Nvidia+programmer+online+reference. http ://developer.nvidia.com.

[Pai02] Dinesh K. Pai. STRANDS : Interactive simulation of thin solids using cosserat models. *Computer Graphics Forum (Proceedings of Eurographics)*, 21(3) :347–352, 2002.

[Pat00] Mayuresh J. Patil. Decoupled second-order equations and modal analysis of a general non-conservative system. In *Proceedings of the AIAA Dynamics Specialists Conference*, Atlanta, Georgia (USA), 3-6 avril 2000.

[PB88a] John C. Platt et Alan H. Barr. Constrained differential optimization. In *IEEE Neural Information Processing Systems Foundation (NIPS)*, pages 612–621, 1988.

[PB88b] John C. Platt et Alan H. Barr. Constraint methods for flexible models. *Computer Graphics (Proceedings of ACM SIGGRAPH)*, 22(4) :279–288, août 1988.

[PBP96] Emmanuel Promayon, Pierre Baconnier et Claude Puech. Physically-based deformations constrained in displacements and volume. *Computer Graphics Forum (Proceedings of Eurographics)*, 15(3) :155–164, 26-30 août 1996.

[PC01] Frank Perbet et Marie-Paule Cani. Animating prairies in real-time. In *ACM Interactive 3D Graphics*, USA, mars 2001.

[PDA00] Guillaume Picinbono, Hervé Delingette et Nicholas Ayache. Real-time large displacement elasticity for surgery simulation : Non-linear tensor-mass model. In Scott L. Delp, Anthony M. DiGioia, et Branislav Jaramaz, editors, *Third International Conference on Medical Robotics, Imaging And Computer Assisted Surgery : MICCAI 2000*, pages 643–652, Pittsburgh, Pennsylvania (USA), 11-14 octobre 2000. Springer.

[PDA03] Guillaume Picinbono, Hervé Delingette et Nicholas Ayache. Non-linear anisotropic elasticity for real-time surgery simulation. *Graphical Models*, 65(5) :305–321, septembre 2003.

[Pic01] Guillaume Picinbono. *Modèles géométriques et physiques pour la simulation d'interventions chirurgicales*. PhD thesis, Université de Nice - Sophia Antipolis, 12 février 2001.

[PLDA00] Guillaume Picinbono, Jean-Christophe Lombardo, Hervé Delingette et Nicholas Ayache. Anisotropic elasticity and forces extrapolation to improve realism of surgery simulation. In *IEEE International Conference Robotics and Automation*, pages 596–602, San Francisco, California (USA), 24-28 avril 2000. IEEE.

[Pro95] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In Wayne A. Davis et Przemyslaw Prusinkiewicz, editors, *Graphics Interface '95*, pages 147–154. Canadian Human-Computer Communications Society, 1995.

[PT97] Les Piegl et Wayne Tiller. *The NURBS Book*. Springer-Verlag, New York, second edition, 1997.

[PW89] Alex Pentland et John Williams. Good vibrations : Modal dynamics for graphics and animation. In *Computer Graphics (ACM SIGGRAPH)*, volume 23 of *3*, pages 215–222, Boston, Massachusetts (USA), 31 juillet - 4 août 1989.

[PX04] Qing Pan et Guoliang Xu. Fast evaluation of the improved loop's subdivision surfaces. In *Proceedings of Geometric Modeling and Processing (GMP)*, pages 205–214, Beijing (China), 13-15 avril 2004.

[QT96] Hong Qin et Demetri Terzopoulos. D-NURBS : A Physics-Based Framework for Geometric Design. *IEEE Transactions on Visualization and Computer Graphics*, 2(1) :85–96, mars 1996.

[RBAB02] Angel del Rio, Montserrat Bóo, Margarita Amor et Javier Díaz Bruguera. Hardware implementation of the subdivision loop algorithm. In *Proceedings of 28th Euromicro Conference - Multimedia and Telecommunications*, pages 189–199, Dortmund (Germany), 4-6 septembre 2002. IEEE Computer Society.

[RCFC03] Laks Raghupathi, Vincent Cantin, François Faure et Marie-Paule Cani. Real-time simulation of self-collisions for virtual intestinal surgery. In Nicholas Ayache et Herve Delingette, editors, *International Symposium on Surgery Simulation and Soft Tissue Modeling*, volume 2673 of *Lecture Notes in Computer Science*, pages 15–26, Juan-Les-Pins (France), 12-13 juin 2003. Springer.

[Rie75] Richard F. Riesenfeld. On Chaikin's algorithm. *IEEE Computer Graphics and Applications*, 4(3) :304–310, 1975.

[RK98] Diego C. Ruspini et Oussama Khatib. Dynamic models for haptic rendering. In *Advances in Robot Kinematics (ARK)*, pages 523–532, Salzburg (Austria), juin 1998.

[RKK97] Diego C. Ruspini, Krasimir Kolarov et Oussama Khatib. The haptic display of complex graphical environments. *Computer Graphics (Proceedings of SIGGRAPH)*, pages 345–352, août 1997.

[Rém00] Yannick Rémion. Animation dynamique : moteur lagrangien généraliste et applications. Habilitation à diriger des recherches, 20 décembre 2000. Université de Reims Champagne-Ardenne.

[Rém03] Yannick Rémion. Prise en compte de "contraintes à variables libres". Technical Report 03-02-01, LERI - Université de Reims, Champagne-Ardenne, février 2003.

[RNG99] Yannick Rémion, Jean-Michel Nourrit et Didier Gillard. Dynamic animation of spline like objects. In *Proceedings of the WSCG'1999 Conference*, pages 426–432, Plzen (Czech Republic), 8-12 février 1999.

[RNG00] Yannick Rémion, Jean-Michel Nourrit et Didier Gillard. A dynamic animation engine for generic spline objects. *Journal of Visualization and Computer Animation*, 11(1) :17–26, février 2000.

[RNN00] Yannick Rémion, Jean-Michel Nourrit et Olivier Nocent. Dynamic animation of n-dimensional deformable objects. In *Proceedings of the WSCG'2000 Conference*, pages 147–154, Plzen (Czech Republic), 7-10 février 2000.

[RNN01] Yannick Rémion, Jean-Michel Nourrit et Olivier Nocent. D-dimensional parametric models for dynamic animation of deformable objects. *The Visual Computer*, 17(3) :167–178, mai 2001.

[She98] Andrei Sherstyuk. Fast ray tracing of implicit surfaces. In *Implicit Surfaces*, pages 145–153, Seattle, Washington (USA), juin 1998.

[She99a] Andrei Sherstyuk. *Convolution surfaces in computer graphics*. PhD thesis, School of computer science and software engineering, Monash University, Australia, janvier 1999.

[She99b] Andrei Sherstyuk. Fast ray tracing of implicit surfaces. *Computer Graphics Forum*, 18(2), juin 1999. An earlier version of this work was presented at the 3rd International Workshop on Implicit Surfaces held in Seattle in 1998.

[She99c] Andrei Sherstyuk. Kernel functions in convolution surfaces : a comparative analysis. *The Visual Computer*, 15(4) :171–182, 1999.

[SHGO02] Chen Shen, Kris K. Hauser, Christine M. Gatchalian et James F. O'Brien. Modal analysis for real-time viscoelastic deformation. Technical Sketch in ACM SIGGRAPH (San Antonio, Texas (USA)), 21-26 juillet 2002.

[SIM] SIMBIONIX. http ://www.simbionix.com/.

[SL02] Kenneth Sundaraj et Christian Laugier. Physically realistic simulation of large deformations using lem for interactive applications. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Lausanne (Switzerland), 30 septembre - 4 octobre 2002.

[SLB03] Kenneth Sundaraj, Christian Laugier et François Boux-de-Casson. Intra-operative ct-free examination system for anterior cruciate ligament reconstruction. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Las Vegas, Nevada (USA), 27-31 octobre 2003.

[SLC01] Kenneth Sundaraj, Christian Laugier et Ivan-Ferreira Costa. An approach to lem modelling : Construction, collision detection and dynamic simulation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Hawaii (USA), 29 octobre - 3 novembre 2001.

[Sta98a] Jos Stam. Evaluation of loop subdivision surfaces. In SIGGRAPH'98 CD-ROM Proceedings, juillet 1998.

[Sta98b] Jos Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *ACM SIGGRAPH*, pages 395–404, Orlando, Florida (USA), 19-24 juillet 1998.

[Sta99] Jos Stam. Stable fluids. In *SIGGRAPH 99 Conference Proceedings*, Annual Conference Series, pages 121–128, Los Angeles, California (USA), 8-13 août 1999.

[Sta00] Jos Stam. Interacting with smoke and fire in real time. *Communications of the ACM*, 43(7) :76–83, 2000.

[Sta03] Jos Stam. Flows on surfaces of arbitrary topology. *Transaction On Graphics (SIGGRAPH)*, 22(3) :724–731, juillet 2003.

[Sun04] Kenneth Sundaraj. *Real-Time Dynamic Simulation and 3D Interaction of Biological Tissue : Application to Medical Simulators*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble (France), janvier 2004.

[Sur] Surgical+Science. http ://www.surgical-science.com/.

[SY01] Jiao-Ying Shi et Li-Xia Yan. Deformationand cutting in virtual surgery. In *International Workshop on Medical Imaging and Augmented Reality (MIAR)*, pages 95–102, Hong Kong (China), 10-12 juin 2001.

[SZBN03] Thomas W. Sederberg, Jianmin Zheng, Almaz Bakenov et Ahmad Nasri. T-splines and t-nurccs. *ACM Transactions on Graphics*, 22(3) :477–484, 2003.

[TGG00] Matthias Teschner, Sabine Girod et Bernd Girod. Direct computation of nonlinear soft-tissue deformation. In *Proceedings of Vision, Modeling, and Visualization VMV'00*, pages 383–390, Saarbrücken (Germany), 22-24 novembre 2000.

[TGMC03] Frédéric Triquet, Laurent Grisoni, Philippe Meseure et Christophe Chaillou. Realtime visualization of implicit objects with contact control. In *Graphite*, volume 1, février 2003. http ://www.anzgraph.org/graphite2003.

[Til92] Wayne Tiller. Knot-removal algorithms for nurbs curves and surfaces. *Computer Aided Design*, 24(8) :445–453, août 1992.

[TMC01] Frédéric Triquet, Philippe Meseure et Christophe Chaillou. Fast polygonization of implicit surfaces. In *WSCG*, volume 2, pages 283–290, février 2001. http ://www.wscg.zcu.cz.

[TPBF87] Demetri Terzopoulos, John C. Platt, Alan H. Barr et Kurt Fleischer. Elastically deformable models. *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, 21(4) :205–214, 27-32 juillet 1987.

[TQ94] Demetri Terzopoulos et Hong Qin. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2) :103–136, 1994.

[VCMT95] Pascal Volino, Martin Courchesne et Nadia Magnenat-Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proceedings of SIGGRAPH, Annual Conference Series*, pages 137–144, Los Angeles, California (USA), 6-11 août 1995.

[VGW97] Allen Van Gelder et Jane Wilhelms. Simulation of elastic membranes with triangulated spring meshes. Technical Report UCSC-CRL-97-12, University of California, Santa Cruz, California (USA), 3 juillet 1997.

[VMT00] Pascal Volino et Nadia Magnenat-Thalmann. Implementing fast cloth simulation with collision response. In *Computer Graphics International (IEEE Computer Society DL)*, pages 257–268, Geneva (Switzerland), 19-24 juin 2000.

[VVW04] Lara M. Vigneron, Jacques G. Verly et Simon K. Warfield. On extended finite element model (xfem) for modelling of organ deformations associated with surgical cuts. In Stéphane Cotin et Dimitris Metaxas, editors, *International Symposium on Medical Simulation*, pages 134–143, Cambridge, Massachusetts (USA), 17-18 juin 2004. Springer - LNCS 3078.

[WFB87] Andrew Witkin, Kurt Fleischer et Alan H. Barr. Energy constraints on parameterized models. *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, 21(4) :225–232, 27-32 juillet 1987.

[WH04] Wen Wu et Pheng Ann Heng. A hybrid condensed finite element model with gpu acceleration for interactive 3d soft tissue cutting. *Computer Animation and Virtual Worlds*, 15(3-4) :219–227, juillet 2004.

[WMW86] Geoff Wyvill, Craig McPheeters et Brian Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4) :227–234, août 1986.

[WW90] Andrew Witkin et William Welch. Fast animation and control of nonrigid structures. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4) :243–252, 1990.

[WW92] William Welch et Andrew Witkin. Variational surface modeling. *Computer Graphics (Proceedings of SIGGRAPH)*, 26(2) :157–166, 1992.

[YC00] Qing Yu et I-Ming Chen. A direct violation correction method in numerical simulation of constrained multibody systems. *International Journal of Computational Mechanics*, 26(1) :52–57, 2000.

[ZC00] Yan Zhuang et John Canny. Haptic interaction with global deformations. In *IEEE Interantional Conference on Robotics and Automation*, San Francisco - California (USA), 24-28 avril 2000.

[ZFV02] Florence Zara, François Faure et Jean-Marc Vincent. Physical cloth simulation on a pc cluster. In D. Bartz, X. Pueyo, et E. Reinhard, editors, *Parallel Graphics and Visualisation*, EG Workshop Proceedings, pages 105–112, Blaubeuren (Germany), 8-9 septembre 2002.

[ZPS01] Yu Zhang, Edmond C. Prakash et Eric Sung. Real-time physically-based facial expression animation using mass-spring system. In *Computer Graphics International*, pages 347–350, Hong Kong, 3-6 juillet 2001.

[ZS95] Craig Zilles et John Kenneth Salisbury. A constrained based god-object method for haptic display. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 146–151, Pittsburgh, 1995.

[ZSD+00] Denis Zorin, Peter Schröder, Tony DeRose, Leif Kobbelt, Adi Levin et Wim Sweldens. Subdivision for modeling and animation. Course note of SIGGRAPH 2000 - New Orleans, Louisiana (USA), 23-28 juillet 2000.

Index des auteurs cités

$\mathcal{N}$

$\mathcal{O}$

$\mathcal{P}$

$\mathcal{Q}$

$\mathcal{R}$

$\mathcal{S}$

$\mathcal{T}$

$\mathcal{U}$

$\mathcal{V}$

$\mathcal{W}$

$\mathcal{X}$

$\mathcal{Y}$

$\mathcal{Z}$

Annexe A

Les énergies de déformation

Les déformations d'un objet se mesurent par rapport aux déformations subies par l'objet mais aussi par rapport aux contraintes propres au matériau.

Du point de vue des notations, soit un point $\mathbf{x}$ de l'objet dans sa configuration non déformée, notons $\mathbf{u}(\mathbf{x})$ le vecteur déplacement qui permet de passer de la configuration non déformée à la configuration déformée. Ainsi, le point $\mathbf{x}$ subit le déplacement $\mathbf{u}(\mathbf{x})$ pour aboutir au point $\mathbf{y}$ de l'objet dans sa configuration déformée :

$$\mathbf{u}(\mathbf{x}) = \mathbf{y} - \mathbf{x}$$

Ainsi posé, le déplacement de l'objet est donné par le champ de vecteur $\mathbf{u}(\mathbf{x})$, ce qui permet d'évaluer également les déformations dans la structure.

A.1 Les Tenseurs de déformation

Les déformations se mesurent à l'aide des variations du champ de vecteur $\mathbf{u}(\mathbf{x})$. Ainsi, les déformations de l'objet peuvent se mesurer à l'aide d'un tenseur de déformation. Ce tenseur de déformation est à choisir selon le type de déplacements autorisés.

- En grands déplacements, on choisira le tenseur de Green/Lagrange qui permet de prendre en compte les déformations axiales et transversales :

$$G_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} + \frac{\partial u}{\partial x_i}\frac{\partial u}{\partial x_j}\right)$$

- En petits déplacements, on choisira le tenseur de Cauchy qui est une approximation du tenseur de Green/Lagrange à l'ordre un :

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

  Le tenseur de Cauchy ne mesure les déformations que sur les axes principaux, aucune mesure n'est effectuée sur les axes transversaux.
- En grande déformation, on peut préférer le tenseur d'Almansi au tenseur de Green/Lagrange puisqu'il se base sur la configuration déformée de l'objet :

$$A_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial y_j} + \frac{\partial u_j}{\partial y_i} - \frac{\partial u}{\partial y_i}\frac{\partial u}{\partial y_j}\right)$$

  Ainsi, pour de grandes déformations, les tenseurs de Green/Lagrange et d'Almansi sont différents.

A.2 Les Tenseurs de contrainte

Les énergies de déformation sont définies à l'aide d'un tenseur de déformation et d'une loi de constitution. La loi de constitution se base sur des coefficients qui déterminent les propriétés physiques du matériau et précise comment utiliser le tenseur de déformation pour calculer l'énergie de déformation de l'objet. La loi de constitution s'exprime également sous forme de tenseur, le tenseur de contrainte.

- Le second tenseur de contrainte de Piola-Kircchoff permet par exemple de définir un matériau de constitution linéaire à l'aide de 36 coefficients. Si le matériau est isotrope (de comportement identique dans toutes les directions), ce nombre se réduit à 2 : λ et μ appelés *coefficients de Lamé*. Ces deux coefficients se déterminent à l'aide du module d'Young E et du coefficient de Poisson ν du matériau :

$$\lambda = \frac{E\nu}{(1-2\nu)(1+\nu)} \text{ et } \mu = \frac{E}{2(1+\nu)}$$

  La loi de constitution de Piola-Kircchoff permet d'aboutir à l'énergie élastique [Mes02, Gar] :

$$e = \frac{\lambda}{2} tr\left(\epsilon\right)^2 + \mu tr\left(\epsilon^2\right)$$

  où $tr(a)$ désigne la trace du tenseur a, et ϵ le tenseur de déformation choisi.
- Une loi de constitution non linéaire est celle de Mooney-Rivlin pour les matériaux hyper-élastiques. Cette loi est définie à l'aide du tenseur d'Almansi et permet d'exprimer l'énergie élastique par :

$$e = c_1(tr(\epsilon) - 3) + 2c_2(tr(\epsilon)^2 - tr(\epsilon^2) - 6)$$

  où $tr(a)$ désigne la trace du tenseur a, ϵ le tenseur de déformation choisi et c_1 et c_2 les paramètres physiques du matériau.

Annexe B

Propriétés des courbes de Bézier

Pour rappel, l'équation d'une courbe de Bézier de $n+1$ points de contrôle $\mathbf{q_i}$ est définie par :

$$\forall s \in [0,1] \qquad \mathbf{P}(s) = \sum_{i=0}^{n} \mathbf{q_i} B_i^n(s)$$

où B_i^n sont les polynômes de Bernstein :

$$B_i^n(s) = C_n^i s^i (1-s)^{n-i}$$
$$\text{avec } C_n^i = \frac{n!}{i!(n-i)!}$$

Relativement à la classification établie en section (1.2.1.4), les propriétés des courbes de Béziers sont :

Normalité :

Les polynômes de Bernstein sont issus du développenement binômiale de $(s+(1-s))^n$, ce qui démontre que les fonctions d'influence des splines de Bézier vérifient la propriété de normalité :

$$\forall s \in [0,1] \qquad 1 = 1^n = (s+(1-s))^n = \sum_{i=0}^{n} C_n^i s^i (1-s)^{n-i} = \sum_{i=0}^{n} B_i^n(s)$$

Positivité :

Les courbes de Bézier sont positives puisque C_n^i est une fraction rationelle de nombre entier naturel, $s \in [0,1]$ donc s^i et $(1-s)^i \in [0,1]$. D'où, la relation $B_i^n(s) \geq 0$.

Régularité :

L'étude des oscillations des courbes de Bézier nous amène à calculer la dérivé des polynômes de Bernstein :

$$dB_n^i(s)/ds = C_n^i s^{i-1}(1-s)^{n-i-1}(i-ns)$$

On remarque donc que les fonctions de Bernstein $B_i^n(s)$ sont strictement croissantes sur $[0, i/n]$, admettent un maximum en i/n puis sont strictement décroissantes sur $[i/n, 1]$.

$$\forall s < \tfrac{i}{n} \qquad \forall k \in \{1..n-i\}$$

$$\begin{aligned} B_{i+k}^n(s) &= C_n^{i+k} s^{i+k} (1-s)^{n-i-k} \\ &= C_n^i \prod_{j=1}^{k} \frac{n-i-j+1}{i+j} s^i (1-s)^{n-i} \frac{s^k}{(1-s)^k} \\ &= B_i^n(s) \prod_{j=1}^{k} \frac{n-i-j+1}{i+j} \frac{s^k}{(1-s)^k} \end{aligned}$$

or, $s < \frac{i}{n}$ donc $s^k < \left(\frac{i}{n}\right)^k$ et $\left(\frac{1}{1-s}\right)^k < \left(\frac{n}{n-i}\right)^k$, d'où :

$$\begin{aligned} B_{i+k}^n(s) &< B_i^n(s) \prod_{j=1}^{k} \frac{n-i-j+1}{i+j} \frac{i^k}{(n-i)^k} \\ &< B_i^n(s) \prod_{j=1}^{k} \frac{i(n-i-j+1)}{(n-i)(i+j)} \end{aligned}$$

or, pour $j \in \{1..k\}$, $i+j > i$ donc $\frac{i}{i+j} < 1$ et, $n-i-j+1 \leq n-i$ donc $\frac{n-i-j+1}{n-i} \leq 1$. L'expression se réduit donc à :

$$B_{i+k}^n(s) \leq B_i^n(s)$$

De la même manière, on démontre que :

$$\forall s > \tfrac{i}{n} \quad \forall k \in \{1..i\} \quad B_{i-k}^n(s) \leq B_i^n(s)$$

Ainsi, les courbes de Bézier vérifient la propriété de régularité.

Approximation des points de contrôle extrémités :

On détermine aisément que

$$\begin{aligned} & B_0^n(0) = 1 \quad B_n^n(1) = 1 \\ \forall k \in \{1..n-1\} \quad & B_k^n(0) = 0 \quad B_k^n(1) = 0 \end{aligned}$$

d'où l'on peut en conclure que toute courbe de Bézier interpole ses points de contrôle extrémités $\mathbf{q_0}$ et $\mathbf{q_n}$, mais elle reste une spline d'approximation.

Annexe C

Propriétés des B-Splines

Le lecteur est renvoyé à la section (1.2.1.9) introduisant les B-splines pour le choix des notations.

De manière générale et relativement à la classification établie en section (1.2.1.4), les propriétés des B-splines sont :

Normalité : La démonstration se fait par récurrence sur l'ordre.

La relation à l'ordre 1

$$\sum_{i=0}^{n} N_i^1(s) = 1$$

est une évidence au vue de l'équation (1.22).

Pour appliquer la récurrence, il faut supposer la relation vraie à un rang $k-1$ donné, soit alors l'hypothèse de récurrence $\sum_{i=0}^{n} N_i^{k-1}(s) = 1$. Examinons ce qui ce passe alors au rang k :

$$\forall j \in \{k-1, .., n\},\ \forall s \in [s_j, s_{j+1}[$$

$$\begin{aligned}
\sum_{i=0}^{n} N_i^k(s) &= \sum_{i=j+1-k}^{k} N_i^k(s) = \sum_{i=j+1-k}^{j} \left[\frac{s - s_i}{s_{i+k-1} - s_i} N_i^{k-1}(s) + \frac{s_{i+k} - s}{s_{i+k} - s_{i+1}} N_{i+1}^{k-1}(s) \right] \\
&= \sum_{i=j+1-k}^{j} \frac{s - s_i}{s_{i+k-1} - s_i} N_i^{k-1}(s) + \sum_{i=j+1-k}^{j} \frac{s_{i+k} - s}{s_{i+k} - s_{i+1}} N_{i+1}^{k-1}(s)
\end{aligned} \tag{C.1}$$

Or, $N_{j+1-k}^{k-1}(s) = 0$ pour $s \in [s_j, s_{j+1}[$ et de la même façon, $N_{j+1}^{k-1}(s) = 0$ pour $s \in [s_j, s_{j+1}[$. Donc, les sommations perdent chacune un indice :

$$\begin{aligned}
\sum_{i=0}^{n} N_i^k(s) &= \sum_{i=j+2-k}^{j} \frac{s - s_i}{s_{i+k-1} - s_i} N_i^{k-1}(s) + \sum_{i=j+1-k}^{j-1} \frac{s_{i+k} - s}{s_{i+k} - s_{i+1}} N_{i+1}^{k-1}(s) \\
&= \sum_{i=j+2-k}^{j} \frac{s - s_i}{s_{i+k-1} - s_i} N_i^{k-1}(s) + \sum_{i=j+2-k}^{j} \frac{s_{i+k-1} - s}{s_{i+k-1} - s_i} N_i^{k-1}(s) \\
&= \sum_{i=j+2-k}^{j} \left[\frac{s - s_i}{s_{i+k-1} - s_i} + \frac{s_{i+k-1} - s}{s_{i+k-1} - s_i} \right] N_i^{k-1}(s) = \sum_{i=j+2-k}^{j} N_i^{k-1}(s) \\
&= \sum_{i=0}^{n} N_i^{k-1}(s) = 1
\end{aligned} \tag{C.2}$$

Les B-splines possèdent donc la propriété de normalité.

Positivité :

Là encore, la démonstration se fait par récurrence sur l'ordre de la spline. La propriété à démontrer est cependant plus précise :

$$\forall k \geq 1,\ \forall i \in \{0, .., n\}$$

$$N_i^k(s) \begin{cases} > 0 & \text{pour } s \in [s_i, s_{i+k}[\\ = 0 & \text{sinon} \end{cases}$$

La propriété de positivité apparaît clairement au rang 1 par l'équation (1.22).

L'hypothèse de récurrence suppose la relation vraie au rang $k-1$, le rang k procure donc l'équation (1.21).

Le support des fonctions $N_i^{k-1}(s)$ et $N_{i+1}^{k-1}(s)$ sont respectivement $[s_i, s_{i+k-1}[$ et $[s_{i+1}, s_{i+k}[$ (par hypothèse de récurrence). La somme de ces deux termes définit donc une fonction dont le support est $[s_i, s_{i+k}[$. De plus, tous les dénominateurs apparaissant dans l'équation (1.21) sont positifs (ou nuls) car ils sont formés de la différence de deux nœuds du vecteur de nœuds, et plus précisemment, d'un nœud avec un nœud d'indice plus petit. Or, le vecteur de nœuds est ordonné de manière croissante. Ainsi donc, les dénominateurs sont tous positifs (ou nuls). Quant aux numérateurs, ils font intervenir la variable s qui est supposée dans l'intervalle $[s_i, s_{i+k}[$, les différences formées sont donc elles aussi positives ou nulles.

Il en résulte la relation

$$N_i^k(s) \begin{cases} > 0 & \text{pour } s \in [s_i, s_{i+k}[\\ = 0 & \text{sinon} \end{cases}$$

qui conclut la démonstration de la positivité des B-splines.

La propriété qui a été démontrée permet également de définir le support d'une fonction d'influence $N_i^k(s)$ comme l'intervalle $[s_i, s_{i+k}[$.

Localité :

Le support fini $[s_i, s_{i+k}[$ des fonctions d'influence $N_i^k(s)$ permet de conclure que les B-splines sont locales. L'ordre de la localité est défini par le nombre de segments présents dans le support, soit k.

Régularité :

La propriété de régularité se démontre à l'aide de la formule de convolution des B-splines.

Annexe D

Repère de Frenet

Les notations vectorielles sont fléchées (i.e. un vecteur **u** est noté $\vec{u}$) dans cette annexe pour faciliter la compréhension.

Le repère de Frenet est un repère local à une courbe. Si la courbe est définie par $\vec{P}(s)$ avec s l'abscisse paramétrique, on a alors le vecteur tangent à la courbe (au sens paramétrique du terme) :

$$\vec{T} = \frac{d\vec{P}}{ds}$$

Si l'on note c l'abscisse curviligne de la courbe, on a alors la relation $\frac{dc}{ds} = |\vec{T}|$ qui représente l'élongation locale à la courbe.

De la même manière, le vecteur de courbure locale est définie par :

$$\vec{C} = \frac{d\vec{T}}{ds} = \frac{d^2\vec{P}}{ds^2}$$

Application sur un cercle :

Si l'on considère maintenant un point $\vec{P}$ se déplaçant sur un cercle de centre $\vec{O}$ et de rayon R. On définit le vecteur $\vec{r} = \vec{OP} = \vec{u}R$ et l'angle $\theta = (\vec{i}, \vec{u})$ dans le repère orthonormé direct $(O; \vec{i}, \vec{j})$ (comme indiqué sur le schéma (a) de la figure (D.1)).

Le vecteur tangent est alors défini par

$$\vec{T} = \frac{d\vec{r}}{ds} = R\frac{d\vec{u}}{ds} = R\frac{d\vec{u}}{d\theta}\frac{d\theta}{ds}$$

où $\frac{d\vec{u}}{d\theta}$ est un vecteur unitaire, orthogonal direct au vecteur $\vec{u}$, on le note $\vec{p}$ (i.e. $(\vec{u}, \vec{p})$ forme une base orthonormée directe). On obtient donc après ré-écriture :

$$\vec{T} = R\frac{d\theta}{ds}\vec{p}$$

Sachant que $\frac{d\vec{p}}{ds} = \frac{d\vec{p}}{d\theta}\frac{d\theta}{ds} = -\frac{d\theta}{ds}\vec{u}$, le vecteur courbure est donné par :

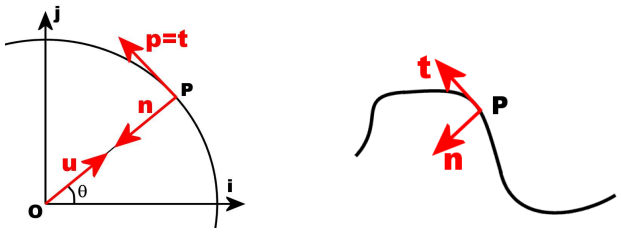
$$\vec{C} = \frac{d\vec{T}}{ds} = R\left[-\left(\frac{d\theta}{ds}\right)^2\vec{u} + \frac{d^2\theta}{ds^2}\vec{p}\right] = -R\left(\frac{d\theta}{ds}\right)^2\vec{u} + R\frac{d^2\theta}{ds^2}\vec{p}$$

Application à une courbe quelconque :

Le principe étudié pour un cercle est applicable localement à une courbe de l'espace comme indiqué sur le schéma (b) de la figure (D.1). On introduit alors la notion d'abscisse curviligne $c(s)$ qui est défini sur le cercle par $c = R\theta$, d'où la relation $\frac{dc}{ds} = R\frac{d\theta}{ds}$.

On obtient donc la tangente

$$\vec{T} = R\frac{d\theta}{ds}\vec{p} = \frac{dc}{ds}\vec{p}$$



(a) Repère de Frenet sur un cercle (b) Repère de Frenet sur une courbe

FIG. D.1 – Schéma du repère local de Frenet

d'où l'on peut remarquer que $\vec{p}$ est le vecteur unitaire tangent localement à la courbe. De la même manière, le vecteur courbure est donné par

$$\vec{C} = -\frac{1}{R}\left(\frac{dc}{ds}\right)^2 \vec{u} + \frac{d^2c}{ds^2}\vec{p}$$

On remarque que $\vec{p}$ est le vecteur unitaire tangent à la courbe orienté dans le sens positif du déplacement : $\vec{p} = \frac{\vec{T}}{|\vec{T}|}$. On renomme donc ce vecteur comme étant le vecteur unitaire localement tangent $\vec{t}$:

$$\vec{t} = \vec{p} = \frac{\vec{T}}{|\vec{T}|} = \frac{d\vec{P}/ds}{|d\vec{P}/ds|}$$

De même, le vecteur $\vec{u}$ est le vecteur unitaire localement orthogonal à la courbe dirigé vers le point de la courbe. Or, le repère de Frenet est défini sur un point de la courbe, les vecteurs de la base partent donc de ce point pour s'en éloigner. On redéfinit donc le vecteur normal $\vec{n}$ comme étant le vecteur opposé à $\vec{u}$:

$$\vec{n} = -\vec{u} = \frac{d\vec{t}/ds}{|d\vec{t}/ds|}$$

Le repère de Frenet est alors défini au point d'abscisse paramétrique s par $(\vec{P}; \vec{t}, \vec{n})$. Dans ce repère, les vecteurs tangent et courbure s'expriment comme suit :

$$\vec{T} = \frac{dc}{ds}\vec{t} \quad \texttt{et} \quad \vec{C} = \frac{d^2c}{ds^2}\vec{t} + \left(\frac{dc}{ds}\right)^2 \frac{1}{R}\vec{n}$$

avec R le rayon de courbure local à la courbe. Généralement, on définit la courbure locale ρ comme l'inverse du rayon de courbure : $\rho = \frac{1}{R}$.

Extension en 3D :

En 3D, le repère de Frenet est appelé trièdre de Frenet, il est composé du vecteur tangent $\vec{t}$, du vecteur normal $\vec{n}$ et d'un vecteur appelé bi-normal $\vec{b}$ qui est le produit vectoriel des deux premiers afin d'obtenir un repère orthonormé direct local. A l'aide de ce troisième vecteur, un paramètre de torsion géométrique τ peut être défini comme la norme de la variation du vecteur bi-normal par rapport à l'abscisse curviligne : $\tau = \left|\frac{d\vec{b}}{dc}\right|$.

La base $(\vec{t}, \vec{n}, \vec{b})$ du trièdre de Frenet peut également être défini par [24] [Far92] :

$$\left(\vec{t} = \frac{d\vec{P}/ds}{|d\vec{P}/ds|}, \quad \vec{n} = \vec{b} \wedge \vec{t}, \quad \vec{b} = \frac{(d\vec{P}/ds) \wedge (d^2\vec{P}/ds^2)}{|(d\vec{P}/ds) \wedge (d^2\vec{P}/ds^2)|} \right)$$

Un trièdre de Frenet vérifie les relations suivantes [25] :

$$\begin{pmatrix} \frac{d\vec{t}}{dc} \\ \frac{d\vec{n}}{dc} \\ \frac{d\vec{b}}{dc} \end{pmatrix} = \begin{pmatrix} 0 & \rho & 0 \\ -\rho & 0 & \tau \\ 0 & -\tau & 0 \end{pmatrix} \begin{pmatrix} \vec{t} \\ \vec{n} \\ \vec{b} \end{pmatrix}$$

[24] source : http ://www.ece.ubc.ca/ elec478/frenetframes.html (University of British Columbia)
[25] source : http ://www.sciences-en-ligne.com/momo/chronomath/chrono1/Frenet.html

Annexe E

Décomposition LU d'une matrice bande

Voici l'algorithme de décomposition LU d'une matrice M carrée de dimension n :

```
DecompositionLU(M_{n×n})
 1  U ← M
 2  L ← Id_n
 3  for i ← 1 to n
 4       do
 5          t[i] ← i
 6  for k ← 1 to n − 1
 7       do
 8          // Recherche un pivot non nul
 9          line ← cherche_pivot_non_nul(U, k)
10          if line ≠ k
11            then
12                 for j ← 1 to n
13                      do
14                         swap(U[k][j], U[line][j])
15                         swap(t[k], t[line])
16
17          // Calcul de L et U
18          for i ← k + 1 to n
19               do
20                  L[i][k] ← U[i][k] / U[k][k]
21          for i ← k + 1 to n
22               do
23                  for j ← k + 1 to n
24                       do
25                          U[i][j] − = L[i][k] ∗ U[k][j]
26
27          for j ← 1 to k + 1
28               do
29                  U[k + 1][j] ← 0
```

En supposant que M est une matrice bande de largeur $2d + 1$, démontrons que les matrices L et U conservent la même structure bande avec la même largeur.

La démonstration se réalise par récurrence.

Vérification au départ :

Au départ, la matrice U est initialisée avec la matrice M, elle a donc la même structure que M. Quant à la matrice L, elle est initialisée avec la matrice identité, elle est donc pourvue d'une structure bande de largeur 1, donc également d'une bande de largeur $2d+1$. La propriété est donc bien vérifiée pour les deux matrices au départ de l'algorithme.

Récurrence :

En supposant qu'à une itération donnée (sur l'indice k), les matrices L et U ont toujours la même structure que M (hypothèse de récurrence), voyons s'il en est toujours de même après une itération supplémentaire :

- L'algorithme commence par modifier la matrice L. On remarque que l'élément $L[i][k]$ est directement déterminé par l'élément $U[i][k]$. Ce calcul est établi pour une partie (sous la diagonale) de la colonne k courante. La matrice L hérite donc de la structure de U sur cette partie.
  Or, par hypothèse de récurrence, la matrice L est bande de largeur $2d+1$, et récupère donc localement la structure de U qui est aussi bande de largeur $2d+1$. Donc L conserve sa propriété de matrice bande de largeur $2d+1$.
- L'algorithme modifie la matrice U en faisant intervenir, des éléments du produit matriciel LU (ligne 25 de l'algorithme) :
  $$U[i][j] -= L[i][k] * U[k][j]$$
  Par hypothèse de récurrence, L est bande de largeur $2d+1$ et il vient d'être démontré que U l'est également. Or, le produit de deux matrices bandes de même largeur et une matrice bande de cette largeur. Donc U est modifiée en lui soustrayant une matrice bande de largeur $2d+1$. Comme les deux matrices (U et le résultat du produit LU) ont la même structure, la soustraction des deux garde cette structure. Ainsi, U conserve sa propriété bande de largeur $2d+1$.

Annexe F

Détermination d'une base orthonormale (u, n_1, n_2)

Soit le vecteur unitaire $\mathbf{u}$, il s'agit de déterminer deux vecteurs unitaires $\mathbf{n_1}$ et $\mathbf{n_2}$ tels que les trois vecteurs $(\mathbf{u}, \mathbf{n_1}, \mathbf{n_2})$ forment une base orthonormale.

Pour cela, nous proposons de définir arbitrairement deux vecteurs unitaires orthogonaux $\mathbf{x}$ et $\mathbf{y}$. Par exemple :

$$\mathbf{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \text{et} \quad \mathbf{y} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

Le but est alors de déterminer numériquement un vecteur orthogonal à $\mathbf{u}$ avec le plus de précision possible.

En dimension 2 , on s'appuie sur le schéma de la figure (F.1). Pour déterminer un vecteur orthogonal à $\mathbf{u}$, on utilise le produit vectoriel. Il faut donc s'assurer que le vecteur utilisé soit le plus orthogonal possible pour éviter tout problème numérique à l'approche de deux vecteurs colinéaires. Pour cela, on calcule le produit scalaire

$$\mathbf{n_1} = \mathbf{u} \wedge \mathbf{x}$$

Puis, on vérifie que le vecteur $\mathbf{u}$ n'appartient pas à la zone 1 de la figure à l'aide du test :

$$\textit{Test :} \qquad |\mathbf{n_1}| = sin(\mathbf{u}, \mathbf{x}) \geq sin(\frac{\pi}{4}) = \frac{\sqrt{2}}{2}$$

Si le test est correct, c'est que $\mathbf{u}$ est dans la zone 2, donc le choix du vecteur $\mathbf{x}$ est correct. Sinon, il faut refaire le calcul du vecteur $\mathbf{n_1}$ avec le vecteur $\mathbf{y}$:

$$\mathbf{n_1} = \mathbf{u} \wedge \mathbf{y}$$

On a alors la certitude que le test est vrai puisque $\mathbf{u}$ est dans la zone 1.

Une fois le vecteur $\mathbf{n_1}$ calculé, il faut le normaliser puis on calcule le dernier vecteur de la base par la simple relation :

$$\mathbf{n_2} = \mathbf{u} \wedge \mathbf{n_1}$$

En dimension 3 , le principe est le même, sauf que les deux zones délimitent maintenant des cônes (le reste de l'espace étant la zone 3). Donc, en appliquant le même calcul avec les deux mêmes vecteurs, on trouve une solution relativement bonne (numériquement parlant) car :

- Soit $\mathbf{u}$ est dans la zone 2 ou 3, dans ce cas, le produit vectoriel $\mathbf{u} \wedge \mathbf{x}$ vérifie le test. Donc on a trouvé $\mathbf{n_1}$ duquel on en déduit $\mathbf{n_2} = \mathbf{u} \wedge \mathbf{n_1}$.
- Soit $\mathbf{u}$ est dans la zone 1, dans ce cas, le produit vectoriel $\mathbf{u} \wedge \mathbf{x}$ ne vérifie pas le test. Donc on calcule $\mathbf{n_1}$ à l'aide du vecteur $\mathbf{y}$ par $\mathbf{n_1} = \mathbf{u} \wedge \mathbf{y}$. Comme $\mathbf{u}$ est dans la zone 1, le vecteur $\mathbf{n_1}$ verifie le test. On détermine alors le vecteur $\mathbf{n_2} = \mathbf{u} \wedge \mathbf{n_1}$.

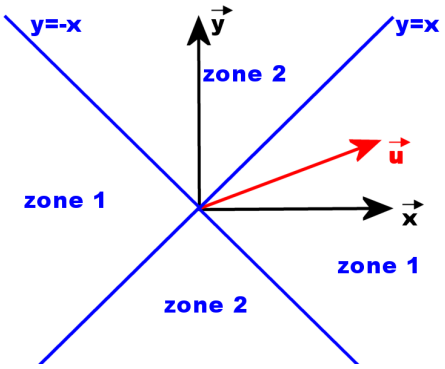


FIG. F.1 – Schéma explicatif pour le calcul d'une base orthonormale.

Annexe G

Exemples de contraintes relatives aux corps rigides

Les corps rigides ont la particularité de posséder des degrés de liberté à la fois en translation (le centre de masse **G**) et en rotation (le vecteur orientation Ω). Une contrainte particulière à un corps rigide est donc la contrainte en rotation.

Contrainte en rotation

Il s'agit de supprimer un degré de liberté en rotation d'un objet rigide. Soit **u** le vecteur lié à l'objet contraint (objet fils) dont la rotation n'est pas permise. Autrement dit, l'objet ne doit pas subir de rotation autour de ce vecteur.

Si l'on définit deux vecteurs **AB** lié à l'objet père et **CD** lié à l'objet fils (comme indiqué sur la figure (G.1)) tels que les vecteurs $(\mathbf{u}, \mathbf{CD}, \mathbf{AB})$ forment une base orthogonale de l'espace $\mathbb{R}^3$, alors la contrainte de rotation s'exprime par :

$$\mathbf{AB}.\mathbf{CD} = 0$$

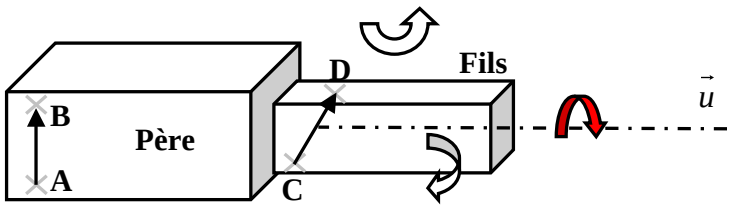


Fig. G.1 – Schéma d'une contrainte de rotation

Contrainte de glissière

Il s'agit de contraindre un axe **CD** d'un objet fils à glisser sur un axe **AB** d'un objet père comme indiqué sur la figure (G.2).

Pour cela, on détermine le plan orthogonal à l'axe **AB**. Ceci nous procure les vecteurs **u** et **v** orthogonaux, unitaires formant une base orthonormale de ce plan.

Ensuite, il suffit de contraindre le point **C** de l'objet fils à être sur l'axe **AB** du père, à l'aide de deux contraintes de translation :

$$\mathbf{AC}.\mathbf{u} = 0$$
$$\mathbf{AC}.\mathbf{v} = 0$$

Pour finir, il reste à supprimer deux degrés de liberté en rotation, autour des axes **u** et **v**. Pour cela, on applique deux fois la contrainte de rotation en choisissant le vecteur **CD** pour l'objet fils et les vecteurs respectifs **v** et **u** pour l'objet père. Donc, il suffit d'imposer au vecteur **CD** d'être orthogonal à la fois au vecteur **u** et au vecteur **v** :

$$\mathbf{CD}.\mathbf{u} = 0$$
$$\mathbf{CD}.\mathbf{v} = 0$$

ainsi le vecteur **CD** est contraint en rotation uniquement autour de l'axe choisi.

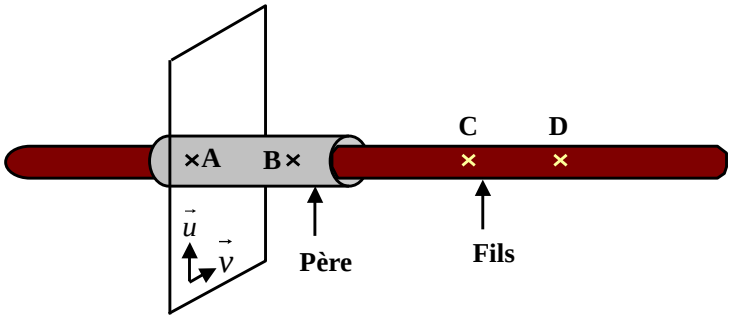


FIG. G.2 – Schéma d'une contrainte de glissière

Annexe H

Force due aux multiplicateurs de Lagrange

Les multiplicateurs de Lagrange permettent de gérer des contraintes en injectant une force directement dans la dynamique de l'objet. On souhaite déterminer avec précision la force et le point d'application de cette force dans le cas d'une contrainte vectorielle de dimension 3 pour notre modèle de spline dynamique.

Le système d'équations régissant la dynamique de la spline est (cf. équation (3.18)) :

$$\begin{pmatrix} M & 0 & L^T & L_g^T \\ 0 & 0 & 0 & L_{gs}^T \\ L & 0 & 0 & 0 \\ L_g & L_{gs} & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ \ddot{\mathbf{s}} \\ -\boldsymbol{\lambda} \\ -\boldsymbol{\lambda}_\mathbf{g} \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{0} \\ \mathbf{E} \\ \mathbf{E_g} \end{pmatrix} \tag{H.1}$$

Prenons l'exemple d'une contrainte de point glissant (cf. section (3.3.2.1)) définie par rapport à la variable libre s_k. On a montré dans la section dédiée à cette contrainte que les trois lignes (i, j et k) de la matrice L_g relatives à la contrainte s'écrivent à l'aide des coefficients $b_i(s_k(t))$:

$$L_g = \left(\begin{array}{c|c|c} \mathbf{x} & \mathbf{y} & \mathbf{z} \\ \hline \cdots & \cdots & \cdots \\ b_0(s_k(t))...b_n(s_k(t)) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & b_0(s_k(t))...b_n(s_k(t)) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & b_0(s_k(t))...b_n(s_k(t)) \\ \cdots & \cdots & \cdots \end{array}\right) \begin{matrix} \\ \\ i \\ j \\ k \end{matrix} \tag{H.2}$$

D'après l'équation (H.1), la contribution de cette contrainte dans le système dynamique est donnée par la relation :

$$L_g^T.\boldsymbol{\lambda}_\mathbf{g} = \left(\begin{array}{c|c|c} i & j & k \\ \hline b_0(s_k(t)) & \mathbf{0} & \mathbf{0} \\ \vdots & \mathbf{0} & \mathbf{0} \\ b_n(s_k(t)) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & b_0(s_k(t)) & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & b_n(s_k(t)) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & b_0(s_k(t)) \\ \mathbf{0} & \mathbf{0} & \vdots \\ \mathbf{0} & \mathbf{0} & b_n(s_k(t)) \end{array}\right) . \begin{pmatrix} \lambda_{g1} \\ \lambda_{g2} \\ \lambda_{g3} \end{pmatrix} \tag{H.3}$$

Cette expression est également la force généralisée due à la contrainte. Or, si $\mathbf{F}$ est une force qui s'excerce sur la spline au point $\mathbf{P}$ d'abscisse paramétrique s, la force généralisée relative à un degré de liberté q_i^α

vaut (cf. équation (2.15)) :

$$Q_i^\alpha = b_i(s).F^\alpha$$

On peut donc aisément remarquer que la contribution de cette contrainte est exactement la force généralisée d'une force composée des multiplicateurs de Lagrange relatifs à la contrainte $\lambda_{\mathbf{g}}$ appliquée au point d'abscisse paramétrique $s_k(t)$.

Donc, la méthode des multiplicateurs de Lagrange appliquée à une contrainte de point glissant insére la force $\boldsymbol{\lambda}_{\mathbf{g}}$ au point d'abscisse paramétrique définie par la variable libre s_k.

On remarque aisément que cette démonstration se déroule de la même manière pour une contrainte de point fixe (non glissant), puisque les coefficients intervenant dans la matrice L sont également des $b_i(s)$ (cf. section (3.1.2.1)).

Annexe I

Frottement de Coulomb

Amonton ($XVII^{eme}$ siècle) puis Charles de Coulomb ($XVIII^{eme}$ siècle) proposent une théorie mécanique sur les frottements solides.

On considère un objet $\mathcal{O}$ posé sur un plan $\mathcal{P}$. On suppose que le contact entre les deux objets est ponctuel en un point **A**. La force **F** de réaction du plan sur l'objet se décompose en une partie normale $\mathbf{F_n}$ et une partie tangentielle $\mathbf{F_t}$ comme indiqué sur la figure (I.1). La force normale contribue en partie à lutter contre la force de pesanteur de l'objet, alors que la force tangentielle va à l'encontre du mouvement, c'est donc elle qui introduit le frottement. On l'appelle la force de frottement. Donc, dans le cas d'un glissement parfait, la force de frottement est nulle. Coulomb a déterminé qu'il existe une relation entre la

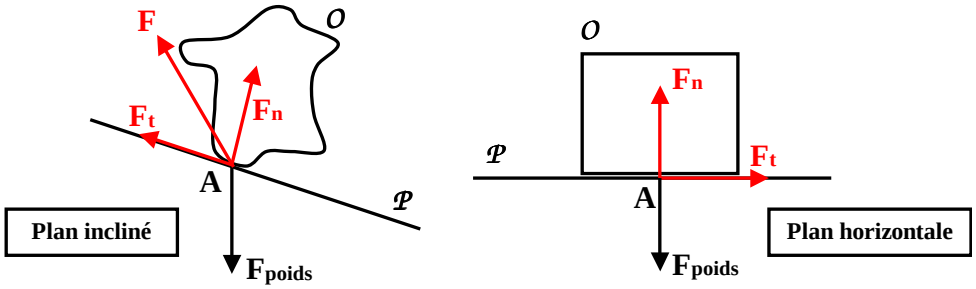


Fig. I.1 – Schéma d'un contact entre deux objets pour le frottement de Coulomb

force de frottement et la force normale. Et, que la force de frottement est relative aux matériaux mis en cause dans le contact (ici le plan et l'objet). Il propose de décomposer le phénomène de frottement solide en deux parties :

Frottement statique :

Dans ce cas, l'objet $\mathcal{O}$ est immobile sur le plan. Ainsi, la force F_t est suffisante pour contraindre l'objet à rester figé sur le plan. Cependant, si la composante normale de la force de réaction (F_n) dépasse un certain seuil, l'objet $\mathcal{O}$ se met à glisser sur le plan $\mathcal{P}$, il rentre en mouvement.

La force de frottement tangentielle maximum est :

$$F_{max} = \mu_s F_n$$

μ_s est appelé le coefficient de frottement statique ou coefficient de frottement d'adhérence. Ce coefficient dépend de la nature des objets en contact, il varie de 0.2 à 1.2.

Ainsi, si on a la relation :

$$F_t < \mu_s F_n$$

alors l'objet reste immobile, c'est le cas de l'adhérence. Sinon il rentre en mouvement, c'est le cas du glissement.

La zone balayée par le vecteur **F** lorque l'on est en adhérence est un cône. Cette zone porte le nom de cône d'adhérence (cf. figure (I.2)).

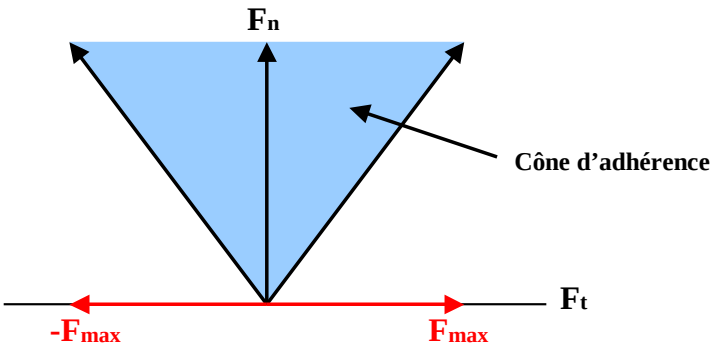


FIG. I.2 – Cône d'adhérence

Frottement dynamique :

Dans ce cas, l'objet $\mathcal{O}$ glisse sur le plan. Coulomb a montré que la force de frottement est indépendante de la vitesse de glissement, elle s'exprime comme :

$$F_t = \mu_d F_n$$

où μ_d est le coefficient de frottement dynamique, il dépend des matériaux en contact et de leur rugosité. Cette force est dans la même direction que la vitesse de glissement mais dans le sens opposé. Cependant, dans le cas pseudo-statique, l'objet ne possède pas de vitesse de glissement, cette force est alors dirigé par le vecteur $\mathbf{F_t}$.

Les deux coefficients de frottement vérifient toujours la relation :

$$\mu_s < \mu_d$$

C'est pourquoi il faut imprégner une grande force pour déplacer un objet immobile alors que s'il est en mouvement, une force moindre permet de conserver ce mouvement. Ce principe est illustré sur la figure (I.3) où la force de frottement est déterminée en fonction d'une force de traction tangentielle au plan. On remarque bien que la force de frottement dans le cas dynamique est bornée par $\mathbf{F_{max}}$.

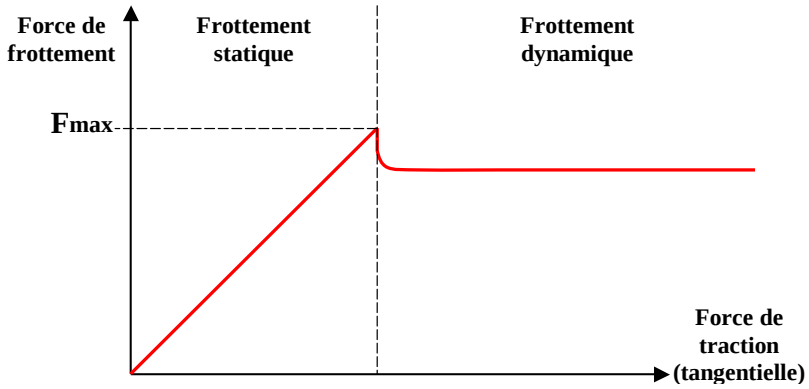


FIG. I.3 – Tracé de la force de frottement en fonction d'une traction